МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М.А. Бонч-Бруевича»

На правах рукописи

Волков Артём Николаевич

ИССЛЕДОВАНИЕ И РАЗРАБОТКА МЕТОДОВ ПОСТРОЕНИЯ ИНФРАСТРУКТУРЫ И ПРЕДОСТАВЛЕНИЯ УСЛУГ СЕТЕЙ СВЯЗИ НА ОСНОВЕ ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

2.2.15. Системы, сети и устройства телекоммуникаций

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
доктор технических наук , профессор
Кучерявый Андрей Евгеньевич

Санкт-Петербург – 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы диссертации. Мир инфокоммуникационных технологий одной ногой стоит уже в сетях следующего (пятого) поколения. Инфраструктура, предлагаемые услуги и приложения которой, отчасти реализуют утопические идеи, предложенные фантастами буквально последних 10-15 лет, не говоря уже о более ранних идеях [1]. Сети связи пятого поколения 5G/IMT-2020 и новые типы услуг являются особо актуальной темой исследований последних 5-6 лет, при этом результатом исследований и разработок мирового масштаба стал плавный переход к концепции сетей связи 2030. Данный переход произошел в связи с большим количеством вновь появившихся типов услуг, основанных на концепции Интернета Вещей. В свою очередь концепция Интернета Вещей породила частные концепции, в рамках которых были сформированы собственные требования к сетям связи. Международный Союз Электросвязи выделяет три основных «кита» сетей связи 5G/IMT-2020: масштабные межмашинные взаимодействия, высокомобильная связь и сети связи с ультрамалыми задержками. К 2021 году можно утверждать, что часть из данных направлений уже в широкомасштабном формате внедряется и носит чисто прикладной характер, а именно межмашинные взаимодействия. Так называемый тип соединения «машина-машина» уже превалирует в сетях связи над типом «человек-машина» и количество подключенных в сеть устройств превышает количество пользователей. При этом остро стоит вопрос по реализации второй, не менее важной составляющей сетей связи – сетей с ультрамалыми задержками. Набор предложенных концепций услуг в данном направлении уже плавно переходит на следующее поколение сетей связи – сетей связи 2030. Такие услуги, как телемедицина, в полной мере, Тактильный интернет, к сожалению, в условиях текущих возможностей сетей связи не реализуемы по причине ограничений как в скорости передачи информации, так и во вносимых задержках на пути следования данных. Кроме того, такой критерий,

как сверхплотность сетей, связи вносит дополнительные требования к методам построения сетевой и вычислительной инфраструктуры. Сверхплотность является одним из признаков не только сетей 5G/IMT-2020, но и всех последующих. Действительно, требование 3GPP по обеспечению необходимого уровня качества обслуживания и восприятия при размещении 1 млн терминалов на 1 кв^2/м принципиально отличается от характеристик плотных сетей, существующих в настоящее временя. В соответствии с известным прогнозом, предельное число Интернета Вещей оценивается 50 трлн устройств, что может быть достигнуто в районе 2030 года.

На основании вышесказанного и множества существующих к данному времени международных рекомендаций и стандартов можно сделать вывод, что концепция сетей связи 5G/IMT-2020 включает в себя целый комплекс концепций и технологий, а не только описывает принципы и технологии организации мобильной сети доступа [3].

В то же время, на начало 2021 г. еще не до конца завершены работы по стандартизации сетей связи пятого поколения, а в Секторе Стандартизации телекоммуникаций Международного Союза Электросвязи была создана специальная рабочая группа по исследованию и последующей стандартизации поколения сетей связи 2030. Данной рабочей группе поставлена не столь тривиальная задача – проанализировать основные тенденции в развитии сетей связи, которые появились сегодня при работах над концепцией и в процессе начала реализации сетей 5G/IMT-2020, и определить основные характеристики и направления стандартизации сетей и услуг. Например, как уже было выше отмечено, одно из направлений – Тактильный Интернет (ТИ). И для реализации ТИ были выдвинуты требования по задержке, которая не должны превышать 1мс [4].

В данный момент параллельным «курсом» и при этом напрямую влияющим на развитие сетей связи и услуг является направление «Искусственный Интеллект в сетях связи». Тематика Искусственного Интеллекта (ИИ) в сетях связи в мире

научных исследований появилась сравнительно недавно и вызывает все больший интерес со стороны как научных работ и проектов, так и бизнеса, и производства. Причиной большого интереса к применению технологий ИИ в сетях связи является надежда на решение множества трудноразрешимых задач сетей связи 5G и последующих поколений [3]. Стоит отметить, что в Секторе Стандартизации МСЭ достаточно активно ведутся работы по разработке рекомендаций в области ИИ в сетях связи. При этом на базе 13 Исследовательской Комиссии (ИК), в декабре 2020 года была создана специальная Фокус-группа по исследованию и стандартизации автономных сетей, в открытии которой принимал участие также и автор данной диссертационной работы. Термин «Автономные сети связи» носит прикладной характер, с точки зрения стандартизации и технической реализации сетей связи с ИИ. Актуальность направления «Искусственный Интеллект в сетях связи» можно также подтвердить существующими международными событиями. Например, в 2020 году в рамках “AI for Good Global Summit” прошел международный конкурс «AI/ML in 5G Challenge”, организуемый МСЭ под эгидой ООН, который собрал множество стран, известных исследовательских университетов и центров, а также крупных компаний таких, как: Vodafone, China Telecom, Cisco и другие. В рамках данного конкурса на базе СПбГУТ и лаборатории «Искусственного Интеллекта в сетях связи» кафедры Сетей связи и Передачи данных был открыт региональный хост от России, где были сформулированы исследовательские задачи по идентификации трафика и его дальнейшему прогнозированию.

Степень разработанности темы. В области Интернета Вещей, сетей связи пятого поколения, Искусственного Интеллекта в сетях связи существует немало работ отечественных и зарубежных ученых: В.М. Вишневского, Б.С. Гольдштейна, В.Г. Карташевского, А.Е. Кучерявого, А.И. Парамонова, К.Е. Самуйлова, В.К. Сарьяна, С.Н. Степанова, Ю.В Гайдамаки, Р.В. Киричка, Е.А. Кучерявого, Д.А. Молчанова, В.О. Тихвинского, А.С.А. Мутханны, J. Andrews, J. Araniti, A.A.A. Ateya, M.Z. Faten и других.

Стоит отметить, что на данный момент большую часть составляют работы по сетям связи пятого поколения, новым услугам, беспроводным и mesh-сетям, технологиям Интернета Вещей. Однако вопрос Искусственного Интеллекта в сетях связи является достаточно свежей темой исследований. К примеру, статья «Искусственный интеллект в сетях связи» [3], описывающая статус данного научно-исследовательского направления и в том числе на уровне разработки международных рекомендаций и стандартов, вышедшая в известном журнале «Электросвязь», индексируемая ВАК только в начале 2021 года. Таким образом, проблемам построения сетей связи и услуг на основе технологий Искусственного Интеллекта, с учетом имеющихся наработок в области ИИ, до сих пор не было уделено достаточного внимания. В рамках данного направления выделяют целый ряд высокоуровневых задач, одними из которых являются задачи идентификации трафика в сетях связи, прогнозирования нагрузки на сеть и ее устройства, эффективного распределения вычислительных ресурсов и задач в сети связи. Данные задачи требуют разработки соответствующих методов на основе технологий ИИ, учитывая особенно требования качества обслуживания для перспективных услуг сетей связи, необходимого быстродействия служебных сервисов и знания прогнозируемых данных. Выше озвученные вопросы и определяют цель, задачи, объект и предмет настоящей диссертационной работы.

Объект и предмет диссертации. Объектом исследования являются сети связи пятого поколения 5G/IMT-2020, а предметом исследования – методы построения этих сетей и услуг на основе технологий ИИ.

Цель и задачи диссертации. Целью диссертационной работы является исследование и разработка методов построения инфраструктуры и предоставления услуг сетей связи с использованием технологий Искусственного Интеллекта.

Для достижения поставленной цели в диссертации последовательно решаются следующие задачи:

- анализ концепций современных и перспективных сетей связи, учитывая долгосрочные до 2030 года перспективы;
- анализ тематики «Искусственный Интеллект в сетях связи», учитывая прогресс стандартизации автономных сетей в МСЭ-Т;
- анализ методов машинного обучения и Больших данных для задач мониторинга и управления трафиком в сетях связи;
- разработка метода идентификации трафика услуг в сетях связи пятого и последующих поколений;
- разработка структуры и метода взаимодействия туманных и граничных вычислений с поддержкой микросервисной архитектуры услуг;
- разработка метода прогнозирования нагрузки на контроллеры программно-конфигурируемых сетей.

Научная новизна. Научная новизна полученных результатов состоит в следующем:

- Разработан метод идентификации трафика услуг в сетях связи пятого и последующих поколений, позволяющий исключить внесение дополнительных задержек и изменение структуры потоков.
- Предложена структура взаимодействия туманных и граничных вычислений, отличающаяся тем, что услуги реализуются в виде микросервисной архитектуры программного обеспечения с учетом динамичности пользователей.
- Разработан метод взаимодействия туманных и граничных вычислений, обеспечивающий функционирование микросервисной архитектуры услуг с возможностью живой миграции, позволяющий уменьшить время выполнения функции микросервиса путем рационального выбора устройства Туманных вычислений.

- Разработан метод прогнозирования нагрузки на контроллеры Программно-конфигурируемых сетей, позволяющий исключить зависимость программного обеспечения мониторинга от особенностей АПК производителей.

Теоретическая и практическая значимость диссертации.

Теоретическая значимость диссертационной работы состоит, прежде всего, в обосновании предстоящей децентрализации сетей связи, вычислительных облачных систем обработки данных и реализации услуг, а также модернизации архитектур программного обеспечения услуг в сторону микросервисного подхода, с появлением сетей связи с ультрамалыми задержками. Кроме того, разработан метод идентификации трафика в программно-конфигурируемых сетях, основанный на аналитике метаданных потоков и алгоритмах машинного обучения, который позволяет не вносить дополнительных задержек в трафик на уровне передачи данных. Разработан также метод прогнозирования нагрузки на контроллер программно-конфигурируемой сети, который позволяет реализовывать мониторинг нагрузки и ее прогноз на уровне приложений программно-конфигурируемой сети, что в свою очередь позволяет уйти от зависимости мониторинговых решений как от производителя АПК контроллера программно-конфигурируемой сети, так и от зависимости от операционной системы, на которой развернуто программное обеспечение контроллера. В дополнение предложена структура и метод взаимодействия туманных и граничных вычислений, при этом обеспечивающие поддержку микросервисной архитектуры услуг с возможностью живой миграции и учетом динамики пользователей.

Практическая значимость диссертационной работы подтверждается актом внедрения и состоит в создании научно-обоснованных рекомендаций по планированию сетей связи пятого поколения в условиях внедрения сетей с ультрамалыми задержками и Искусственного интеллекта в сетях связи.

Полученные в диссертационной работе результаты использованы в ФГУП НИИР при выполнении государственных контрактов по научно-техническому и

методическому обеспечению выполнения Министерством цифрового развития, связи и массовых коммуникаций функций Администрации связи Российской Федерации в Секторе стандартизации электросвязи Международного союза электросвязи в работах по разработке стандартов для сетей связи пятого и последующих поколений с учетом использования технологий искусственного интеллекта и в Санкт-Петербургском Государственном Университете Телекоммуникаций им. проф. М.А. Бонч-Бруевича (СПбГУТ) при чтении лекций и проведении практических занятий по дисциплинам “Интернет Вещей и самоорганизующиеся сети”, "Искусственный интеллект в сетях и системах связи", "Интернет вещей", "Системы, сети и устройства телекоммуникаций", а также при выполнении НИР "Исследование проблемных вопросов сетевой поддержки перспективных услуг сетей связи 2030, включая телеприсутствие, и путей их решения, в том числе на основе технологий искусственного интеллекта, при подготовке отраслевых кадров".

Методология и методы исследования. Для решения поставленных в диссертации задач использовались методы машинного обучения, теории оптимизации, имитационного и эмуляционного моделирования. Эмуляционное моделирование проводилось на базе модельной программно-конфигурируемой сети в рамках кафедры Сетей связи и передачи данных СПбГУТ. Имитационное моделирование выполнялось с помощью языка программирования Python версий 2.7, 3.4 с необходимыми программными пакетами и библиотеками обработки данных.

Основные положения, выносимые на защиту:

1. Метод мониторинга, идентификации трафика услуг в сетях связи пятого и последующих поколений, основанный на аналитике метаданных потоков и алгоритмах Машинного обучения, позволяющий исключить внесение дополнительных задержек и изменение структуры потоков.

2. Структура и метод взаимодействия туманных и граничных вычислений, основанные на алгоритмах Больших данных, обеспечивающие функционирование микросервисной архитектуры с возможностью живой миграции, позволяющие

уменьшить время выполнения функции микросервиса за счет рационального распределения ресурсов на величину до 70%.

3. Метод прогнозирования нагрузки на контроллеры Программно-конфигурируемых сетей на основе технологий Искусственного интеллекта, использующий для оценки нагрузки анализ метаданных служебных потоков, что позволяет исключить зависимость программного обеспечения мониторинга от особенностей АПК производителя.

Степень достоверности и апробация результатов. Достоверность результатов диссертации подтверждается корректным применением математического аппарата, результатами имитационного моделирования с использованием пакетов и библиотек языка программирования Python версий 2.7 и 3.4, результатами эмуляционного моделирования на базе модельной программно-конфигурируемой сети лаборатории кафедры ССиПД СПбГУТ, а также обсуждением результатов диссертационной работы на конференциях и семинарах, публикаций основных результатов диссертации в ведущих рецензируемых журналах.

Основные результаты диссертационной работы докладывались и обсуждались на следующих международных и российских конференциях и семинарах: международный глобальный саммит «Искусственный интеллект во Благо» 2020 года (виртуальный формат, Женева), проводимый МСЭ под эгидой ООН, а также на заседании 13 ИК в рамках 21 вопроса МСЭ-Т 1-12 марта 2021 года (виртуальный формат, Женева), конференциях DCCN (International conference on Distributed Computer and Communication Networks: Control, Computation, Communications) и New2Wan (International Conference on Next Generation Wired/Wireless Networking).

Публикации по теме диссертации. По теме диссертации опубликовано 8 научных работ, из них: 4 статьи в рецензируемых научных журналах; 2 статьи в изданиях, индексируемых в международных базах данных; 2 – в других изданиях и материалах научных конференций.

Соответствие паспорту специальности. Содержание диссертации соответствует пунктам 3, 4, 11 и 14 паспорта специальности 05.12.13 – Системы, сети и устройства телекоммуникаций.

Личный вклад автора. Основные результаты диссертации получены автором самостоятельно.

ГЛАВА 1. АНАЛИЗ КОНЦЕПЦИЙ СОВРЕМЕННЫХ И ПЕРСПЕКТИВНЫХ СЕТЕЙ СВЯЗИ

1.1. Введение

Всем знаком закон Мура о росте вычислительной мощности микроэлектроники. Однако скорость роста технологий, возникающих на основе открывшихся возможностей микроэлектроники еще не формализована и не оценена. На протяжении 20-го века, основной связью считалась телефонная связь, где использовалась коммутация каналов, понятная и четкая логика и алгоритмика расчета нагрузки в сетях связи. Однако, за последние 20 лет, технологии связи стали меняться со скоростью, не позволяющей уже самим специалистам области знать всю теорию, инструменты, технологии, как это было ранее. С приходом пакетной коммутации, а затем уже технологий создания программного обеспечения начался тот «большой взрыв» (технологический взрыв), последствия которого мы имеем сейчас. И как мы знаем из физических основ «большого взрыва», расширение космического пространства не останавливается до сих пор, так и в технологиях сетей связи. На данный момент уже ведется подготовка «узкоспециализированных» специалистов, в каждую область сетей связи. При этом, любой специалист должен сейчас обладать знаниями не только в области сетей связи, но и в области разработки и обслуживания программного обеспечения.

Сети связи 5G/IMT-2020 стали одной из масштабных концепций, приход которой изменил не только осознание возможностей передачи данных на текущих технологиях физической среды, но и перевернул в сознании понимание что такое услуга сетей связи и что она может дать пользователю, оператору, государству,

экономике. В современном мире бюджет ряда международных ИТ-компаний может как в отдельности, так и в объединении с другой такой же компанией, поспорить с объемами бюджетов целых стран и не только. Одним из ценных «капиталов», имеющихся у данных компаний, является внимание и время людей-пользователей их сервисов. В результате действий некоторых компаний в 2021 году, можно сделать вывод, что данные ИТ-компании уже представляют собой отдельные «государства», которые ведут политические действия не только внутри стран, но и на международном уровне. Таким образом, исходя их принципов информационного общества, определенных философами последних лет, можно сделать вывод, что информационное общество уже наступило. С целью не упустить и иметь свои «рычаги», большинство развитых и развивающихся стран преследуют концепцию так называемой в Российской Федерации – цифровой экономики.

Все эти глобальные изменения, не только в области конечных услуг пользователям, повышения их качества и тому подобное, произошли благодаря резкому росту технологий сетей связи. Можно сказать, что в Интернет пространстве, где главным ресурсом, на данный момент, является – внимание пользователя, создаются уже совсем другие сущности, неподвластные законам физического реального мира. На одних только продажах внимании пользователя строятся неосязаемые бюджеты маркетинга и продвижения Интернет-ресурсов. Здесь же можно привести пример появления криптовалют и использующиеся в них принципы блокчейна, виртуальных устройств Интернета Вещей, цифровых аватаров пользователей (ожидаются в сетях связи 2030), телепортации информации, наносетей и других, кажущимися ранее утопическими идеями фантастов.

Возвращаясь к концепции цифровой экономики, суть которой можно выразить как обеспечение полной прозрачности всех финансовых и производственных процессов, необходимо определить, что для ее полной реализации требуется реализация в полном масштабе Интернета Вещей, и как в следствие, сетей связи

5G/IMT-2020 и сетей уже последующего поколения – сетей связи 2030, которые должны будут закрыть вопросы, не разрешенные в сетях пятого поколения и принести новые возможности. Таким образом, можно определить, что сети связи 5G/IMT-2020 это не только новые технологии в мобильных сетях (ядра/технологий радиосвязи и т.п.), но и в опорных (ядро/агрегация/доступ) сетях, услугах. Стоит также отметить, что при росте трафика, увеличении количества типов услуг, имеющих новые, гораздо более жесткие требования, в научном и инженерном сообществе пришли к пониманию необходимости внедрения технологий Искусственного Интеллекта (ИИ). Под ИИ подразумевается область математического знания, отвечающая на вопросы машинного обучения и эффективной обработки Больших данных. Необходимость применения данного класса технологий обусловлена потребностью в эффективной обработке большого количества метрических данных, собираемых мониторинговыми системами сетей связи и вычислительными (облачными) системами, на базе которых разворачиваются и реализуются современные услуги связи. На основе этих данных, в сетях связи 5G/IMT-2020 и в будущих сетях связи будут приниматься решения о коммутации трафика, его обработке и распределении.

1.2. Анализ основных технологий сетей связи 5G/IMT-2020

1.2.1. Концепция сетей связи 5G/IMT-2020 и основные направления услуг

Сети связи 5G/IMT-2020 должны создать экосистему для технических и бизнес-инноваций [1]. Как уже ранее упоминалось, ожидается, что сети IMT-2020 дадут возможность эффективно и экономично пускать множество новых услуг. При этом, стоит отметить, что обозначение «5G», ранее обозначающее технологический этап

мобильных сетей, в данном поколении, де-факто на международном уровне отражает новую эру сетей связи и сервисов в целом, затрагивая изменение в том числе и в технологиях ядра сетей, принципов предоставления услуг и т.д. Аббревиатура «G» дается консорциумом 3GPP (3rd Generation Partnership Project), при этом аббревиатуру IMT-2020 (International Mobile Telecommunications) использует МСЭ-Т. На момент 2021 году, в МСЭ-Т существует несколько Исследовательских комиссий (ИК), работа которых полностью посвящена разрешению вопросов стандартизации в области технологий инфраструктуры и сервисов 5G/IMT-2020. Стоит отметить, работы следующих ИК:

- 11 ИК: Фокусируется на требованиях к сигнализации, протоколах, спецификаций испытаний и борьбе с контрафактной продукцией. В рамках данной ИК ведутся особо активные работы по разработке рекомендаций в области протоколов ИВ их совместимости, архитектур тестирования, идентификации устройств Интернета Вещей. Стоит также обратить внимание на рекомендации, описывающие методы тестирования контроллеров Программно-конфигурируемых сетей (ПКС).
- 13 ИК: Фокусируется на будущих сетях (Future Networks) и в частности IMT-2020, облачным вычислениям и доверенным сетевым инфраструктурам. В рамках данной ИК рассматриваются инфраструктурные рекомендации: архитектуры, фреймворки (структуры), правила и принципы построения. Стоит отметить одно из самых актуальных направлений, которым также занимаются в рамках данной ИК – Искусственный Интеллект в сетях связи.
- 20 ИК: Фокусируется на концептуальных вопросах Интернета Вещей, умных городах и сообществах. В рамках данной ИК часто рассматриваются вопросы разработки рекомендаций концептуального плана, а также приложениях глобальной концепции Интернета Вещей, например – Виртуальная и Дополненная реальность (с англ. Augmented & Virtual Reality).

Стоит также обратить внимание, что помимо основных ИК, существует еще один формат партнерства и совместной разработки как – Фокус группа. Какие именно задачи и в каком формате (рекомендательном, обозревательном и т.д.) Фокус группа будет выполнять определяются в процессе ряда сессий по вопросу открытия данной Фокус-группы. К примеру, в рамках 13 ИК, в декабре прошлого года, от инициатора Rakuten Mobile (Япония) совместно с приглашенными сторонами-единомышленниками, в числе которых были также СПбГУТ им. проф. М.А.Бонч-Бруевича, Ростелеком, был подан запрос об открытии специализированной Фокус группы по так называемым Автономным сетям связи. Автономные сети (с англ. Autonomous Networks) – термин, который более точно отражает сети связи с внедренным Искусственным Интеллектом. Словосочетание «Искусственный Интеллект в сетях связи» несет больше маркетинговый/рекламный характер. Таким образом, в декабре 2020 года, в результате ряда проведенных, достаточно «жарких» сессий-заседаний всех сторон процесса, Фокус группа по автономным сетям была открыта и начала свой рабочий путь с 2021 года. Работа данной фокус-группы в первую очередь направлена на проведение аналитических работ, связанных с тематикой Автономных сетей, разработку проектов-рекомендаций, которые будут подаваться от имени Фокус-группы в основную ИК 13 по теме ИИ в сетях связи.

Возвращаясь к вопросу инфраструктуры и услуг сетей связи 5G/IMT-2020, стоит отметить, что предыдущие изменения (поколения) были в первую очередь связаны с появлением сотовых сетей связи третьего поколения 3G (1999 год) и пакетных сетей связи общего пользования (2000 год). Кроме того, стоит учитывать, что ранее сети связи делились на три группы технологий: сети передачи данных (СПД), сети подвижной (мобильной) связи, а также телефонные сети общего пользования (ТФоП). На сегодняшний момент, сети пятого поколения призваны интегрировать в себе все достижения мобильных и фиксированных сетей связи, обеспечить скорость передачи данных в 10 Гбит/с и выше, а также приблизить возможности новых структур

облачных вычислений (Fog-туманные и MEC-пограничные) непосредственно к пользователю.

5G/IMT-2020 являются гетерогенными, то есть объединяющими в себе множество различных типов сетей, от традиционных фиксированных сетей связи общего пользования и мобильных, до летающих и сенсорных [2]. В англоязычной литературе такой тип сетей получил название HetNet (Heterogeneus Networks) [2]. Однако изначально свойство гетерогенности было замечено при исследовании взаимодействия систем длительной эволюции (с англ. LTE – long Term Evolution) и сенсорных сетей. Системы IMT-2020 – это не только средство связи для пользователей, но и инструмент, содействующий развитию других отраслей, таких как медицина, транспорт, образование и другие [5]. МСЭ-Т определил в рекомендации МСЭ-Р М.2083-0 следующие сценарии использования 5G/IMT-2020 [5]:

- *Усовершенствованная подвижная широкополосная связь. (с англ. EMB – Enhanced Mobile Broadband)* Спрос на услуги подвижной широкополосной связи будет расти и дальше.
- *Сверхнадежная передача данных с малой задержкой. (с англ. URLLC – Ultra-reliable and Low Latency communications).* В данном сценарии использования предъявляются жесткие требования к таким показателям, как пропускная способность, задержка и готовность сети. В данном случае можно привести пример: дистанционная хирургия, автоматизация распределения энергии в «умных электросетях», безопасность на транспорте, в том числе автономном транспорте.
- *Крупномасштабные системы межмашинной связи. (с англ. MMC – Massive machine type communications).* Данный сценарий характеризуется большим количеством подключенных устройств на кв. км.

МСЭ определило данные сценарии как основные три «кита» сетей связи 5G/IMT-2020, в каждом из векторов которого будет происходит как частное развитие, так и совместное с другими векторами, что должно принести достаточно большой

положительный синергетический эффект. Результаты синергии данных направлений развития сетей связи уже видны в настоящее время: появление беспилотного транспорта (в тестовом режиме), появление систем типа «Умный город», и другие. Проработанность существующих решений, можно сказать, находится на начальном уровне, но оценить сам эффект уже возможно сейчас.

Рисунок 1.2.1 – сценарии использования IMT-2020 и далее [5]

В качестве опорных сетей МСЭ в рекомендации [5] предлагает использовать концепции Программно-конфигурируемых сетей (с англ. SDN – Software-Defined Networking) и Виртуализации сетевых функций (с англ. NFV – Network Function Virtualization) в пункте 2.3.2 Сетевые технологии – «Для IMT-систем будущего потребуются более гибкие сетевые узлы, настраиваемые на основе организации сетей с программируемыми параметрами, а также архитектуры и виртуализации сетевых функций в целях оптимальной обработки функции узлов и повышения эксплуатационной эффективности сетей». Согласно рекомендации МСЭ-Т Y.3100 “Термины и определения для IMT-2020 сетей» от сентября 2017 года, IMT-2020 (на основе [5]) – это системы, системные компоненты и связанные с ними аспекты, которые поддерживают предоставление гораздо более расширенных возможностей,

чем те, которые описаны в рекомендации (МСЭ-Р М.1645 – «Структура и общие цели будущего развития IMT-2000 и последующих систем».) Рекомендации МСЭ-Р М.2083-0 “Видение IMT - Структура и общие цели будущего развития IMT на 2020 год и далее” от сентября 2015 отображены те расширенные возможности, которые предоставляют сети IMT-2020. Согласно той же рекомендации, сети IMT-2020 служат средством коммуникации для людей и машин, помогают в развитии других отраслей промышленности. Учитывая тенденции развития технологии ИКТ, сети IMT-2020 должны внести свой вклад в следующее:

- *Беспроводная инфраструктура в качестве сети доступа:* широкополосная связь приобретает тот же уровень важности, как и доступ к электроэнергии. В будущем частным и корпоративным клиентам будет предоставлен широкий спектр услуг, начиная от информационно-развлекательных сервисов и заканчивая новыми для промышленного и профессионального применения (промышленный Интернет Вещей).
- *Новый рынок ИКТ:* Развитие будущих ИКТ (информационно-коммуникационных) систем будет способствовать появлению интегрированной индустрии, которая в свою очередь станет движущей силой для экономики услуг. Относительно существующих интеграционных ИКТ проектов в Российской Федерации можно отметить национальный проект «Цифровая экономика», в рамках которого также необходимо внедрение сетей связи 5G/IMT-2020 для обеспечения необходимого качества и ресурсных возможностей, в том числе устойчивости ИКТ-инфраструктуры.
- *Преодоление цифрового разрыва:* IMT-2020 должны продолжать также развиваться в сторону преодоления так называемого «цифрового разрыва», предоставления услуг связи следующего поколения в любой точке мира, где есть подключение.

- *Новые способы коммуникации:* IMT-2020 должны предоставить связь в «в любое время», «в любом месте», «с любого устройства».
- *Новые формы обучения:* IMT-2020 может изменить методы обучения, предоставляя легкий доступ к цифровым учебникам или облачным хранилищам знаний в Интернете (электронное обучение, электронное здравоохранение, электронная коммерция и так далее).
- *Повышение энергоэффективности:* IMT-2020 реализует новые типы связи, такие как M2M (межмашинные коммуникации), которые позволят предоставить новые типы услуг для оптимизации производственных и бизнес-процессов.
- *Социальные изменения:* IMT-2020 является одним из базовых движущих сил перехода к постиндустриальному (информационному) обществу.
- *И другие, определенные в МСЭ-Р М.2083-0* [5].

По вышеперечисленным вкладам, которые должны быть внесены сетями связи пятого поколения, можно прийти к выводу, что уже на момент 2021 года заметны достаточно немалые изменения как в технических аспектах, так и в социальных, при этом полноценного внедрения сетей связи 5G/IMT-2020 еще не состоялось. Внедрение находится на стадии пилотных проектов во всем мире и в том числе в Российской Федерации. На основе анализа проявления «эффектов» сети и услуг 5G/IMT-2020, которые определены в рекомендации МСЭ, можно определить следующие причины:

- Развитие сервисов и услуг происходит гораздо быстрее изменения сетей;
- Появление новых сервисов (не операторского уровня), которые ранее не были определены, в том числе можно отнести и смешанного типа услуги;
- Доступность вычислительной базы и сетевых ресурсов для обычных пользователей сети Интернет, что создало почву для «бума» различного рода стартапов и микро-компаний в ИТ-секторе. Данные решения воспринимают сеть как набор сетевых и вычислительных ресурсов, где необходимо платить лишь сервисному

провайдеру за определенный пул вычислительных ресурсов, где развернут свой продукт.

- Опережающее внедрение вычислительных технологий нового поколения быстрее внедрения сетей связи. Например, вычислительные облака предоставляющие услуги на всех уровнях: SaaS (Software as a Service – с англ, программное обеспечение как услуга), PaaS (Platform as a Service – с англ., платформа как услуга), IaaS (Infrastructure as a Service – с англ., Инфраструктура как услуга). Кроме того, крупные игроки данного рынка «пошли дальше» и уже предоставляют новое поколение облаков и подходов к разработке коммерческого программного обеспечения, а именно – Сервлет-разработка. Данный подход ускоряет реализацию задач, обеспечивая программиста всеми необходимыми инструментами, избавляя его от вопросов работы с «инфраструктурным программным обеспечением», совместимостью фреймворков или библиотек, предоставляя вычислительные ресурсы и среду разработки прямо в облаке в формате «подписки»;
- Особо резким катализатором стала новая инфекция Covid-19 в 2020 году, в результате которой финансовые модели, бизнес-модели, производственные модели терпели свое изменение буквально «через боль». Однако в течении года работы формата «удаленного участия», произошли изменения в формах образования, процессах предприятий (повышение эффективности), а также и в социальной сфере. К примеру, уже сейчас человек имеет возможность «замкнуть» его потребности в рамках своего дома, получая информационные и денежные ресурсы в цифровом виде, а физические ресурсы (продукты питания, энергию и т.д.) в формате доставки, при этом осуществляя профессиональную деятельность также в формате «удаленного участия» через сеть связи.

Так, можно сделать вывод, что события последнего года стали катализатором ускорения внедрения нового типа услуг, для которых необходимо обеспечить новые уровни качества обслуживания, устойчивости и емкости сети.

Ключевые требования сети 5G/IMT-2020. Ключевыми принципами проектирования сети 5G/IMT-2020 являются гибкость, масштабируемость и разнообразие услуг. Следующие критерии считаются ключевыми возможностями сетей пятого поколения [5]:

- *Пиковая скорость для пользователя:* Максимальная достижимая скорость передачи данных в идеальных условиях для каждого пользователя устройства до 10 Гбит/с;
- *Задержка:* должна быть обеспечена минимально возможная задержка (например, для услуг Тактильного Интернета – не более 1мс);
- *Мобильность:* необходимо обеспечить выполнение требований QoS при высокой скорости передвижения объекта/пользователя;
- *Высокая плотность подключенных устройств;*
- *Энергоэффективность:* необходимо обеспечить энергоэффективность как на стороне пользователя, так и на стороне оператора;
- *Площадь пропускной способности:* Пиковая пропускная способность сети на единице пространства, то есть Мбит/с/м^2. IMT-2020, как уже ранее говорилось, предполагает реализацию концепции Интернета Вещей, что способствует увеличению количества подключенных устройств на единицу пространства, что влечет увеличение нагрузки на оборудование в несколько раз.

В результате тех требований, которые поставлены перед сетями пятого поколения, с целью достижения выполнения жесточайших критериев качества к новых услугам, например, таким как, Тактильный Интернет, требуется проработать вопрос миграции сетевых технологий на те технологические решения, которые смогут обеспечивать необходимую стабильную работу сетевой инфраструктуры, ее масштабируемость, модульность, высокую абстракцию уровня управления, что в конечном итоге позволить создать единую гибкую систему контроля и управления через стандартные программные интерфейсы. Также одним из принципов, согласно

которым разрабатывается инфраструктура сетей пятого поколения - «Network Slicing», где «Network Slice», иначе говоря «Сетевой срез» - это логическая сеть, которая обеспечивает определенный набор сетевых и вычислительных возможностей, а также характеристик (согласно МСЭ-Т Y.3100), то есть по сути является «программируемой единицей», которая позволяет логически изолировать предоставляемые сетевые функции, например: «slice» для организации услуги VoIP, E-health и так далее.

1.2.2. Анализ концепции Программно-конфигурируемых сетей и протокола OpenFlow

Современные пакетные сети состоят из множества отдельных элементов сети, выполняющих соответствующие функции: маршрутизаторы, коммутаторы, балансировщики нагрузки, NAT (Network Address Translation), брэндмауэры и другие. Концепция Программно-конфигурируемых сетей предлагает уйти от такой тенденции развития пакетных сетей, выполнив переход от отдельных сетевых элементов сети и платформ в целом, к программируемым «сущностям». Обеспечив управление инфраструктурой с помощью стандартных программных интерфейсов, можно добиться эффективной передачи транспортных потоков, ввода новых протоколов на сеть и конечной цели - организации «Интеллектуального управления», посредством реализации так называемого «Искусственного Интеллекта». При этом прозрачная и полная программируемость сетевых и вычислительных сущностей является достаточным и необходимым условием реализации ИИ в сети связи. Каждый маршрутизатор в отдельности, особенно высокопроизводительный операторского уровня является дорогим устройством. При росте сети, объемов трафика, типов услуг, необходимо модернизировать сеть, добавлять новые дорогостоящие устройства для

обработки дополнительных вычислений, обеспечения выполнения новых функций. Что в итоге приводит к потребности постоянного обновления устройств, их программного обеспечения, а это дополнительные издержки, которые вынужден нести владелец сети. Это хорошая бизнес-модель для компаний-производителей: при покупке достаточного количества сетевых устройств владельцам сетей необходимо регулярно докупать обновления ПО и модернизировать свое оборудование, чтобы обеспечить требуемые критерии качества предоставляемых услуг на сети, развивать новые услуги и в конечном итоге — не терять клиентов.

В соответствии с этим, можно определить следующие проблемы современных пакетных сетей:

- *научно-технические:* на данный момент развития сетей связи очень сложно и отчасти даже невозможно контролировать и надежно предвидеть поведение сетей связи. Сложность управления такими сетями.
- *Экономические:* стоимость современного сетевого высокопроизводительного оборудования, в том числе частые дорогостоящие обновления программного обеспечения. Необходимость в высококвалифицированных специалистах для обеспечения работы сети, сертифицированными под определенное вендорское оборудование.
- *Проблемы развития* — в архитектуре современных сетей имеются существенные барьеры для быстрого создания пилотных зон и создания новых услуг.

Разрешением кризиса развития пакетных сетей связи стало появление принципиально нового подхода к их построению – Программно-конфигурируемые сети. Изначально, концепцию новой сетевой архитектуры Программируемых сетей предложили в 2007 году сотрудники Стэнфордского университета. В первые годы, сети SDN развивались преимущественно в научной лаборатории Стенфорда и Беркли и в промышленно значимых масштабах их еще никто не пробовал. Исследователи из Стенфорда и Беркли предположили, что в пакетных сетях возможно разделить

функции управления и передачи данных, которые в данный момент реализованы в виде аппаратно-программного решения.

В основе концепции SDN лежит стремительно развивающийся открытый стандарт OpenFlow — стандарт протокола взаимодействия контроллера сети (устройства управления) и коммутаторов сети. Стоит отметить, что первые версии также, как и сама концепция, зарождались в стенах исследовательских центров Стенфорда и Беркли, вспоследствии выродившись в сообщество ONF (Open Networking Foundation). Поэтому при рассмотрении вопроса стандартизации и развития архитектурных решений и протоколов требуется обратить внимание также и на развитие концепции SDN сообществом ONF, которое на момент 2021 года вошло в состав консорциума «The Linux Foundation».

Согласно определению ONF, Программно-конфигурируемая сеть — это динамичная, управляемая и адаптируемая сетевая инфраструктура, в которой разделены уровни управления сетью и передачи данных, что обеспечивает программное управление сетью и абстрагирование уровня сетевой инфраструктуры от уровня приложений и сетевых услуг (сервисов).

Рабочая группа по инженерным задачам Интернет (с англ., IETF — Internet Engeneering Task Force), дает следующее определение в RFC 7426: Программно-конфигурируемая сеть — это подход к построению сетей, обеспечивающий прямое управление ресурсами и сетями, а также их распределение за счет добавления собственных средств обработки, администрирования и программного управления, посредством открытых сетевых и программных интерфейсов и абстрагирования от уровня сети.

В тоже время Международный Союз электросвязи определяет Программно-конфигурируемую сеть, как технологию построения сетей, которая позволяет реализовать централизованный, программируемый уровень управления и изоляцию уровня данных. В настоящее время исследованиями в области SDN в МСЭ-Т занимаются ИК 13 (архитектуры и функциональные требования к SDN, ИИ в сетях

связи) и ИК 11 (эталонные архитектуры сигнализации SDN, требования к сигнализации и протоколы SDN, включая протоколы взаимодействия, а также тестирование на соответствие и взаимодействие). Интерес к концепции Программно-конфигурируемых сетей проявляют также ИК 15 (транспорт в SDN) и ИК 17 (безопасность в SDN). В МСЭ-Т ИК 13 SDN занимается РГ 3/13 «SDN и сети будущего» в рамках исследовательского вопроса (ИВ) 11 «Развитие сетевых технологий и услуг, ориентированных на пользователя, и взаимодействие с перспективными сетями, включая SDN» и ИВ 14 «Сети SDN и функционирование перспективных сетей с учетом оказываемых услуг», а также РГ 2/13 «Облачные вычисления и основные возможности» в части прикладных вопросов — QoS, безопасность, мобильность. На данный момент, основные рекомендации Исследовательских групп:

- Y.3300 «Framework of software-defined networking» - Структура Программируемых сетей;
- Y.3301 «Functional requirements of software-defined networking»;
- Y.3302 «Functional architecture of software-defined networking»;
- Y.3320 «Requirements for applying formal methods to software-defined networking»;
- Y.3321 «Requirements and capability framework for NICE implementation making use of software-defined networking technologies»;
- Y.3011 «Framework of network virtualization for future networks» - структура виртуализации сетевых функций для сетей будущего;
- Y.3100 «Terms and definitions for IMT-2020 network» - Термины и определения для сетей 5G/IMT-2020;
- Y.3111 «IMT-2020 Network Management Framework» - Структура управления сетью IMT-2020;

- Y.3110 «IMT-2020 Network Management Requirements» - Требования к управлению сетью IMT-2020;
- Y.3100-series Supplement 44, «Standardization and open source activities related to network softwarization of IMT-2020»;
- Q.3745 «QoS management protocol for time constraint applications over SDN» - Протокол управления QoS для приложений поверх SDN, требовательных ко времени;
- Q.4061 «Framework of SDN controller testing» – Структура тестирования контроллера SDN;
- Q.3963 «The compatibility testing of SDN-based equipment using OpenFlow protocol» - Тестирование на совместимость оборудования SDN использующих протокол OpenFlow;
- и другие.

В процессе разработки и защиты рекомендаций Q.3745, Q.4061, Q.3963 автор настоящей диссертационной работы принимал непосредственное участие.

На момент 2021 года, основные вопросы, относительно инфраструктуры уже стандартизированы, и работа плавно переходит на вопросы стандартизации и выработки рекомендаций в интеграционных вопросах относительно реализации новых услуг в сетях. В том числе, одним из активно исследуемых вопросов является реализация Искусственного интеллекта в сетях связи.

Концептуальная архитектура Программно-конфигурируемых сетей отражена на рисунке 1.2.2.1.

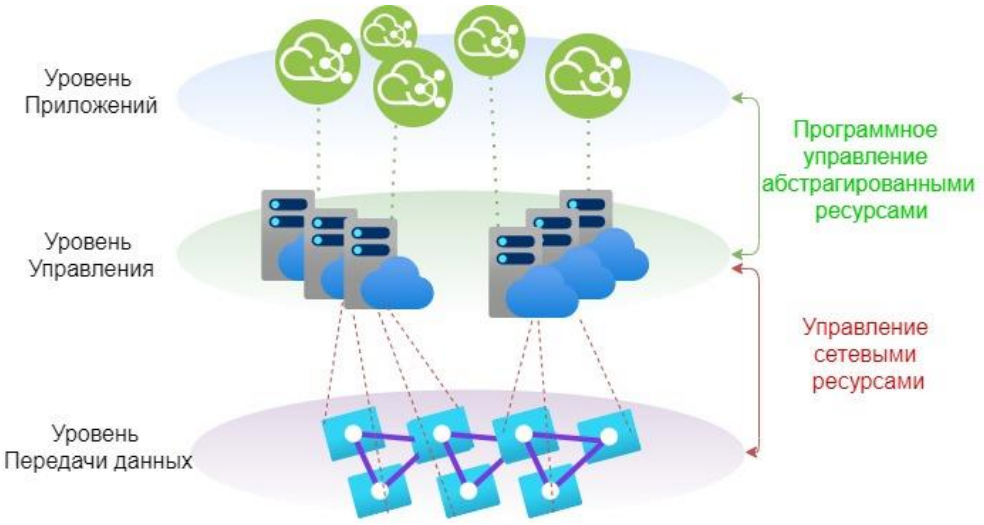

Рисунок 1.2.2.1 – Концептуальная архитектура Программно-конфигурируемых сетей

В Программно-конфигурируемых сетях выделяют три уровня [1]:

- *Уровень приложений*. Это уровень, где реализуется логика услуг, служебных сервисов на сети, обеспечивая автоматизацию бизнес-процессов предоставления услуг оператором, путем управления сетевыми ресурсами программным способом;
- *Уровень управления.* Данный уровень предоставляет инструменты для динамического управления поведением сетевых ресурсов (динамическая политика QoS и т.п.), согласно логике алгоритмов, реализованных на уровне приложений. При этом данный уровень, согласно рекомендации МСЭ-Т Y.3300 функционально подразделяется на подуровни [6]:

➢ *Поддержка уровня приложений*. Функция поддержки уровня приложений заключается в обеспечении стабильной работы программного интерфейса для SDN-приложений с целью получения доступа приложений к абстрактным моделям сетевых ресурсов.

➢ *Оркестрация.* Главным образом, функции оркестрации направлены на управление сетевыми ресурсами, которые могут быть представлены как физическими, так и виртуальными топологиями сети, сетевыми элементами, потоками трафика.

Также одной из главных функций данного подуровня является обеспечение согласования запросов с уровня приложений к абстрактным сущностям сетевых ресурсов путем организации политики доступа (ограничения, управления очередями), тем самым обеспечивая стабильность функционирования сетевой инфраструктуры.

- *Абстракция.* Функции абстракции заключаются в организации взаимодействия сетевых ресурсов и их представления в виде абстрактных моделей сетевых ресурсов. По сути данный подуровень реализует стандартный интерфейс взаимодействия между контроллером Программируемой сети и сетевыми ресурсами, с последующим их представлением в виде абстрактных сущностей(моделей), то есть программных объектов со стандартными программными интерфейсами.

- *Уровень передачи данных.* Данный уровень представляется набором сетевых устройств, реализующих главным образом обработку пакетов и их транспортировку по сети в соответствии с решениями (правилами коммутации), принятыми на уровне управления SDN. При этом также стоит отметить, изменение принципа обработки пакета в коммутаторе SDN, теперь коммутация происходит по принципу работы TCAM. При этом, если нет правила коммутации (так называемого «action»), пакет отправляется на контроллер Программируемой сети, инкапсулированный в пакет протокола OpenFlow OF_PKTIN (Packet IN). Функции данного уровня также подразделяются на несколько подмодулей:

  - *Поддержка управления.* Данный подмодуль уровня ресурсов реализует функции управления устройством путем организации взаимодействия с уровнем управления SDN, который представляется контроллером Программируемой сети. По сути, данный подмодуль представлен в виде подсистемы управления физического устройства (ресурса), на практике это «прошивка устройства», которая реализует логику протоколов управления и взаимодействия с контроллером SDN.

  - *Транспортировка и обработка данных.* Данный подмодуль реализует транспортировку данных (потоков) согласно правилам коммутации, установленным подсистемой управления. При этом правила устанавливаются с учетом требований и

запросов от приложений SDN. Например, приложение SDN настроило MPLS туннель и пакеты, удовлетворяющие соответствующим проверочным данным в таблице (Math Field), далее коммутируются в установленный туннель, соответственно к данному туннелю может быть применена логика управления параметрами QoS и так далее.

Стоит также отметить, что кроме полного контроля и управления, согласно принципам построения и архитектуры SDN, реализуется также и полный мониторинг сетевых ресурсов, а именно: мониторинг потоков в сети, статистика по пакетам (успешно скоммутированных, отброшенных, с ошибкой), нагрузка в сети, статистика физических портов и так далее.

Контроллер Программно-конфигурируемой сети имеет два стандартных интерфейса [1]:

- Южный интерфейс (интерфейс для взаимодействия с сетевыми ресурсами, коммутаторами SDN);
- Северный интерфейс (предоставление интерфейса программного взаимодействия для приложений SDN);

Таким образом, контроллер SDN управляет и конфигурирует сеть логически централизованным способом и предоставляет доступ к управлению сетевыми ресурсами приложениям SDN через их абстрактные представления (сущности). Благодаря архитектуре SDN, приложение может автоматически проводить работу с ресурсами сети через посредника (контроллера), работать в автоматизированном режиме с «сущностями» сетевых ресурсов. Стоит также отметить, что контроллер Программируемой сети может предоставлять различные типы интерфейсов для приложений SDN (например, объектно-ориентированные или более абстрактные, к примеру — REST API, либо внутреннее API контроллера, например — Java API).

Протокол OpenFlow.

Изначально коммутатор SDN архитектурно проектировался как коммутатор, способный только принимать, разбирать пакет (анализировать поля), производить поиск по Flow Table и в соответствии с правилом обрабатывать пакет (отбросить, отправить на контроллер ПКС, скоммутировать пакет на другой порт), при этом память такого коммутатора должна строиться на TCAM памяти и соответственно коммутатор должен иметь поддержку протоколов SDN. TCAM память или Ассоциативное запоминающее устройство (АЗУ) является особым видом машинной памяти, используемой в реализации очень быстрого поиска. В отличие от обычной машинной памяти (памяти произвольного доступа или RAM), в которой пользователь задает адрес памяти ОЗУ, возвращает «слово данных», хранящееся по этому адресу, Ассоциативная память (АП) разработана таким образом, чтобы пользователь задавал «слово данных», и осуществляла его поиск, чтобы выяснить, хранится ли оно где-нибудь в памяти. Если «слово данных» было найдено АП возвращает либо адрес хранения, либо «слово данных» или другие, связанные с ним части данных.

Таким образом, АП — аппаратная реализация того, что в терминах программирования назвали бы ассоциативным массивом. Данная архитектура позволяет снизить цену коммутатора, в такой компоновке одна из самых дорогих частей будет память TCAM. Однако многие компании-производители телекоммуникационного оборудования пошли по пути мягкой интеграции, реализуя модуль протокола OpenFlow, как добавление функциональности к своим и так уже многофункциональным коммутаторам, что в данной ситуации повышает его цену на рынке телекоммуникационного оборудования. Консорциум ONF, являясь разработчиком открытых стандартов SDN, выделяет следующие протоколы взаимодействия различных уровней архитектуры [1]:

1. OpenFlow — открытый расширяемый протокол установления соединения между контроллером и коммутатором, непосредственного управления потоками и формирования правил их обработки коммутатором;

2. OF-CONFIG — открытый расширяемый протокол конфигурирования сетевой среды и управления ею (operational context), включая виртуальные и физические устройства;

3. NB-API (NorthBound API) — API «северного» интерфейса, предоставляемый контроллером сетевым приложениям, например, REST API.

Модуль реализации протокола OpenFlow находится как на контроллере ПКС, так и соответственно на коммутаторе. При этом контроллер, в свое время предоставляет доступ к работе со своим модулем OpenFlow сторонним разработанным приложениям через свой северный-интерфейс. OpenFlow является открытым стандартом, создающим широкие возможности разработчикам для экспериментов с протоколами в локальной сети. OpenFlow стал внедрятся программно- и аппаратно- в коммерческие Ethernet коммутаторы, маршрутизаторы и беспроводные узлы доступа в качестве новой функции. Стандарт OpenFlow в настоящее время принят большинством производителей сетевого оборудования. На рынке телекоммуникаций в настоящее время доступны коммутаторы с поддержкой OpenFlow таких производителей как Cisco, Juniper, Brocade, Huawei и др. Протокол OpenFlow является первой, получившей признание и ставшей активно развиваться, реализацией протокола SDN и может использоваться в проводных и беспроводных сетях. Протокол OpenFlow реализует три вида сообщений [1]:

1. Сообщения контроллер-коммутатор – инициализируются контроллером, служат для управления коммутатором и контролем за событиями, происходящими на нем.

2. Асинхронные сообщения – инициализируются OpenFlow коммутаторами, предназначены для извещения контроллера о событиях на сети, например; сбои, ошибки, изменения состояния.

3. Симметричные сообщения – рассылаются как коммутаторами, так и контроллером.

На данный момент развития протокола OpenFlow он имеет уже несколько версий. Первая версия OpenFlow 1.0, разработанная в 2009 году имеет поддержку работы на MAC, IP уровне, на коммутаторах реализуется простая Таблица потоков (с англ. - Flow Table), но имеется поддержка работы с полем ToS протокола IP. Однако, в протоколе OpenFlow версии 1.1 имеется поддержка настройки MPLS туннелей, Multiple flow table (Несколько таблиц потоков), group table (групповые таблицы), OpenFlow 1.3 отличается своей многофункциональной поддержкой по сравнению со своими предшественниками. OpenFlow 1.4 уже реализует возможность мониторинга и конфигурирования оптических портов коммутатора (частота, мощность излучения). В версию OpenFlow 1.5 добавили еще больше функций (например, выходные Таблицы потоков и другое). На практике же немногие компании-производители ушли в реализации дальше версии протокола 1.0, что очень сужает возможности проектирования и построения сетей более сложной конфигурации и «урезает» гибкость администрирования сети. Протокол OpenFlow версии 1.3 реализован в своих коммутаторах крупными компаниями, такими, как: Cisco, HP, Huawei. Также реализацию OF 1.3 предлагает Российская компания Brain4Net. Протокол OpenFlow является не единственным в своем роде протоколом SDN. Протокол OF-CONFIG разработан для решения задач более высокого уровня, по сравнению с протоколом OpenFlow. В основном это задачи по построению сетевой среды в целом, конфигурации коммутаторов, назначения одного или более контроллеров OpenFlow коммутатору, конфигурирование портов и очередей, удаленное изменение свойств портов и тому подобное.

1.2.3. Виртуализация сетевых функций в сетях связи 5G/IMT-2020

Виртуализация сетевых функций (с англ., - NFV, Network Function Virtualization) – технология виртуализации физических сетевых элементов (физических ресурсов)

телекоммуникационной сети путем исполнения сетевых функций программными модулями, работающими на стандартных серверах и виртуальных машинах (VM) в них. При этом, данные программные модули могут взаимодействовать между собой для предоставления услуг связи, чем ранее занимались аппаратные решения.

С развитием телекоммуникационных и информационных технологий, произошла их конвергенция, все больше и больше информационные технологии стали проникать в мир Телекома, тем самым на данный момент развития технологий мы уже говорим об инфокоммуникационных технологиях. Подход виртуализации сетевых функций в мир телекоммуникаций пришел из мира информационных технологий, перевернув представление о том, как могут быть построены сети. При этом, в этот же момент времени назрела проблема масштабирования сетей, тенденция увеличения трафика и потребность абонентов в новых типах услуг (информационных) заставила телеком-операторов увеличивать парк дорогостоящего оборудования. Чтобы интегрировать в систему новое оборудование, часто требуется много капитальных вложений (CAPEX), времени, а также требуется увеличивать штат высококвалифицированных инженеров для обслуживания возросшей в масштабах сети. Кроме стоимости оборудования, оператору необходимо вкладываться в необходимые решения по обеспечению работы оборудования (серверные стойки, климатические установки, возможно новое помещение и так далее). Стоит также отметить, что при масштабировании сетевой инфраструктуры у оператора сети увеличиваются также операционные расходы (энергопотребление, кондиционирование, зарплатный фонд сотрудникам и так далее). Таким образом, эволюция сетей связи в сторону их виртуализации позволит снизить темп роста показателей CAPEX и OPEX, при этом также в некоторых случаях меняется модели бизнеса оператора, например, телеком-оператор может стать также и сервис-оператором для небольших предприятий, предоставив им соответствующие сетевые функции «по-запросу», как услугу (виртуальные VoIP станции, сетевые экраны и так далее), что несомненно принесет прибыль оператору. Различные типы сетевого

оборудования могут быть расположены на стандартных промышленных, больших вычислительных мощностей, серверах, коммутаторах и систем хранения, которые могут быть расположены в Центре 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 необходимости, без необходимости установки нового оборудования. Текущие сети связи состоят из большого количества разнообразных устройств (сетевых функций), которые обычно реализованы в виде АПК-решения. Структура NFV и ее описание отображается в спецификации ETSI GS NFV 002 от 2013 года.

В спецификации ETSI GS NFV 002 выделяют три главные подсистемы:

- Виртуальная сетевая функция(-ии), в качестве программной реализации сетевой функции, которая может работать поверх NFVI.
- NFV инфраструктура (NFVI), включающая разнообразные физические устройства, а также необходимое программное обеспечение, их объединяющее и реализующее функции виртуализации (гипервизор) для функционирования NF’s.
- Управление и оркестрация NFV. На данную подсистему возложены функции оркестрации и управления жизненным циклом физических и/или программных ресурсов, которые в свою очередь реализуют NFVI, а также жизненным циклом VNFs.

Абстрактная структура NFV, по спецификации ETSI отображена на рисунке 1.2.3.1.

Рисунок 1.2.3.1 – Абстрактная структура виртуализации сетевых функций

Как уже отмечалось выше, NFV позволяет динамически управлять сетевыми функциями (VNF) и отношениями между ними, то есть, по сути, менять их конфигурацию и связи. Данный уровень автоматизации достигается путем унифицированности интерфейсов взаимодействия систем, что позволяет как раз использовать единую систему управления и мониторинга.

1.2.4. Программирование как подход к управлению сетями

Выше уже были обозначены проблемы существующих технологий, механизмов, методов построения и управления сетями связи. Также были обозначены технологические вектора сетей связи 5G/IMT-2020, в процессе развития которых определились требования к услугам и технологиям построения и управления сетей связи и были выработаны рекомендации для обеспечения гибкости сетей и возможности их быстрого масштабирования. Так, в качестве основных технологий построения пакетных сетей были рекомендованы SDN и NFV. Благодаря программной абстрагированности и прозрачности всех функций и процессов в данных

технологиях, существует технологическая возможность программируемости всех ресурсов сети.

Программируемость ресурсов в современное время является необходимостью для эффективного мониторинга и управления, в связи с требуемой эффективностью и скоростью принятия решений в условиях Больших данных. Как результат развития вектора программируемости сетевых и вычислительных ресурсов через стандартные и открытые программные интерфейсы, появилась возможность разработки и реализации функций машинного обучения и алгоритмов Больших данных. Так, разработка и реализация так называемого «Искусственного Интеллекта» в сетях связи является уже достаточно практической задачей как на уровне международной науки, так и уже на уровне стандартизации и готовых рыночных продуктов. Ожидается, что внедрение ИИ в сети связи позволит достичь необходимого уровня управления сетью, ее ресурсами для обеспечения требуемых жестких критериев, относительно качества обслуживания для новых и будущих типов услуг. Говоря про разработку служебных сервисов, необходимо определить следующие три критерия:

- Виртуализация. Программное обеспечение должно выполняться в изолированной программной среде для обеспечения необходимого абстрагирования от версий и особенностей операционного ПО. Виртуализация позволяет также использовать уже имеющиеся системы оркестрации для управления самим ПО служебных сервисов.
- Модульность. Архитектура ПО служебных сервисов должна реализовывать принцип модульности с целью повышения устойчивости решений и возможности обновления части решений, при необходимости.
- Масштабируемость. Необходимо предусмотреть такую архитектуру программного обеспечения служебных сервисов, которая позволит производить быстрое масштабирование его необходимых функций, в том числе и географически.

1.3. Сети связи пятого поколения: на пути к сетям 2030

Как уже ранее было сказано, скорость появления новых технологий в услугах, сетях связи стала причиной плавного перехода к следующей концепции – сетей связи 2030 в то время, как работы по сетям пятого поколения завершенными еще считать ошибочно как на уровне стандартизации, исследований, так и на производственном уровне. Таким образом, с целью регулирования концепции 2030 в МСЭ была создана специальная рабочая группы по исследованию и последующей стандартизации поколения сетей связи 2030, которая должна определить основные характеристики и направления стандартизации сетей и услуг. Стоит отметить, чтобы заглянуть в 2030 год, необходимо, проанализировать основные тенденции в развитии сетей связи, которые появились сегодня при работе над концепцией 5G/IMT-2020. Например, уже неоднократно упоминался тип услуги «Тактильный Интернет» (ТИ), по ряду технических причин, нуждающийся в глубоком исследовании и доработке до того, как ТИ станет в один ряд с другими услугами. Напомним, что для ТИ были выдвинуты требования по задержке, которая не должна превышать 1мс и на данный момент развития технологий, в том числе технологий физического уровня, является трудноразрешимой, а в некоторых сценариях использования – неразрешимой задачей. Переходя к концепции сетей связи 2030, которая на данный момент уже принята в МСЭ и описана в следующем документе – “FG NET-2030 Technical Specification on Network 2030 Architecture Framework” от июня 2020 года. Можно определить следующие фундаментальные изменения в развитии сетей связи:

- *Сети связи с ультрамалыми задержками.* Тактильный Интернет привел к еще более значительным изменениям в области построения сетей связи, так как в этом случае потребовалось передавать данные с задержкой в 1мс, что на данный момент в 100 раз меньше, чем в существующих сетях связи. При этом стоит отметить,

что концепция ТИ приводит к такому процессу, как децентрализация сети связи, поскольку фундаментальные ограничения по скорости передачи света – непреодолимы [4].

- *Сверхплотные сети.* Сверхплотность сетей является одним из признаков не только сетей 5G/IMT-2020, но и всех последующих сетей. Как уже было отмечено, прогнозируется около 1 млн. устройств на 1 кв/м и в соответствии с прогнозами, предельное число Интернет Вещей составляет 50 триллионов, что может быть достигнуто как раз ближе к 2030 году. Таким образом, понятие сверхплотности будет только подтверждаться в процессе развития сетей и услуг к 2030 году [4].
- *Интернет навыков.* Одной из важных концепций услуг следует отметить концепцию Интернета навыков, которая появилась в 2017 году и также требует наличие характеристик сетей с ультрамалыми задержками. Интернет навыков, как концепция, позволяет реализовать в сетях с ультрамалыми задержками новые виды услуг. Данные услуги смогут позволить использовать сеть для приобретения людьми и робототехническими устройствами новых навыков [7, 8].
- *Летающие сети.* Стоит отметить, еще одно фундаментальное изменение в развитии сетей связи – объединение летающего и наземного сегментов сетей в единую сеть связи. Стоит отметить, что при небольшой высоте полета БПЛА в таких сетях, измеряемая в сетях связи с ультрамалыми задержками.
- *Беспилотный транспорт.* Одной из активно развивающихся концепций в услугах сетей связи является концепция беспилотного транспорта. Данное направление развивается не только специалистами ИКТ, но и специалистами автомобильной отрасли. На момент 2021 года, в данном направлении, можно сказать, что производство «обгоняет» уровень стандартизации, не говоря уже про сетевую и внешнюю вычислительную составляющую (вне борта автономного транспорта). Здесь можно отметить быстро развивающуюся компанию «Tesla», которая, несмотря на скептические отзывы большинства за последние 10 лет добилась успеха и на

данный момент имеет самую продвинутую систему «автопилот». Однако, опыт применения беспилотных образцов показал, что существуют трудноразрешимые и отчасти невозможные для расчета сценарии. Так, научное сообщество пришло к выводу, что необходимо объединить беспилотные средства передвижения в единую сеть с придорожными вычислительными мощностями, поддерживающими цифровую модель дороги или части дороги, с которой уже работает беспилотный транспорт. Таким образом, «беспилотник» может знать заранее о том, что происходит через 10 км, что находится по факту вне видимости бортовых систем.

На основе анализа фундаментальных изменений в развитии сетей связи, можно утверждать, что сети связи 2030 будут является сверхплотными сетями с ультрамалыми задержками, стремящимся к децентрализованной структуре сети, а также данное поколение сетей связи приобретет ряд новых характеристик за счет развития технологий в области сетей и систем связи (например, квантовые коммуникации) и в том числе в смежных отраслях.

Стоит также отметить следующие тенденции сетей связи 2030, которые упомянуты в документе МСЭ:

- *Услуги Телеприсутствия.* Персонализация сети. Или иначе говоря – присутствие цифровых аватаров в сети (двойников). Это одно из перспективных приложений для услуг сетей 2030. При этом предусматривается возможность нескольких аватаров одного человека. Для реализации данного типа сети необходимо построение сетей с ультрамалыми задержками. При этом услуги телеприсутствия не ограничиваются только воспроизведением действий пользователя. При использовании голографических приложений и аватаров будет возможно, например, смотреть футбольный матч не по телевизору, а как голографическую модель под любым углом[4].
- *Летающие сети.* На данный момент развитие летающих сетей требует активных исследований для их применения в гражданских целях. Поэтому, согласно

трендам летающие сети могут войти в широкую гражданскую эксплуатацию ближе к 2030 году [4].

- *Наносети.* В 2030 году различные применения наносетей[9] и нановещей скорее всего уже смогут быть в широко распространенными. Как минимум – в медицинской отрасли. Также одним из направлений в наносетях является молекулярные сети.

Согласно рекомендации МСЭ “FG NET-2030 Technical Specification on Network 2030 Architecture Framework”, сеть связи 2030 будет поддерживать различные и очень строгие функциональные и нефункциональные требования, включая строгие требования к низкой задержке и большому объему обмена данными. Дополнительные новые характеристики и возможности сети 2030 [10]:

- Детерминизм в задержках и передача без потерь;
- Встроенная поддержка нескольких типов услуг, своевременная активация и доступность услуг;
- Гибкость настройки сетевых сервисов и сетевых функций;
- Эффективный программируемый сетевой протокол;
- Безопасные и доверительные сети;
- Более высокий уровень устойчивости перед отказами;
- Интеграция большого количества интеллектуальных методов (методы на основе ИИ, Машинного обучения и Больших данных) в сетевую инфраструктуру, с целью контроля и управления;
- Эволюция архитектуры в сторону распределенного управления.

Архитектурные принципы сети связи и услуг — это проектные решения, используемые в качестве основы для работы системы. Каждый принцип применим к определенному набору точек зрения на архитектуру. С точки зрения пользователя, принципы системы понимаются как существенные характеристики системы, отражающие назначение системы и ее эффективное функционирование [10]. В

документе МСЭ предлагаются следующие архитектурные принципы, которые отражены на рисунке 1.2.4.1.

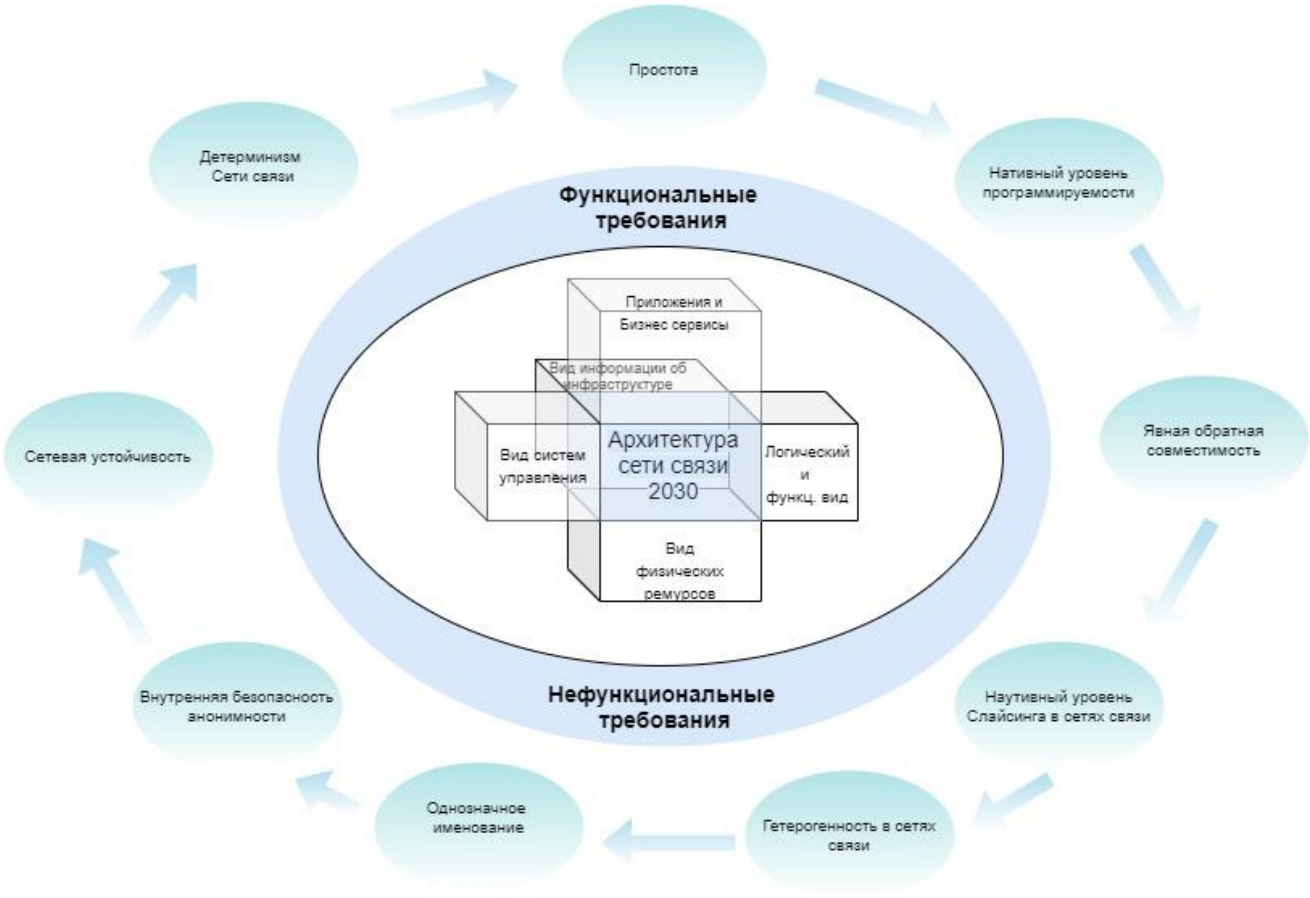

Рисунок 1.2.4.1 – принципы архитектуры сети связи 2030 [10]

Большинство характеристик сетей связи 2030 будут определяться новыми технологиями, которые найдут широкое внедрение для реализации этих сетей. ИИ (либо первые его версии/прототипы) будут интеллектуально управлять потоками трафика, квантовые компьютеры позволят терминалам пользователя выполнять множество новых трудоемких задач, нанотехнологии смогут открыть мир связи в нано измерении и предоставить различные новые типы услуг связи. Кроме того, концепция Индустрии 4.0 выйдет на новый уровень и другие изменения, которые на данный момент еще сложно оценить [4].

1.4. Искусственный интеллект в сетях связи

1.4.1. Введение

Причиной большого интереса применения технологий ИИ в сетях связи является надежда на решение множества трудноразрешимых задач перед сетями связи 5G и последующих поколений. В глобальном плане сети связи должны реализовывать требования сверхнадежных сетей с ультрамалыми задержками (с англ. URLLC – Ultra Reliable Low Latency Communications) [11], не говоря уже о все возрастающем количестве трафика межмашинных коммуникаций в сети, а также иметь возможность контроля и управления множеством современных и вновь появляющихся услуг с высокими требованиями по качеству обслуживания и восприятия, например, для услуг тактильного интернета или услуг Телеприсутствия [12].

Однако, кроме новых решений на всех уровнях сетей, а также решений в области облачных технологий, позволяющих отчасти приблизиться к тем требованиям, которые предъявляют услуги для сети, возникают не менее важные задачи по модернизации логики обработки трафика. В решении задач мониторинга и управления трафиком в сетях связи предполагается использование технологий ИИ. Потребность во внедрении столь ресурсоемких технологий обработки данных появилась в результате требования к динамичности сетевой инфраструктуры и ее способности адаптироваться к изменениям условий. В настоящее время генерируется множество служебных данных, которые необходимо эффективно обрабатывать и это становится под силу только технологиям ИИ [2]. В решении подобных задач проявляется одно из положительных фундаментальных преимуществ сетей связи, построенных на технологиях SDN/NFV – программируемость. Программируемость сети означает качественное изменение модели управления сетью: сетевой

администратор – сеть. Благодаря «прозрачности», которая предоставляется с помощью ряда технологий и протоколов SDN, сетью может управлять программное обеспечение, которое может реализовывать как раз те самые алгоритмы обработки больших данных и машинного обучения. Благодаря заложенным функциональным возможностям протокола OpenFlow, OFConfig, программируемым северным интерфейсам контроллера (например, REST, SOAP и др.), программное обеспечение верхнего уровня, представленное в виде служебных сервисов, может в режиме онлайн получать всю необходимую информацию о передаваемых потоках данных на уровне передачи данных (ПД).

1.4.2. Задачи Искусственного Интеллекта в сетях связи

Большинство задач относительно внедрения ИИ в сети можно подразделить на два типа: распознавание и прогнозирование, например, распознавание типа трафика, либо распознавание атаки на контроллер сети и так далее. В качестве задач прогнозирования, кроме прогнозирования изменения того или иного контролируемого потока ПД, возможно прогнозирование нагрузки на инфраструктуру оператора в целом с учетом множества критериев.

Таким образом, технологии построения сетевой инфраструктуры SDN/NFV позволяют разрешить множество задач, которые были поставлены перед сетями нового поколения. При этом их решение лежит через интеллектуализацию сети путем разработки служебных сервисов с применением технологий ИИ.

На рис. 1.4.2.1 приведена концептуальная схема сети с ИИ, построенная на технологиях SDN/NFV с имплементацией структуры граничных вычислений (с англ. MEC – Multi-access Edge Computing) и принципа слайсинга в сетях [3].

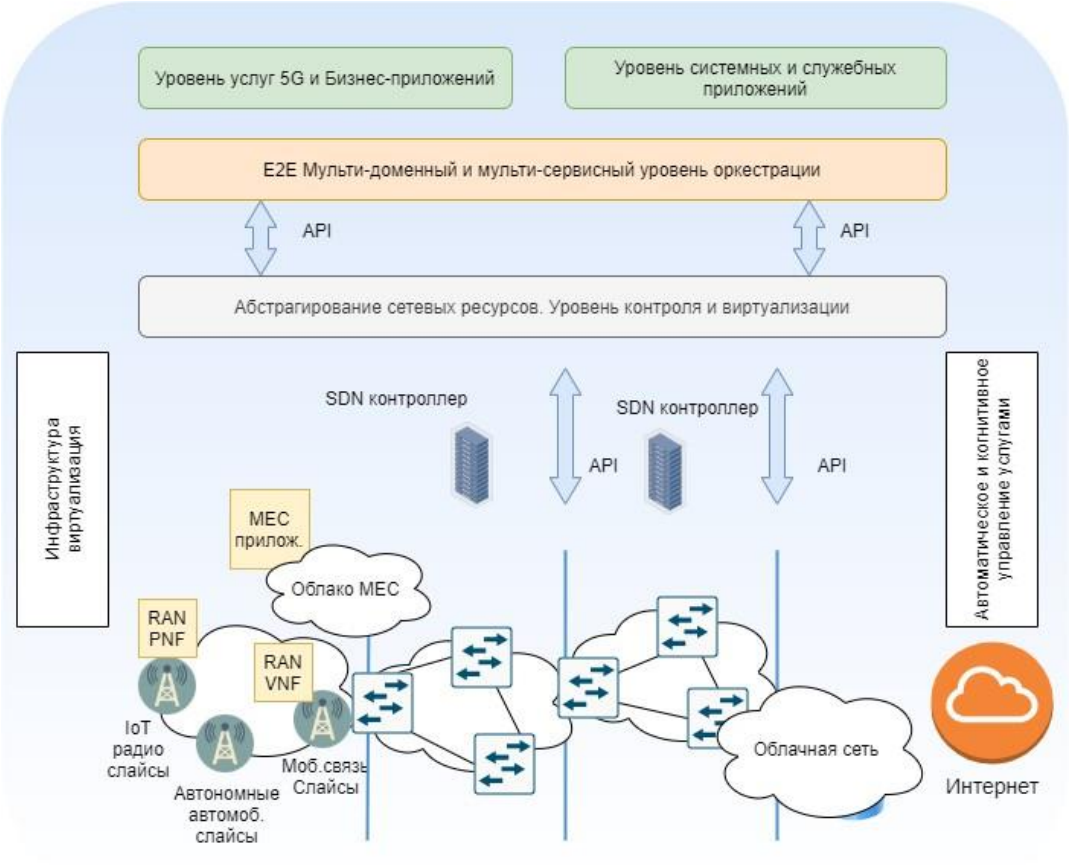

Рисунок 1.4.2.1 – Концептуальная схема сети с поддержкой ИИ

В последнее время машинное обучение применяется во множестве приложений, например, в виртуальных персональных помощниках, видеонаблюдении в социальных сетях, фильтрации спама и вредоносных программ по электронной почте, поисковых системах и т.д. Сегодня в качестве исследований в области использования ИИ [3] для сетей связи можно выделить ряд глобальных задач:

- Однозначная идентификация трафика в сети связи, не вносящая дополнительных задержек в поток и обеспечивающая требования сетей связи с ультрамалыми задержками;

- Системный онлайн мониторинг сети связи от потока данных (в том числе виртуального) до многопараметрических моделей сегмента сети с множественным доступом устройств и систем;
- Кратковременное и долговременное прогнозирование нагрузки как на элементы сети, так и на целые сегменты;
- Кратковременное и долговременное прогнозирование поведение потоков ПД на уровне ПД и служебных потоков на уровне управления;
- Долговременное прогнозирование нагрузки на сетевую и вычислительную инфраструктуру с учетом трендов изменения профилей трафика и типов сервисов с целью определения и автоматического формирования предложения по сокращению или расширению сети, а также ее пороговых характеристик.
- Эффективное распределение радиоресурсов покрытия 5G с прогнозированием нагрузки на соты.
- Повышение качества сигнала с помощью прогнозных кодеков физического уровня.
- Кратковременное и долговременное прогнозирование потребностей пользователей в тех или иных услугах.
- Прогнозирование передвижения пользователя географически, а также формирование модели его предпочтений в контенте.
- Распознавание и прогнозирование атак злоумышленников на систему с формированием опережающей реакции на возможную атаку.
- Применение технологий ИИ для согласованного распределения сервисов по сети на структурах пограничных вычислений (MEC) и туманных вычислений (Fog).
- и другие.

Данный перечень определенных задач в области ИИ в сетях связи, естественно не представляет собой все задачи в области исследований ИИ. На базе проведенного анализа существующих наработок на уровне как исследований, так и стандартизации,

предлагается вести исследования использования ИИ в сетях, в первую очередь, как распределенной, независимой структуры с учетом децентрализации сети. Децентрализованный и распределенный ИИ в виде систем и подсистем, взаимодействующих между собой, представляет собой новую устойчивую структуру, которая также имеет возможность автоматического восстановления, а при необходимости – и расширения. Суть децентрализации отражена на рисунке 1.4.2.2.

В сети должен существовать оркестратор функций (модулей) машинного обучения, выполняющий функции мониторинга, контроля и передачи соответствующих функций машинного обучения системам управления сетью. Учитывая сложность и многокомпонентность сети, данный оркестратор не является централизованным и определен на соответствующей сети, подсети, услуге (например, Умный город или Тактильный интернет). Такие оркестраторы должны иметь возможность взаимодействия между собой с целью получения большего синергетического эффекта от внедрения технологий ИИ в сети связи. При этом, как это показано на рисунке 1.4.2.2, при оркестраторе существуют соответствующие хранилища функций и модулей машинного обучения, которые при необходимости мигрируют в качестве служебных микросервисов в соответствующую систему управления сети. Это сопровождается непрерывным обменом данными между оркестратором и всеми ему подконтрольными модулями функций машинного обучения, участвующих в конвейерах Машинного обучения [3].

В данной концепции применения ИИ в сетях связи особенно важен принцип микросервисного подхода, который позволит выделять соответствующие функции машинного обучения в независимые модули, взаимодействующие как между собой, так и с вышестоящим (управляющим) модулем. Появляется гибкость в управлении системой, ее обновлении (при необходимости), а также легком и независимом масштабировании. Например, если нужно увеличить производительность или обеспечить новые системы интеллектуальным модулем управления, достаточно

размножить соответствующие микросервисы и произвести их миграцию в соответствующий сегмент сети [3].

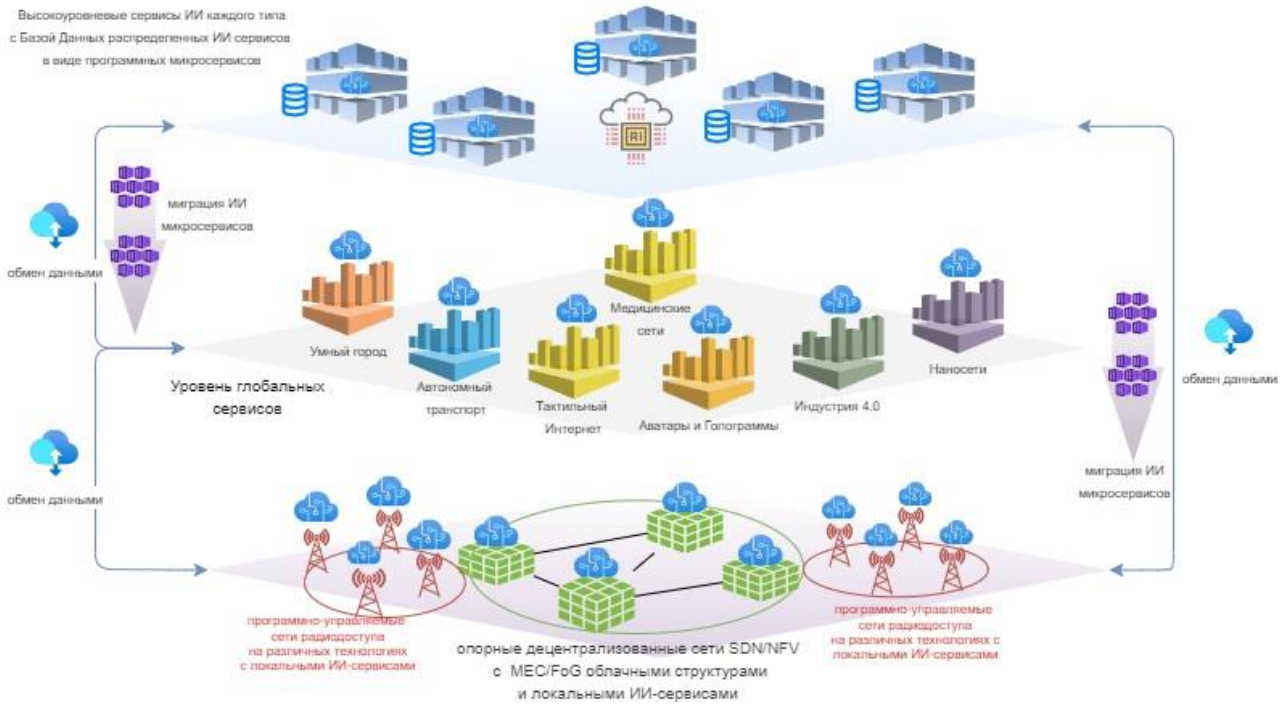

Рисунок 1.4.2.2 – Концептуальная архитектура распределенного ИИ в сетях связи и перспективные направления исследований

Резюмируя, можно сказать, что исследования по использованию технологий ИИ в сетях связи особенно актуальны сегодня, при этом актуальность этих исследований будет сохраняться как минимум на протяжении 10 лет до появления сетей связи 2030[3].

1.4.3. Аналитический обзор по стандартизации Искусственного Интеллекта в сетях связи

В настоящее время в МСЭ-Т ведется активная работа по анализу и разработке проектов рекомендации в области ИИ в сетях связи. Как уже выше было упомянуто, основной Исследовательской комиссией в данной области является 13 ИК, при которой в декабре 2020 года была открыта специальная Фокус-группа по проведению всестороннего анализа и совместной разработки проектов рекомендаций. В данную ИК входят как международные известные компании, так и участники-академии, в числе которых также и СПбГУТ им. проф. М.А. Бонч-Бруевича. К данному моменту, МСЭ-Т уже выпустило ряд рекомендаций в области машинного обучения, но основополагающей считается Рек.Y.3172. Функционально одним из основных элементов в Рек. Y.3172 является конвейер машинного обучения, т.е. набор логических узлов, каждый из которых имеет определенные функции. Их можно комбинировать для формирования приложения машинного обучения в телекоммуникационной сети. В рекомендации также представлена высокоуровневая архитектура конвейера с определением всех основных функциональных блоков и их взаимодействием между собой [3]. Архитектура представлена на рисунке 1.4.3.1.

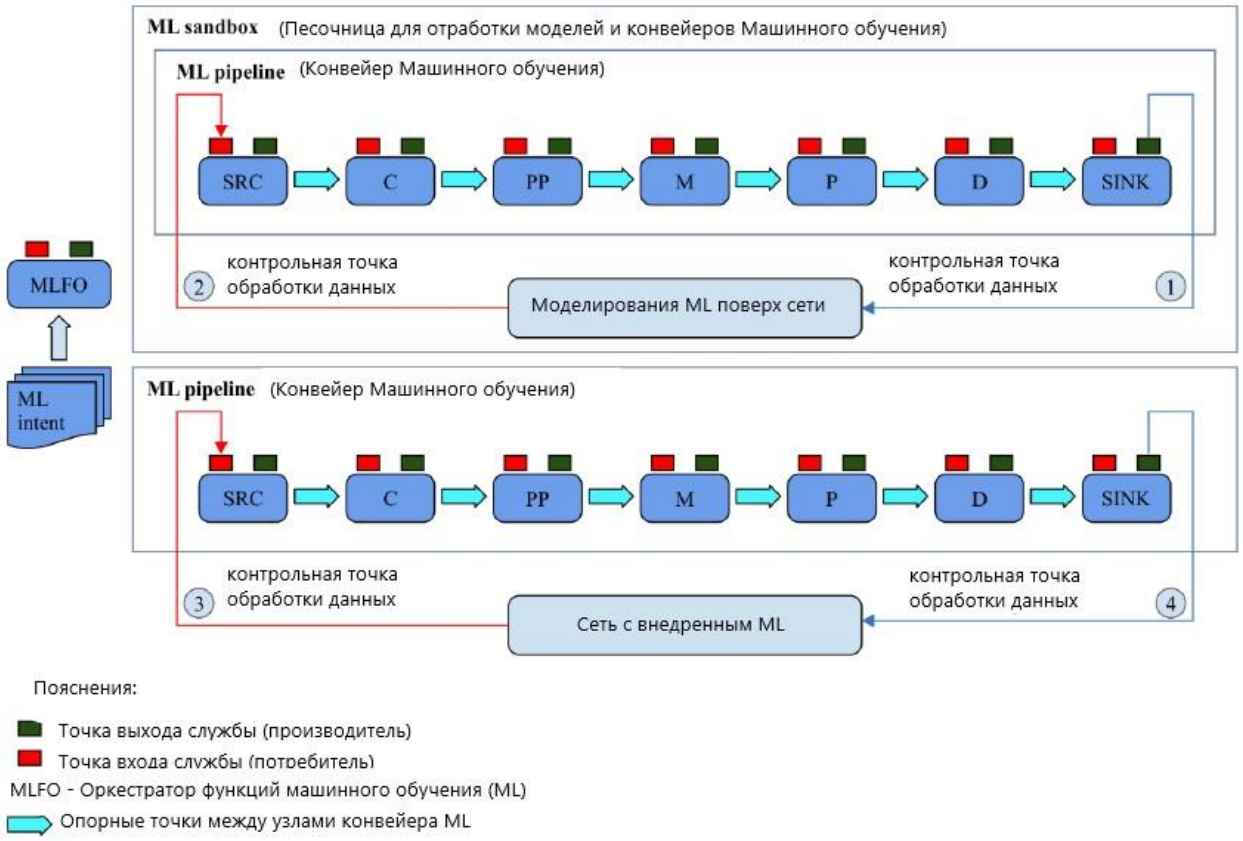

Рисунок 1.4.3.1 – Конвейер Машинного обучения для сетей связи

Элементы высокоуровневой архитектуры конвейера машинного обучения, согласно рек. МСЭ-Т Y.3172 «Архитектура для Машинного обучения в будущих сетях, включая IMT-2020»:

- SRC (Source) – источник запроса на обслуживание. Потенциальный источник включает в себя пользовательское оборудование (с англ. UE-User Equipment), Функции управления сессиями (с англ. SMF – Session Management Function, согласно рек. МСЭ-Т Y.3104 [15]) и функции приложений (с англ. AF – Application Function, согласно рек. МСЭ-Т Y.3104 [15]);
- C (Collector in ML pipeline) – коллектор сбора запросов для системы машинного обучения на уровне 1 для сети доступа и на уровне 2 для ядра сети. Коллектор сбора запросов имеет возможность конфигурировать SRC. Например, протокол функций контроля радиоресурсов (с англ. RRC – Radio Resource Control,

согласно стандарту 3GPP TS 23.501 [18]), может быть использован для конфигурирования пользовательского устройства (UE), представленного в качестве SRC устройства Коллектор может использовать специальные протоколы операций, администрирования и обслуживания (с англ. OAM – Operations, administration and maintenance) для настройки функций управления сессиями (SMF), выступающего в качестве узла SRC. Такие конфигурации могут использоваться для управления характеристиками данных, их детализацией и периодичностью, в процессе их генерации из SRC;

- PP (Preprocessor) – Элемент предварительной подготовки данных. Данный узел ответственен за «очистку» данных, агрегирование данных или выполнение любой другой предварительной обработки, необходимой для того, чтобы данные были в подходящем формате для их дальнейшего использования в моделях машинного обучения;
- M (Model) - Это модель машинного обучения в форме, которая может использоваться в конвейере машинного обучения;
- P (Policy) – Узел управления политикой. Этот узел позволяет применять различные политики к выходным данным узла модели. Конкретные правила могут быть введены сетевым оператором для обеспечения работоспособности сети, например, серьезные обновления могут выполняться только в ночное время или при низком трафике данных в сети;
- SINK - это узел-цель вывода ML. Например, UE, действующее как узел SINK, может регулировать периодичность измерения канала на основе вывода ML.

Примечание: «Точка выхода службы» узла - это точка, где можно получить доступ к службам, предоставляемым узлом (обеспечивает производитель). «Точка входа службы» узла - это точка, откуда могут потребляться службы других узлов (обеспечивает производитель);

- D (Distributer) – узел распределения. Этот узел отвечает за идентификацию так называемых SINK и распределение выходных данных M-узла на соответствующие узлы SINK;
- MLFO (Machine Learning Function Orchestrator) – оркестратор функций машинного обучения. Это логический узел с функциями, которые управляют и координируют узлы конвейеров ML на основе данных ML и/или динамических сетевых условий;
- ML sandbox - это изолированный домен, который позволяет размещать отдельные конвейеры машинного обучения для их обучения, тестирования и оценки перед развертыванием в действующей сети. Для обучения или тестирования песочница машинного обучения может использовать данные, сгенерированные из имитированных базовых сетей машинного обучения и/или живых сетей.
- ML Intent - это декларативное описание, которое используется для определения приложения ML. ML Intent не определяет какие-либо сетевые функции, а обеспечивает основу для сопоставления приложений ML с различными сетевыми функциями, зависящими от технологий. ML Intent может использовать метаязык, специфичный для машинного обучения, для определения приложений ML.

Реализация конвейера, учитывая архитектуру сети связи 5G/IMT-2020[13] приведены на рисунке 1.4.3.2. Здесь приняты следующие обозначения из рек. МСЭ-Т Y.3102 [19]:

- SMF (Session Management Function) – функция управления сессиями;
- NACF (Network Access Control Function) – Функция контроля доступа в сеть. Функциональные возможности NACF включают управление регистрацией, управление соединением и выбор функции управления сеансом (SMF).
- AF (Application Function) – функции приложения. Эта функция предоставляет информацию о сеансе в PCF, чтобы SMF мог использовать эту информацию для управления сеансом. AF взаимодействует со службами приложений,

которым требуется динамическое управление политиками. AF извлекает информацию, относящуюся к сеансу (например, требования QoS) из сигнализации приложения, и предоставляет ее PCF для поддержки генерации правил.;

- PCF (Policy control function) – функция контроля политик. Эта функция предоставляет функциональные возможности для контроля и управления правилами политики, включая правила обеспечения QoS, билинга и маршрутизации трафика.
- UPF (User Plane Function) - эта функция обеспечивает функции для маршрутизации трафика, управления туннелем сеанса PDU (IP или не-IP протокольная единица – с англ. Protocol Data Unit) и обеспечения QoS.
- CEF (Capability exposure function) – эта функция предоставляет функциональные возможности для сетевых функций и сетевых сегментов, чтобы предоставлять их возможности в качестве услуги третьим сторонам.
- NFR (Network function registry function) - эта функция предоставляет функции, помогающие обнаруживать и выбирать требуемые сетевые функции. Каждый экземпляр сетевой функции регистрируется при создании и обновляет свой статус (т.е. активация/деактивация), так что NFR может поддерживать информацию о доступных экземплярах сетевых функций.
- USM (Unified Subscription Management function) - эта функция хранит и управляет унифицированным UE и информацией о подписке, включая, помимо прочего, информацию о регистрации UE и управлении мобильностью, информацию о сетевых функциях, которые обслуживают UE, информацию об управлении сеансом для установления сеанса PDU. USM также предоставляет информацию аутентификации UE в ASF.
- ASF (Authetification Server Function) - эта функция выполняет аутентификацию между UE и сетью. NACF инициирует аутентификацию UE, вызывая ASF. На основе информации аутентификации UE, полученной от USM, ASF выбирает

метод аутентификации и выполняет процедуры аутентификации UE (согласно стандарту 3GPP TS 33.501) [18].

- NSSF (Network Slice Selection Function) - эта функция предоставляет функциональные возможности для выбора подходящих экземпляров сетевого сегмента (т.н. «Слайса») для UE. Когда UE запрашивает регистрацию в сети, NACF отправляет запрос выбора сетевого сегмента в NSSF с предпочтительной информацией о выборе сетевого сегмента. NSSF отвечает сообщением, включающим в себя список подходящих экземпляров сетевого сегмента для UE.
- UE-AN data transport – уровень передачи данных между пользовательским устройством и сетью доступа
- RP-au (Reference Point) – опорная точка взаимодействия между пользовательским оборудованием и UPF;
- RP-ud (Reference Point) – опорная точка между UPF и пакетной сетью передачи данных.

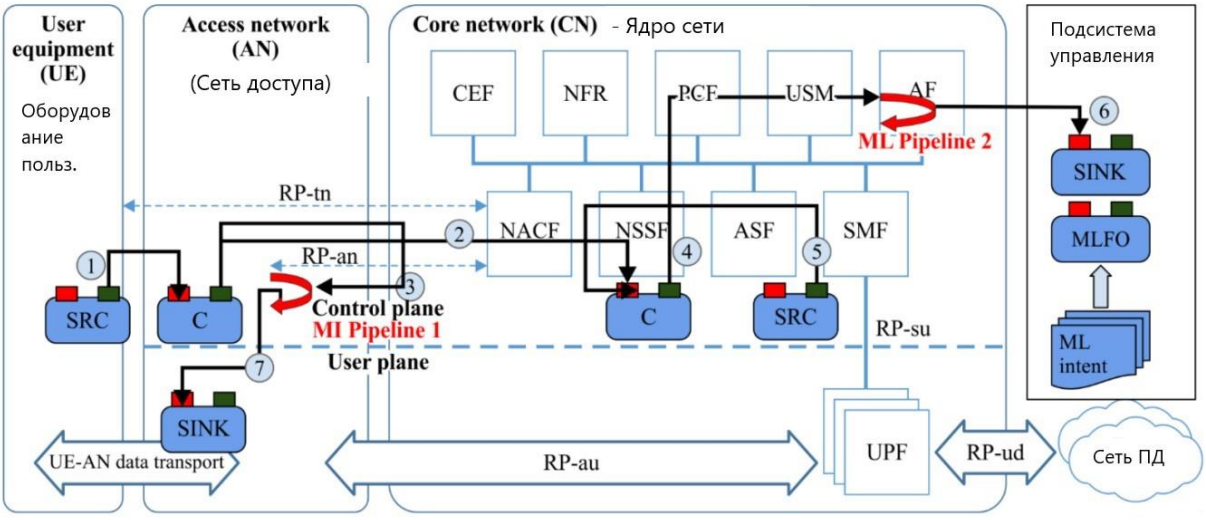

Рисунок 1.4.3.2 – Высокоуровневая архитектура конвейера машинного обучения (МСЭ-Т рек. Y.3172) и его реализация в сети 5G/IMT-2020

Из рисунка 1.4.3.2 видно, что предложенный конвейер машинного обучения охватывает все уровни сети 5G/IMT-2020 [19], внедряется в основные модули управления сетью (рисунок 1.4.3.1) и тем самым обеспечивает прозрачный мониторинг и контроль функционирования сети. Стоит также отметить, что в рамках данной рекомендации определены функции такого важного элемента сети, как оркестратор функций машинного обучения (MLFO). Этот модуль реализует функции управления и оркестрации других модулей, относящихся к конвейеру машинного обучения, и учитывает в принятии решений динамику функционирования сети и ее характеристики.

К настоящему времени в МСЭ-Т уже утверждены следующие рекомендации в области машинного обучения:

- Y.3170 «Требования к обеспечению качества обслуживания на основе машинного обучения для сети IMT-2020» [14];
- Y.3174 «Структура обработки данных для обеспечения машинного обучения в будущих сетях связи, включая IMT-2020» [15];
- Y.3175 «Функциональная архитектура обеспечения качества обслуживания на основе машинного обучения для сети IMT-2020» [16];
- Y.3176 «Естественная интеграция машинного обучения в будущие сети, включая сети IMT-2020» [17].

Кроме принятых рекомендаций, в МСЭ-Т, в том числе в 13 ИК на данный момент существует немалое количество документов, находящихся в разработке и обсуждении. В том числе проект рекомендации по методу распознавания трафика на основе аналитики метаданных на основе Машинного обучения - Y.IMT-2020_mAI: “Traffic typization IMT-2020 management based on an artificial intelligent approach”, который был подан совместно от СПбГУТ и ПАО «Ростелеком» и планируется к подаче на принятие его в качестве рекомендации МСЭ-Т на осенней сессии в 2021 году.

В свою очередь, подкомитетом объединенного технического комитета (JTC 1/SC 42) международной организации по стандартизации и международной электротехнической комиссии (ИСО/МЭК) ведется работа по разработке стандартов по Искусственному Интеллекту. Следующие документы, с которыми на данный момент возможно ознакомиться (данные документы также находятся в процессе разработки):

- ISO/IEC JTC 1/SC 42/AHG 1 Dissemination and outreach;
- ISO/IEC JTC 1/SC 42/AHG 2 Liaison with SC 38;
- ISO/IEC JTC 1/SC 42/AHG 3 Intelligent systems engineering;
- ISO/IEC JTC 1/SC 42/JWG 1 Joint Working Group ISO/IEC JTC1/SC 42 - ISO/IEC JTC1/SC 40: Governance implications of AI;
- ISO/IEC JTC 1/SC 42/SG 1 Computational approaches and characteristics of artificial intelligence systems;
- ISO/IEC JTC 1/SC 42/WG 1 Foundational standards;
- ISO/IEC JTC 1/SC 42/WG 2 Big Data;
- ISO/IEC JTC 1/SC 42/WG 3 Trustworthiness;
- ISO/IEC JTC 1/SC 42/WG 4 Use cases and applications;
- ISO/IEC JTC 1/SC 42/WG 5 Computational approaches and computational characteristics of AI systems.

Стоит отметить, что столь активная работа по исследованию и разработке стандартов в данной области в первую очередь направлена на достижение цели разработки Интеллектуальной сети, которая сможет обеспечить устойчивость, высокий уровень качества обслуживания. Также системы управления, использующие в своих решениях алгоритмы и подходы Искусственного интеллекта должны существенно понизить такие показатели, как CAPEX и OPEX.

1.5. Анализ методов мониторинга и управления трафиком в сетях связи 5G/IMT-2020 на основе технологий Машинного обучения и Больших данных

Искусственный интеллект – это быстро развивающаяся область, включающая в себя широкий спектр областей знания, в который в том числе входит представление знаний, упрощение, планирование, принятие решений, Машинное обучение, оптимизация, метаэвристические алгоритмы. Всем известный Тест Тьюринга дает исчерпывающее определение интеллекта, где компьютер должен ответить на некоторые вопросы, сформированные человеком. Компьютер проходит тест, если исследователь не может определить, был ли ответ написан человеком или машиной (компьютером). Для того, чтобы пройти тест Тьюринга, компьютер должен иметь расширенные возможности, такие как обработка языка, представление знаний, автоматическое рассуждение, машинное обучение и компьютерное зрение [20].

С момента выхода первых работ в данной области знаний (1950-е), подходы ИИ росли, благодаря ключевым вкладам, которые привели к появлению новых областей знаний, такие как экспертные системы, нечеткая логика и эволюционные вычисления. Дальнейшие усилия стимулировали исследования в области ИИ, совершенствуя существующие методы и предлагая новые гибридные интеллектуальные подходы. Метаэвристические алгоритмы, машинное обучение и системы нечеткого вывода могут быть широко использованы в концепции Программно-конфигурируемых сетей и виртуализации сетевых функций. На сегодняшний день технологии искусственного интеллекта (ИИ) позиционируют, как основные факторы, стимулирующие экономический рост государств в современных условиях. Под цифровой трансформацией понимаются глубокие и всесторонние изменения в производственных и социальных процессах, связанные с тотальной заменой аналоговых технических систем цифровыми и с широкомасштабным применением цифровых технологий. Цифровая трансформация охватывает не только саму

производственную деятельность, но и организационные структуры компаний и бизнес-моделей». Использование цифровой информации и знаний, современных информационных сетей является ключевым фактором производства для различных видов экономической деятельности. Эффективное применение ИКТ выступает серьезной движущей силой повышения результативности и оптимизации структуры экономики.

К 2025 году, как следует из Отчета GIV 2025 [21] в мире будет насчитываться 40 млрд интеллектуальных устройств на базе искусственного интеллекта и 100 млрд сетевых соединений, способствующих переходу на цифровые технологии в приоритетных сферах – социальной, производственной и банковской. Эти два фактора серьезно сократят потребность в использовании хранилищ информации, будет сделана ставка на быстрый, безопасный и интеллектуальный обмен данными. В результате ежегодный объем глобальных данных достигнет 180 млрд Тбайт, т.е. в 20 раз больше, чем сегодня (9 млрд Тбайт). Средний объем использования данных в сетях мобильной связи на пользователя в день вырастет примерно в 30 раз – до 1 Гбайт.

Машинное обучение - это обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных «обучаться». Термин «Машинное обучение» ввел американский компьютерный ученый Артур Самуэль в 1959 году как «способность компьютера учиться, не будучи явным образом запрограммированным» [22].

Стоит выделить основные области, в которых применяются методы и алгоритмы Машинного обучения:

- *Ассоциативная память.* В традиционной модели работы с памятью, обращение к данным доступно только с помощью адреса хранения, не зависящего от содержания самой памяти. Ассоциативная память или память, адресуемая по смыслу, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или при поврежденном содержании. Ассоциативная память может быть использована в мультимедийных

информационных базах данных. Кроме того, ассоциативная память используется в принципе работы коммутатора Программно-конфигурируемых сетей.

- *Распознавание образов.* Суть задачи в данной области состоит в определении принадлежности входного образа, представленного вектором признаков к одному или нескольким предварительно определенным. К известным задачам относятся распознавание букв, классификация сигнала клеток крови, распознавание языка.

- *Управление.* Суть задач в данной области заключается в том, чтобы определить входное воздействие на систему, при котором система будет действовать по желательной траектории, заданной эталонной моделью.

- *Аппроксимация функций.* В данном направлении, суть задач заключается в нахождении неизвестной функции, которая генерирует известные обучающие выборки (пары данных вход/выход). Аппроксимация функций необходима при решении многочисленных инженерных и научных задач в моделировании систем.

- *Прогнозирование*. Здесь задача заключается в предвидении следующего значения дискретного отсчета, относительно имеющейся выборки в последовательные моменты времени. Прогнозирование имеет большое значение для принятия решений в бизнесе, науке и технике.

- *Оптимизация*. В данной области знаний, основной целью является нахождение такого решения, которое сможет удовлетворить системе ограничений и минимизирует или максимизирует целевую (фитнесс) функцию.

- *Кластеризация и категоризация.* В данной области, алгоритмы направлены на определение подобий образов и помещение подобных образов в один кластер.

Обученная нейронная сеть обрабатывает исходные данные и выдает прогноз будущего поведения изучаемой системы. Основным недостатком программ на основе Искусственных Нейронных Сетей (ИНС) является как раз проблема правильного

обучения нейросети и исключение избыточного обучения, что очень может повлиять на адекватность модели системы. При этом, нахождение данного баланса лежит через эмпирический уровень определения параметров ИНС.

Существует множество методов идентификации и классификации, которые используют различный математический аппарат и различные подходы при реализации. Однако эффективность этих методов зависит от конкретной решаемой задачи. Несмотря на то, что всё последнее десятилетие мировое научное сообщество занимается проблемой повышения качества результата процесса Машинного обучения, на сегодняшний день не существует методов, которые могли бы однозначно эффективно решить задачи классификации и идентификации.

Выделяют следующие четыре группы методов Машинного обучения:

1. *Обучение с учителем (с англ. Supervised Learning).* Данный метод заключается в следующем: имеется множество объектов (ситуаций) и множество возможных ответов (откликов, реакций). Существует некоторая зависимость между ответами и объектами, но она неизвестна. Известна только конечная совокупность прецедентов – пар «объект, ответ», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость, то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Для измерения точности ответов определённым образом вводится функционал качества. Стоит отметить, что под учителем понимается либо сама обучающая выборка, либо тот, кто указал на заданных объектах правильные (маркированные) ответы.

2. *Обучение без учителя (с англ. Unsupervised Learning).* Данный метод охватывает широкий класс задач обработки данных, в которых известны только описания множества объектов обучающей выборки, и требуется обнаружить внутренние зависимости, взаимосвязи, закономерности, которые могут существовать между объектами. Обычно обучение без учителя часто противопоставляется обучению с учителем, когда, например, для каждого обучающего объекта задается

«правильное решение», и требуется найти зависимость, соответственно, между объектами и ответами.

3. *Обучение с подкреплением (с англ. Reinforcement leaning).* Данный метод охватывает задачи, когда агент должен действовать в окружении, чтобы максимизировать некоторый долговременный выигрыш. Стоит отметить, что алгоритмы с подкреплением, или иначе с частичным обучением, пытаются найти стратегию, приписывающую состоянием окружающей среды действия, которые должен предпринять агент в этих состояниях.

4. *Обучение с частичным привлечением учителя (с англ. Semi-Supervised Learning).* Данный метод, используется при обучении как размеченные (маркированные), так и неразмеченные данные. Обычно используется небольшое количество размеченных и значительный объём неразмеченных данных. Частичное обучение является компромиссом между обучением без учителя (без каких-либо размеченных обучающих данных) и обучением с учителем (с полностью размеченным набором обучения). Стоит отметить, что неразмеченные данные, будучи использованными совместно с небольшим количеством размеченных данных, могут обеспечить значительный прирост качества обучения. Сбор размеченных данных для задачи обучения требует, чтобы эксперт вручную классифицировал объекты обучения. При этом затраты, связанные с процессом разметки, могут сделать построение полностью размеченного набора прецедентов невозможным, в то время как сбор неразмеченных данных сравнительно недорог. В подобных ситуациях ценность частичного обучения сложно переоценить. Примером частичного обучения может послужить со-обучение: два или более обучаемых алгоритма используют один и тот же набор данных, но каждый при обучении использует различные — в идеале некоррелированные — наборы признаков объектов. Альтернативный подход заключается в моделировании совместного распределения признаков и меток.

В качестве примеров методов для выше озвученной классификации приведем следующие:

1. Методы обучения с учителем в задачах мониторинга и управления трафиком.

- Метод Искусственных Нейронных Сетей (ИНС). Данный метод объединяет различный тип ИНС. Рассмотрим следующие. Метод рекуррентных нейронных сетей для обучения системы (с англ. RNN – Recurrent Neural Networks) – является расширением нейронных сетей с прямой связью, которое решает проблемы с последовательными (например, текста или речь) или временными рядами [23, 24]. В отличие от традиционной нейронной сети с прямой связью, выход RNN зависит от предыдущих вычислений, это основная причина, почему он называется рекуррентной нейронной сетью. Алгоритм обратного распространения через время (BPTT) в основном используется для этапа обучения RNN. Следующий метод – Свёрточная нейронная сеть (с англ. Convolutional Neural Network) - специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году [25] и нацеленная на эффективное распознавание образов [26], входит также в состав технологий глубокого обучения (с англ. Deep Learning). Сверточные нейронные сети (ConvNets) были предложены для обработки данных, которые поступают в виде множественных массивов [56], таких как изображения. ConvNets используются в обучении функций для крупномасштабной классификации изображений. Типичная сеть CovNet состоит из трех уровней: сверточного уровня, уровня подвыборки (уровня пула), полностью связного уровня [27].
- Метод опорных векторов (с англ. SVM – Support Vector Machine) – набор алгоритмов обучения с учителем, способных выполнять линейную и нелинейную классификацию, регрессию и также выявление выбросов в данных. Методы SVN особенно хорошо подходят для классификации сложных, но небольших или средних наборов данных. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора, поэтому метод также известен как метод классификатора с максимальным зазором. Метод

SVM успешно применяется в полях классификации, аппроксимации функций и прогнозирования временных рядов [28];

- Метод Дерева Решений (ДР) для обучения системы. Стоит отметить, что Деревья решений являются классом очень эффективной модели Машинного обучения, позволяющей получить высокую точность в решении многих задач, при этом сохраняя высокий уровень интерпретации. При этом, метод Деревьев достаточно эффективен в задачах классификации. Дерево Решений может описывать набор данных в виде древовидной структуры [29]. Достоинство Деревьев Решений состоит в том, что они легко представляемы. В каждом узле можно точно увидеть, какое решение принимает модель. На практике можно точно определить, что является причиной точности и ошибок, с какими видами данных модель будет справляться и как значения признаков влияют на выход. Многие модели машинного обучения требуют предварительной обработки данных (например, нормализации) и нуждаются в сложных схемах регуляризации. Метод Дерева Решений не требует объемной подготовки данных. Но при этом Деревья Решений более эффективны после настройки некоторых параметров. К недостаткам данного метода можно отнесли следующие: деревья подвержены «переобучению», деревья уязвимы к смещению классов, которые есть в большинстве наборов данных. Данный недостаток может быть нивелирован с помощью периодической балансировки классов.

Стоит также отметить, что входные и выходные данные могут быть дискретными или непрерывными. Деревья Решений могут представлять все булевы функции.

- Метод ансамбля для обучения системы. Метод ансамбля объединяет предсказания различных подходов (путем взвешенного или невзвешенного голосования) и в основном используется для повышения производительности алгоритмов обучения. Пакетирование, представляет собой первый эффективный метод, используемый для повышения точности путем создания улучшенного составного классификатора, который объединяет различные выходные данные

изученных классификаторов в единый выходной. Каждый из этих классификаторов обучается на примерах, генерируемых случайной выборкой с заменой из исходного набора данных [30]. При повышении, в отличие от пакетирования, на каждый классификатор влияет производительность предыдущего классификатора, и он старается уделять больше внимания ошибкам, допущенным предыдущим классификатором [31].

- Метод глубокого обучения системы с учителем. Глубинное обучение является универсальным многоуровневым подходом к обучению машины по представлению. При обучении по представлению машина может научиться автоматически обнаруживать представления, требуемые для задачи классификации или обнаружения, основываясь только на необработанных данных, тогда как традиционные методы машинного обучения не могут обрабатывать естественные данные в их необработанном виде. Многоуровневое представление позволяет преобразовать представление с низкого уровня в более высокий, абстрактный. Достаточное количество этих преобразований позволяет изучать более сложные функции [32]. Стоит отметить, что методы глубокого обучения достигли лучшей производительности по сравнению с традиционными алгоритмами, используемыми для многих задач машинного обучения, таких как распознавание речи, обнаружение атак, понимание естественного языка. [32, 33]. Модели глубокого обучения подразделяются на три группы: порождающие модели, дискриминационные модели, гибридные модели. Дискриминационные модели в основном используют контролируемые подходы к обучению, тогда как генеративные модели используют неконтролируемые подходы к обучению. Гибридные модели, с другой стороны, используют как дискриминационные, так и генеративные модели [34].

2. Методы обучения без учителя в задачах мониторинга и управления трафиком

- Метод кластеризации «К-средних» (с англ. – K-means). Метод K-средних является одним из известных подходов кластеризации [35]. Для работы данного

алгоритма требуется предварительное знание параметра k, который указывает число полученных кластеров. Каждая точка данных будет назначена ближайшему центроиду каждого кластера. Алгоритм минимизирует целевую функцию, которая представляет расстояние между точками данных и их соответствующими центроидами [36]. Процесс обновления центроидов, основанный на их назначенных точках данных, будет повторяться до тех пор, пока центроиды не останутся такими же или пока точки не изменятся. Стоит тут также отметить модифицированный метод кластеризация нечетких С-средних (FCM) [37], которая также известна как мягкое K-среднее, когда каждая точка данных может принадлежать более чем одному кластеру. Другими словами, точка данных может принадлежать всем кластерам с различной степенью принадлежности.

- Метод самоорганизующейся карты Кохонена (с англ. SOM – Self-Organizing Map) для обучения системы. это хорошо известный неконтролируемый подход к обучению в искусственных нейронных сетях, который отображает многомерное распределение в низкоразмерное представление, называемое SOM-картой [38, 39]. Метод самоорганизующейся карты оказался успешным в различных задачах распознавания образов, включая очень зашумленные сигналы [40]. Процесс обучения в данном методе строит и реорганизует карту с использованием входных данных. После этого он классифицирует новый входной вектор на основе нахождения узла на карте [38].

- Метод скрытой Марковской модели (с англ. HMM – Hidden Markov Model). Скрытая Марковская модель [41] представляет собой статистическую модель, где разрабатываемая система считается Марковским процессом с ненаблюдаемыми (скрытыми) состояниями. Марковский процесс - это случайный процесс, который подходит для Марковского предположения, где Марковское предположение состоит в том, что вероятность одного состояния зависит только от предыдущего состояния. Параметры HMM могут быть обучены контролируемым или неконтролируемым образом.

- Метод ограниченной машины Больцмана (с англ. RBM - Restricted Boltzmann Machine). RBM представляет стохастическую ИНС, которая состоит их двух уровней: входного слоя и скрытого слоя. RBM по сравнению с базовой машиной Больцмана заключается в том, что связность нейронов, где каждый нейрон во входном слое связан со всеми скрытыми нейронами и наоборот, но между любыми двумя нейронами в тот же слой. Кроме того, блок смещения связан со всеми видимыми и скрытыми нейронами [42].
- и другие.

3. Методы обучения с подкреплением в задачах управления и мониторинга трафика. Один из методов Машинного обучения, в ходе которого испытуемая система или иначе – агент обучается, взаимодействуя со средой. Агент воздействует на среду, а среда воздействует на агента. О такой системе говорят, что она имеет обратную связь. Впервые такого рода обучение с обратной связью было предложено и изучено в 1961 году в работе М.Л. Цетлина. Системой подкрепления называется любой набор правил, на основании которых можно изменять с течением времени матрицу взаимодействия (или состояние памяти) перцептрона. Существует различные реализации данного метода в Языках программирования: BURLAP в Java (лицензия LGPL), MMLF в Python (лицензия GPL) и другие.

4. Методы обучения с частичным привлечением учителя в задачах мониторинга и управления трафиком. Это способ использовать неразмеченные (с англ. unlabeled) данные для того, чтобы повысить точность модели. Такое обучение предполагает использование для каждого прецедента как пар «ситуация-решение», так и просто набор ситуаций. Обучение с частичным привлечением учителя похоже на обучение с учителем, однако использует как размеченные, так и неразмеченные данные. Размеченные данные – это, по сути, наборы единиц информации с приписанными им метками (тегами). В неразмеченных данных таких меток нет. Комбинируя методы обучения, алгоритмы могут обучаться размечать неразмеченные данные. Отсутствие меток, а также помеченная часть могут содержать случайный

шум, образующий ситуацию между контролируемым и неконтролируемым обучением. Этот метод наиболее реалистичен, как и во многих реальных приложениях, часто трудно собрать много помеченных данных из-за того, что эксперты помечают данные вручную, тогда как проще собрать много немаркированных данных. Поскольку он включает в себя некоторые небольшие помеченные данные, эффективность подходов к обучению с «полудиспансерным» обучением превосходит обучение без надзора [43].

Как уже ранее было упомянуто, математический аппарат обучающихся систем начал предлагаться еще в 50-х годах 20-го века, благодаря как отечественным ученым, так и зарубежным. Однако, техническая возможность их реализации, с точки зрения доступности данных технологий в массах, стала появляться только лет 10 назад. И к 2021 году использование инструментов Машинного обучения в задачах различных областей стало отчасти обыденностью. В тоже время, данный класс технологий, названных Искусственным Интеллектом, только начинает свой «путь» модернизации и усовершенствования известных моделей (бизнес, производственных, технологических). ИИ берет на себя «рутинные» функции, к примеру – роботы кол-центра, ведение самолета по заданному курсу, автопилоты и так далее. Однако сети связи для ИИ являются еще новым, «не паханным» полем, работой в котором еще следует научить алгоритмы ИИ. В области мониторинга и управления в ИКТ задачи являются более трудоемкими, где необходимо учитывать, как меняющиеся характеристики сетей, требований к услугам, множество протоколов и подходов управления, так и динамичность пользователей и самих программных систем. При этом, одновременное упрощение сетей и систем связи с возрастающими требованиями и количеством новых услуг, а также необходимой скоростью принятия решений и знания о том, что будет в будущий момент времени с системой, ведет к неизбежному внедрению управляющих систем на основе методов ИИ, что определяет актуальность настоящей диссертационной работы не только в данный момент времени, но и на ближайшее будущее.

1.6. Цель и задачи диссертационной работы

Исходя из изложенного целью диссертационной работы является исследование и разработка методов построения инфраструктуры и предоставления услуг сетей связи с использованием технологий Искусственного Интеллекта.

Для достижения поставленной цели в диссертации последовательно решаются следующие задачи:

- анализ концепций современных и перспективных сетей связи, учитывая долгосрочную до 2030 году перспективу;
- анализ тематики «Искусственный Интеллект в сетях связи», учитывая прогресс стандартизации автономных сетей в МСЭ-Т;
- анализ методов машинного обучения и больших данных для задач мониторинга и управления трафиком в сетях связи;
- разработка метода идентификации трафика услуг в сетях связи пятого и последующих поколений;
- разработка структуры и метода взаимодействия туманных и граничных вычислений с поддержкой микросервисной архитектуры услуг;
- разработка метода прогнозирования нагрузки на контроллеры программно-конфигурируемых сетей.

1.7. Выводы по Главе 1

1. В данной главе был проведен анализ концепций современных и перспективных сетей связи, в том числе учитывая долгосрочную до 2030 года перспективны. Было определено, что сети 5G/IMT-2020 и последующего поколения,

сети связи 2030, обладают такими новыми характеристиками как «ультрамалые задержки», «сверхнадежная связь», «сверхплотные сети», что требует разработки новых методов их построения.

2. Так как требуется обеспечить свойства сетей связи, указанные в п.1 данного вывода, для реализации требований по «сверх плотности», «ультрамалым задержкам» требуется рассмотреть новые технологии и методы построения сетей связи и систем предоставления услуг, такие как Программно-конфигурируемые сети, Виртуализация сетевых функций, граничные вычисления с множественным доступом (MEC), туманные вычисления, микросервисный принцип проектирования ПО услуг.

3. Кроме озвученных в п.2 технологий и методов для построения сетей связи пятого и последующих поколений, требуется изменение принципов управления сетью и услугами через применение технологий Искусственного Интеллекта. Особенно актуальными вопросами являются однозначная эффективная идентификация трафика на уровне передачи данных, прогнозирование данного трафика, прогнозирование нагрузки на контроллеры Программно-конфигурируемых сетей, эффективное распределение вычислений на уровне туманной инфраструктуры и пограничных вычислений с множественным доступом. В диссертационной работе предложено для идентификации трафика использовать метаданные потоков, формирующиеся на основе полей MatchField открытого протокола управления OpenFlow. Принцип метаданных в качестве исходных данных был предложен также для метода прогнозирования нагрузки на котроллеры Программно-конфигурируемых сетей. Для исследования и разработки метода идентификации трафика и прогнозирования нагрузки на контроллеры SDN была использована модельная сеть программно-конфигурируемых сетей, реализованная на кафедре сетей связи и передачи данных СПбГУТ.

ГЛАВА 2. МЕТОД ИДЕНТИФИКАЦИИ ТРАФИКА УСЛУГ В СЕТЯХ СВЯЗИ ПЯТОГО И ПОСЛЕДУЮЩИХ ПОКОЛЕНИЙ

2.1. Сети связи с ультрамалыми задержками как основа построения сетей связи пятого поколения

Ранее, в анализе современных и перспективных концепций сетей связи, были приведены три основных принципа, на основе которых строится сеть пятого поколения и которые, соответственно, будут соблюдаться и в последующем поколении:

- Широкие межмашинные взаимодействия (с англ. MMC – Massive Machine type Communications);

- Улучшенная мобильная широкополосная связь (с англ. eMBB – Enhanced Mobile Broadband);

- Сверхнадежная связь с ультрамалыми задержками (с англ. URLLC – Ultra-reliable and low latency communications);

На данный момент, URLLC представляют одну из самых сложных задач, стоящих перед научно-техническим сообществом. Данный класс сетей призван быть сверхнадежным и при этом иметь сверхнизкую задержку передачи данных. Таким образом URLLC выдвигает определенные и при этом жесткие новые требования качества обслуживания (QoS), обеспечение которых позволит “дать жизнь” новым типам сервисов. Данные сервисы реализуются в следующих направлениях: Электронная медицина (e-health), Автономный транспорт, опасные индустриальные производства формата 4.0 и другие. Одним из самых явных типов сервисов сетей URLLC является Тактильный Интернет. В данном случае следует отметить, что на архитектуру сети наибольшее влияние оказывают именно ультрамалые задержки:

происходит децентрализация сети вследствие ограничений по расстояниям, на которых возможно оказывать услуги Тактильного Интернета. Это, в свою очередь, должно привести к децентрализации экономики и стиранию цифрового неравенства между территориями страны [2]. Взаимодействие тактильных ощущений может происходить в режиме реального времени, только если сеть обеспечит задержку меньше, чем соответствующая физиологическая константа, выражающая инерционность сенсорной системы человека в 1мс. Задача выполнения требований по величине задержки в 1мс накладывает большие ограничения на системно-сетевые решения по построению сетей связи. Тактильный Интернет, как тип услуги, может быть реализован в Медицинских целях, например: удаленный осмотр больного, удаленная операция и другое.

Учитывая последнюю эпидемиологическую ситуацию в мире с COVID-19 можно сделать вывод, что услуги Тактильного Интернета могли бы сохранить жизни множества врачей, которые, не жалея собственной жизни, лечили пациентов. Например, в палате с зараженными COVID-19 мог бы находится медицинский робот, управляемый врачом удаленно и передающий все необходимые тактильные ощущения при осмотре и т.д. В данном случае, также Интернет навыков также смог бы найти свое применение, как надстройка над Тактильным Интернетом. Однако, уровень развития технологий существующих сетей не позволил реализовать данный тип Сервиса [44].

Таким образом, при большем внедрении технологий как сетевых, так и вычислительных, необходимо пересмотреть принципы управления сетями. Учитывая те возможности программируемости, которые были изначально заложены в Программно-конфигурируемые сети и системы оркестрации вычислительных структур, необходимо их реализовывать, учитывая технологии Искусственного Интеллекта. Современный объем трафика, его гетерогенность и разнообразие сервисов Интернета Вещей, в том числе тех, которые относятся к URLLC, диктует новые требования к оперативности принимаемых решений по обеспечению качества

обслуживания. Существующие инструменты не могут предоставить необходимый уровень [44]. И в данном случае уже в большинстве решений требуется иметь прогнозы нагрузок на те или иные сервисы, учитывая географические факторы и динамические (передвижение абонента), в том числе на больших скоростях. При этом у оператора есть потребность в полноценных системных прогнозах развития инфраструктуры, учитывая скорость внедрения новых технологий сервисов в сети Интернет, изменения образа жизни людей (децентрализация и увеличение количества перемещений в пространстве).

Отметим те KPI, достижение которых позволит утверждать о готовности сетей пятого поколения (согласно 3GPP TR 38.913), с точки зрения требований к задержкам. Сквозная задержка в сети (User Plane latency):

- 4мс для услуг типа eMBB;
- 1мс для услуг типа URLLC;
- сквозная задержка на уровне управления (Control Plane latency) - 10мс;

а также, надежность передачи данных (вероятность потери пакета данных): $1 \cdot 10^{-5}$ для пакета в 32 байт со сквозной задержкой в сети в 1мс для услуг URLLC. И плотность подключенных абонентских устройств: 1млн на 1 км^2.

Стоит также заметить, что в процессе разработки решений в сетях пятого поколения, разработки стандартов, проведения исследований, пришли к выводу, что требования URLLC сложно выполнимы, с точки зрения широкомасштабного внедрения данного типа услуг. Они могут быть предоставлены в ограниченном географическом пространстве, по выделенным сетевым каналам, и при других ограничениях. Однако ожидается, что в рамках эволюционирования концепции пятого поколения в концепцию сетей связи 2030, данный вид услуг будет реализован. Для реализации рассматриваются одновременно несколько направлений разрешения ограничения:

1. Изменение технологий физического уровня и переход на так называемые «квантовые коммуникации», которые реализуют иной физический принцип передачи информации, основанный на принципе квантовой запутанности. Однако данный метод находится на стадии ранних исследований.

2. Децентрализация сетей связи и децентрализация систем вычислений, гибкое их управление с имплементацией Искусственного Интеллекта в качестве систем мониторинга и управления.

На данный момент можно заметить развитие сети в озвученном вопросе по следующему вектору: изменение инфраструктурных решений на программируемые структуры с совместной разработкой и пошаговым внедрением технологий ИИ от тривиальных задач до более сложных. Важной эпохой развития сетей связи будет переход на квантовые компьютеры и квантовые коммуникации. Однако управление данными системами уже будет немыслимо без систем, реализующих в замкнутом цикле все функции ИИ в сетях связи. И к этому времени данная система будет иметь достаточное количество статистической информации, на которой она обучится и сможет сама плавно перейти на управление сетями связи на основе квантовых коммуникаций.

Возвращаясь к текущему уровню технологий и близлежащим задачам, в рамках задач внедрения ИИ на сети связи, имеются особенно актуальные задачи: однозначная идентификация трафика с последующим прогнозированием, эффективное и динамическое распределение вычислений на инфраструктуре распределенных вычислений, прогнозирование и идентификация возможных перегрузок управляющих систем. При этом задача идентификации трафика включает в себя необходимость распознавания большого количества типов (учитывая URLLC) сервисов, не внося дополнительных задержек в трафик, возможность расширения и подстройки алгоритма под определенное географическое расположение сети и сервисов [44].

2.2. Задача идентификации трафика услуг в сетях связи

Кроме новых решений на всех уровнях сетей, а также новых решений в области облачных технологий, позволяющие отчасти приблизиться к тем требованиям, которые предъявляют сервисы перед инфраструктурой, стоят не менее важные задачи по модернизации логики обработки трафика. Так, например, на данный момент времени качество обслуживания в сетях связи предоставляется с помощью технологий DiffServ, а также других TE (с англ. Traffic Engineering) решений, например, MPLS-TE, использующее в основе протокол резервирования ресурсов RSVP-TE. Однако данные решения имеют ряд недостатков для их применения в ряде сервисов в сетях пятого и последующих поколений. Основными недостатками являются отсутствие динамического управления в зависимости от изменчивости профиля трафика в сети, возможности быстрой переконфигурирования политик обслуживания подконтрольного домена сети, а также ограниченный набор классификаторов трафика, что в условиях, появляющихся и отличающихся по требованиям к качеству обслуживания сервисов Интернета Вещей и других, является очень критичным недостатком. В стандарте 3GPP TS 23.501, предоставляется карта требований QoS к различным видам услуг. В ней определены основные 18, от голоса и видео, до eMBB и Дополненной реальности (AR, с англ. Augmented Reality). Также приводится понятие Слайсинга, которое напрямую связывается с предоставлением QoS. Однако не все возможные услуги определены в данном 3GPP стандарте. При этом в каждом из направлений возможно определить приоритетность под-услуг. Так, например, в Тактильном Интернете возможно выделить медицинскую область основным приоритетом, а остальные области применения – более низким [42].

При этом стоит отметить, что пакеты в трафике, которым требуется предоставить определенные качества обслуживания [46], должны быть заранее

помечены меткой - поле ToS в заголовке IP. В настоящее время поле ToS называется поле «differentiated services» и имеет 6 бит поля DiffServ Code Point (DSCP) и 2 бита поля «Explicit Congestion Notification». Существующие механизмы удобны для использования их с целью обеспечения качества обслуживания в таких сферах, как: телефония, видео, телевидение и т.п. Однако их применение для обеспечения качества обслуживания множеству сервисов Интернета Вещей не удобно, а в некоторых случаях невозможно. С учетом вышеуказанных недостатков в данный момент, трафик Интернета Вещей смешан с общим потоком трафика передачи данных [47]. Трафик IoT может идти вместе с трафиком сервиса Video-on-Demand, серфинга страниц в Интернете и другим. Всем давно известная проблема “big headers with small bodies” влияет на работоспособность устройств уровня передачи данных так, что буферы устройств переполняются, что напрямую влияет на уровень качества сервиса [45]. Во многих работах приводится впечатляющая статистика экстенсивного и интенсивного роста трафика, генерируемого различными смарт-устройствами. Неоднородность трафика, сложность расчета его роста, а также оперативного расчета изменения его профиля, приводят к тому, что существующие «Ручные» методы расчета теряют свою актуальность. Для обеспечения оперативного реагирования систем управления на рост трафика (в том числе периодического), изменения его профиля, требуется рассмотреть иные методы расчета и управления. Для того, чтобы обеспечить оперативное реагирование, сети связи и соответственно устройства коммутации и маршрутизации трафика должны иметь необходимый уровень абстрагирования от «физики» процессов. Необходимый уровень абстрагирования от физических процессов [48] как раз реализуют концепции SDN и NFV.

Определим задачу исследования: Для оперативного реагирования систем управления, в случае концепций SDN и NFV - контроллера SDN и Оркестратора, требуется производить высокоточную идентификацию трафика без непосредственного вмешательства в поток на уровне передачи данных и

соответственно, внесения изменения в его профиль, а также задержек. Стоит отметить, что для заранее зарегистрированных потоков трафика не стоит задача его распознавания, например – трафика VoIP или IpTV. В данной работе рассматривается задача выявления трафика Интернета Вещей в сети SDN из общих (заранее не зарегистрированных) потоков. Для решения вышеописанной задачи был проведен комплексный анализ математических методов классификации, представленный выше, с учетом особенностей входных, анализируемых данных о потоках, а также особенности изначального требования – работы системы в режиме «на лету». Как результат проведенного анализа, был выбран подход использования нейронных сетей определенной архитектуры. В соответствии с поставленной целью и особенностями входных наборов данных была выбрана архитектура нейронной сети.

2.3. Архитектура модельной сети

Проведение исследований предлагаемого метода включает в себя разработку программного обеспечения (приложения контроллера SDN), его тестирование. Для данных работ необходимо разработать лабораторный стенд Программно-конфигурируемых сетей, с поддержкой северного программного интерфейса, а также развернутых на нем сервисов Интернета Вещей, который будет эмулировать работу одного из приложений Умного города, а также сервис потокового видео. Наличие двух типов трафика в сети необходимо для тестирования метода.

Основой лабораторного стенда является модельная сеть SDN лаборатории «Программно-конфигурируемых сетей и магистральных DWDM-сетей» кафедры «Сетей связи и передачи данных (ССиПД)» Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича. В качестве трафика генератора Интернета Вещей будет использоваться решение, разработанное в данной

работе [49]. Стоит отметить, что инфраструктура лаборатории для данной задачи была также модернизирована. Данная модельная сеть включает в себя также и другие исследовательские лаборатории кафедры ССиПД: IpTV, Интернета Вещей, Качества восприятия, моделирования и оптимизации. Данная модельная сеть позволяет производить исследования в области разработки и реализации методов и алгоритмов ИИ для сетей связи 5G/IMT-2020 для части Программно-конфигурируемой сети. Архитектура используемой модельной сети приведена на рисунке 2.3.1.

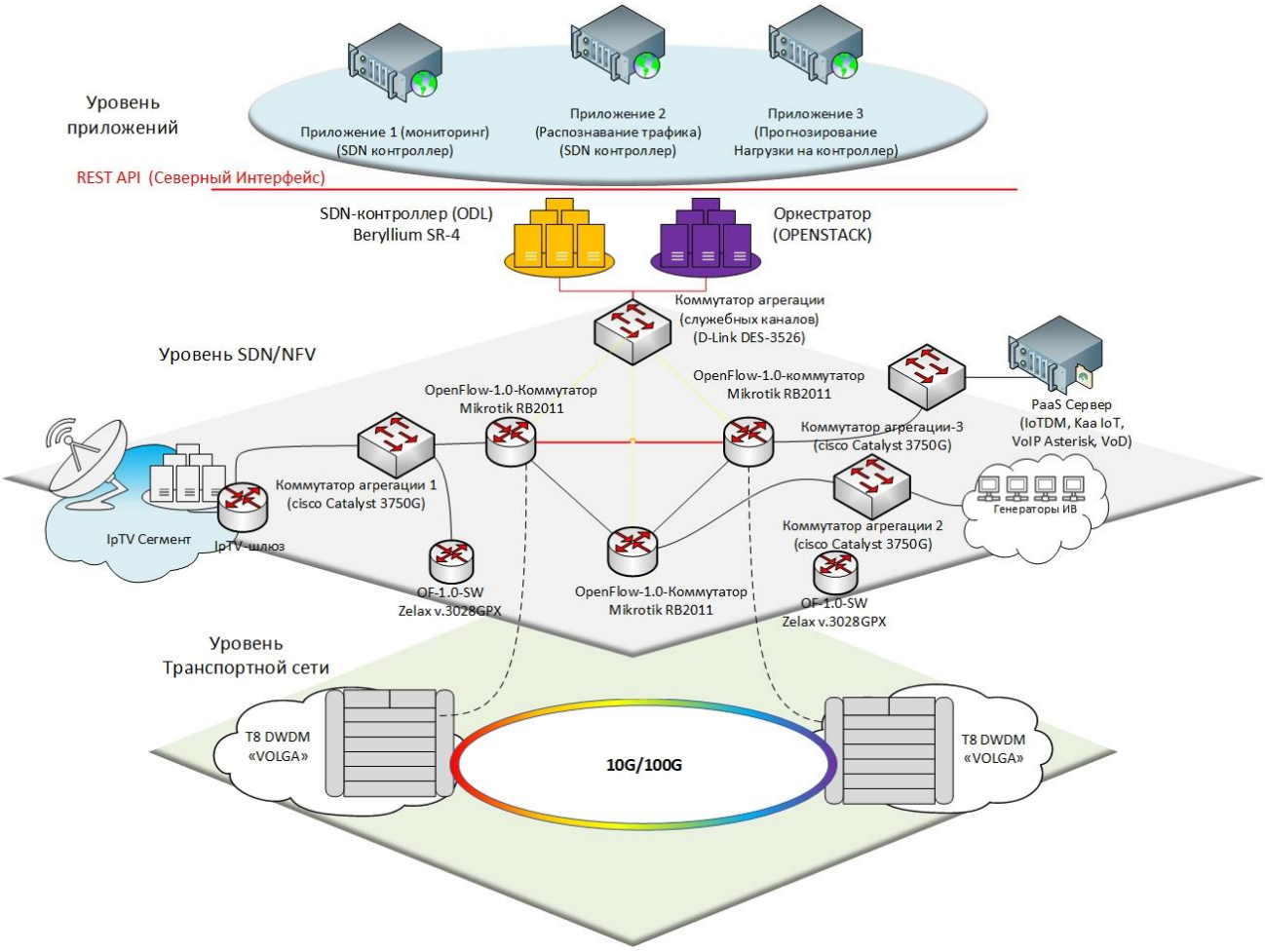

Рисунок 2.3.1 – Архитектура используемой модельной сети лаборатории

Лабораторный стенд представляет собой соединенные в кольцо три программно-управляемых коммутатора SDN, реализующие протокол OpenFlow версии 1.0.

Выбранный контроллер с открытым исходным кодом OpenDaylight программно-конфигурируемой сети реализует распределение нагрузки по сети изначально по умолчанию. Так как данный стенд был собран в первую очередь с целью проведения исследований, в уровень управления был интегрирован обычный коммутатор Ethernet, который агрегировал все служебные потоки и обеспечивал доступность контроллера SDN. Также в сети данного коммутатора был реализован доступ к северному уровню контроллера SDN. Таким образом, приложения сети могут быть разработаны и развернуты на других серверах, которые работают с контроллером или оркестратором сети через данный коммутатор. Также коммутатор на данном уровне позволяет масштабировать сеть путем увеличения количества подключенных коммутаторов к одному контроллеру в сети. Для расширения количества возможных подключений к каждому коммутатору SDN был подключен последовательно простой Ethernet-коммутатор, выполняющий роль агрегации в данной топологии.

Стоит обратить внимание на контроллер OpenDaylight. Данное решение является одним из лучших на рынке открытого ПО контроллеров Программно-конфигурируемых сетей с точки зрения следующих критериев: техническая поддержка, открытость, архитектурное решение контроллера, широкая функциональность, масштабируемость и скорость обработки потоков OpenFlow. В основе программного обеспечения данного контроллера лежит модульная архитектура Java Karaf, тем самым обеспечивая необходимую расширяемость контроллера путем установки необходимых дополнительных модулей, или собственной разработки модулей с дальнейшей интеграцией в программную шину контроллера. Также у данного контроллера реализован достаточно удобный и хорошо задокументированный программный интерфейс (с англ. API – Application Programming Interface) формата REST для приложений, работающих поверх северного интерфейса. Стоит отметить, что благодаря модульной архитектуре этого контроллера, существует возможность устанавливать необходимые программные

модули (OSGi-контейнеры) в текущей конфигурации. При этом не требуется полная установка дополнительных модулей, что также снижает нагрузку на аппаратную часть контроллера. Стоит отметить, что подключение собственных разработанных программных модулей хорошо отображено в документации разработчика, которая также находится на официальном сайте проекта OpenDaylight в свободном доступе. В качестве же оркестратора сети используется также открытое решение – OpenStack, который интегрируется также с решением OpenDaylight через программный модуль «OpenStack Neutron».

Для реализации трафик-генератора типового сервиса Интернета Вещей был развернут сервер с операционной системой Linux Ubuntu версии LTS с установленным и настроенным сервисом IoTDM (Internet of Things Data Management), разработанный также сообществом OpenDaylight консорциума The Linux Foundation. Стоит отметить, что данный сервер реализует спецификации OneM2M по межмашинному взаимодействию. Использование решений от OpenDaylight обусловлено тем, что в данное сообщество входит достаточно большое количество международных именитых компаний, которые перенимают данные разработки в свои проприетарные продукты.

На рисунке 2.3.1 можно заметить на сервисном уровне сети несколько приложений, которые реализуют разработанные и представленные в данной диссертации методы. Данные приложения разработаны на языке программирования Python с соответствующими фреймворками и библиотеками, и взаимодействуют с контроллером SDN логически по северному интерфейсу REST, а физически – через прослойку сети управления.

Основное оборудование, которое использовалось при построении лабораторного стенда для проведения исследований отображено в таблице 2.3.1.

Таблица 2.3.1 – Параметры использованного оборудования

№ п.п.	Название (как на рисунке 2.3.1)	Техническое описание
1	SDN-контроллер (ODL) Beryllium SR-4	ЦПУ: Intel Xeon® E3-1220V2 ОЗУ: 16 Гб ОС: Linux Ubuntu 14.04 LTS Версия ПО: Opendaylight Beryllium SR4
2	OpenFlow 1.0 коммутатор Mikrotik RB 201 1UI AS-RM	Mikrotik RB 201 1UI AS-RM Версия ПО: OpenFlow 1.0
3	Коммутатор агрегации №1	Cisco Catalyst 3750G series PoE-24
4	Коммутатор агрегации №2	D-Link DES 3526
5	Коммутатор агрегации №3	Cisco Catalyst 3750G series PoE-24
6	Коммутатор агрегации (служебных каналов)	D-Link DES 3526
7	PaaS Сервер (IoTDM - сервер)	ЦПУ: Intel Xeon(R) E3-1220V2 ОЗУ: 12 Гб ОС: Linux Ubuntu 14.04 LTS Версия: Opendaylight Boron SR2
8	Приложение 1 (мониторинга)* Приложение 2 (Распознавание трафика) * Приложение 3 (Прогнозирование нагрузки на контроллер) *	ЦПУ: Intel Core i5 ОЗУ: 8 Гб ОС: Linux Ubuntu 16.04 LTS

*- все приложения развертывались на одном физическом сервере. В процессе разработки роль сервера выполнял персональный ноутбук, в процессе тестирования – стационарный ПК.

2.4. Трафик исследуемых сервисов

Так как используемая модельная сеть в данной работе является реальной программно-конфигурируемой сетью, для проведения разработки и тестирования

необходимо воссоздать трафик приложений как на уровне передачи данных, так и на уровне управления сетью. Реальный трафик в модельной сети, реализованной в формате стенда возможно воссоздать либо действующими устройствами Интернета Вещей, либо генератором трафика, эмулирующим работу устройств Интернета Вещей. В рамках данной работы был выбран путь использования трафик-генератора, эмулирующего работу устройств Интернета Вещей, который был разработан в данной работе [49]. В работе [49] трафик-генератор использовался для тестирования платформы Интернета Вещей IoTDM и исследования взаимодействия трафика ИВ в Программно-конфигурируемых сетях, в том числе исследования работы динамической автобалансировки нагрузки в подконтрольной сети. Для вышеупомянутого исследования с целью приближения работы эмулятора к реальному масштабу была разработана модель на основе одного из тривиальных сервисов Умного города – система мониторинга экологических параметров тремя видами датчиков.

Реализованная модель в трафик-генераторе работала следующим образом:

1. Согласно заложенной архитектура – построение логического дерева ресурсов;

2. Инициализация устройств Интернета Вещей, где каждое устройство при инициализации к дереву ресурсов платформы отправляет соответствующий запрос на API REST с информацией готовности датчиков.

3. Генерация трафика ИВ. В процессе работы каждое устройство посылает запросы каждую секунду к дереву ресурсов, наполняя его.

В результате получилась модель, имитирующая работу 960 устройств ИВ. При этом, в один момент времени работало с платформой 240 устройств ИВ. Для проведения исследований в диссертации, принцип развертывания трафик-генератора был модифицирован. Если ранее, данный трафик-генератор разворачивался в каждом агрегационном сегменте, то для работ в рамках данной диссертации, трафик генератор был развернут только в одном агрегирующем сегменте, а сама платформа ИВ

располагалась в другом агрегационном сегменте. Также стоит отметить, что в данной работе не требовалось проводить тестирования на различных протоколах ИВ, поэтому был выбран трафик-генератор, работающий с платформой по протоколу HTTP 2.0, при этом содержащий данные в формате JSON. В дополнении, для работы по идентификации трафика не требовался столь интенсивный поток и для эксперимента эмулировалась работы всего 240 устройств.

Используемая платформа на GUI (от англ. Graphical User Interface) отображает дерево ресурсов (рисунок 2.3.2), которое получается при его построении программным способом и последующим заполнением передающимися данными с устройств ИВ (в данном случае – эмулированными устройствами).

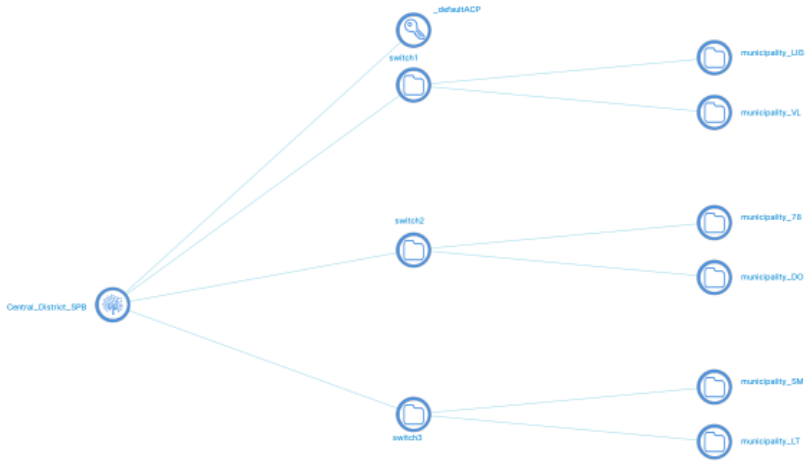

Рисунок 2.3.2 – структура дерева ресурсов (скриншот интерфейса платформы)

2.5. Разработка метода мониторинга и идентификации трафика услуг в сетях связи пятого и последующего поколений

На данный момент существует немало работ, направленных на распознавание типа трафика в сетях связи. Подходы, которые предлагаются в большей мере основаны на периодическом захвате трафика и анализе его заголовков. Такой метод

имеет ряд недостатков, а именно: внесение задержки в поток, реализация аналитического модуля в виде дополнительного аппаратно-программного решения уровня передачи данных (с англ. Data Plane), сложность такого устройства [45]. Также стоит отметить недостаток в масштабировании такого устройства. Для глубокого изучения содержимого трафика используются системы глубокой инспекции пакетов (с англ. DPI – Deep Packet Inspection). В данной работе предлагается реализация аналитической системы на сервисном уровне инфраструктуры Программно-конфигурируемых сетей. В конечном счете система может быть представлена как аппаратно-программный комплекс, так и программное решение (в виде приложения сети). Принципиальная архитектура предлагаемого решения представлена на рисунке 2.5.1.

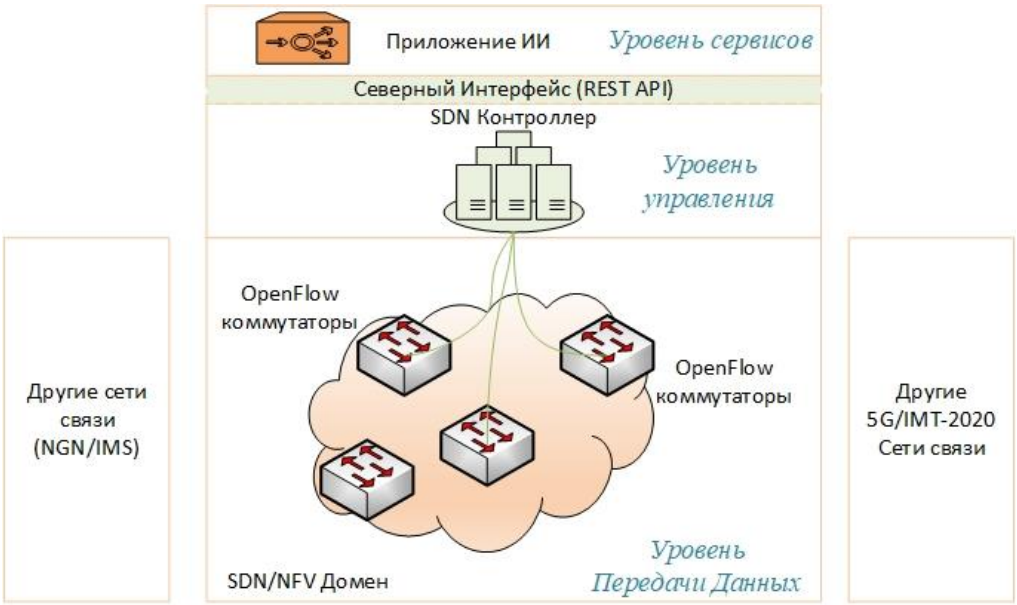

Рисунок 2.5.1 – принципиальная архитектура

Таким образом, появляется возможность переносимости системы, независимости от среды передачи данных и тем более интеграции с уровнем передачи данных. Для аналитической системы все устройства и потоки являются «цифровым объектом», имеющим ряд параметров и функций (действий над параметрами),

представленным набором методов [45]. Такой уровень абстракции позволяет реализовать систему аналитики, которая будет работать с данными о потоках (метаданных) в режиме «На лету». Таким образом, данная система позволит не вносить дополнительных задержек в трафик, а также каким-либо образом изменять его активность (изменения законов распределения, интенсивность и т.п.). Можно сказать, что система «наблюдает» за активностью потоков и составляет «общую картину» происходящего в подконтрольной сети связи.

2.5.1. Входные исследуемые данные

С учетом принципиальной архитектуры предлагаемого решения, а именно реализации приложения, работающего на сервисном уровне с контроллером SDN по REST интерфейсу, входные данные формируются на основе тех данных, которые система может получить через северный интерфейс контроллера и оркестратора сети. Так как в данной работе рассматривается аналитика активности потоков и выявления потоков ИВ на уровне передачи данных, аналитическая система имеет возможность запросить данные таблиц потоков (с англ. Flow Tables) со всех подконтрольных коммутаторов SDN у контроллера. Если проанализировать данные, которые отображаются в двух глобальных частях таблицы: Match Field и Actions, то можно прийти к выводу, что на их основе можно составить метамодель потоков. Сокращенная структура таблицы потоков коммутатора отображены на рисунке 2.5.1.1.

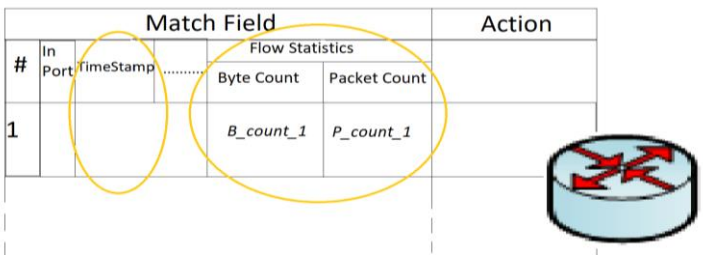

Рисунок 2.5.1.1 – Структура таблицы потоков коммутатора SDN

На рисунке 2.5.1.1 оранжевыми кругами выделены те данные, которые будут использоваться для формирования метамодели об исследуемом потоке в сети. Одной из важных особенностей этих данных является то, что на основе счетчиков «Byte Count» и «Packet Count» нельзя точно определить точную длину пакета в потоке. Так как за один момент времени счетчики могут быть равны: «Byte Count» -1500, «Packet Count» - 2. Соответственно, на основе этих данных нельзя точно определить длину каждого из пакетов, зарегистрированных в потоке за промежуток времени: $\Delta T = 1$ [c].

Стоит также отметить, что счетчики отображают суммарное значение параметров [Byte Count», «Packet Count»]. Однако, кроме данных счетчиков в таблице потоков существует еще один параметр «Time Stamp», который позволяет оценить в каждый момент времени мгновенное значение [ByteCount_delta] и [PacketCount_delta]. Таким образом, за произвольный период времени ΔT, имея отсчеты значений [Byte Count], [Packet Count], [TimeStamp] возможно составить набор данных с установленной структурой данных, где каждый отсчет отображает мгновенное значение [ByteCount_delta] и [PacketCount_delta]. Отсчеты значений [Byte Count], [Packet Count], [TimeStamp] – формируются путем ежесекундных запросов на контроллер сети SDN через программный интерфейс REST API. Структура формируемого на основании запросов $\mathrm{DataSet_{RQ}}$ с «сырыми» данными (2.5.1) и формула (2.5.2) его преобразования в требуемый формат $\mathrm{DataSet_{ML}}$ с мгновенными значениями (2.5.3) приведены ниже.

Пусть, PacketCount_delta – $\mathrm{PC_{delta}}$,

ByteCount_delta – $\mathrm{BC_{delta}}$, а

TimeStamp_deltas – TS = 1 [sec.] = const, тогда:

$$\mathrm{DataSet_{RQ}} = \begin{matrix} [\mathrm{TimeStamp}] & [\mathrm{ByteCount}] & [\mathrm{PacketCount}] \\ \mathrm{TimeStamp_{11}} & \mathrm{ByteCount_{12}} & \mathrm{PacketCount_{13}} \\ \mathrm{TimeStamp_{21}} & \mathrm{ByteCount_{22}} & \mathrm{PacketCount_{23}} \\ \cdots & \cdots & \cdots \\ \mathrm{TimeStamp_{N1}} & \mathrm{ByteCount_{N2}} & \mathrm{PacketCount_{N3}} \end{matrix} \quad (2.5.1);$$

$$\begin{cases} BC\_delta_{\mathrm{N2}} = ByteCount_{N2} - ByteCount_{(N-1)2}, & if\ N \geq 1 \\ PC\_delta_{\mathrm{N2}} = PacketCount_{N2} - PacketCount_{(N-1)2}, & if\ N \geq 1 \end{cases} \quad (2.5.2);$$

$$\mathrm{DataSet_{ML}} = \begin{matrix} [\mathrm{TimeStamp}] & [\mathrm{ByteCount}] & [\mathrm{PacketCount}] \\ \mathrm{TS} & \mathrm{BC_{delta_{12}}} & \mathrm{PC_{delta_{13}}} \\ \mathrm{TS} & \mathrm{BC_{delta_{22}}} & \mathrm{PC_{delta_{23}}} \\ \cdots & \cdots & \cdots \\ \mathrm{TS} & \mathrm{BC_{delta_{N2}}} & \mathrm{PC_{delta_{N3}}} \end{matrix} \quad (2.5.3);$$

При этом расчет суммарных значений параметров за установленный промежуток времени производится по следующим формулам (2.5.4, 2.5.5):

$$ByteCount_{\Delta\mathrm{T}} = \sum_{N=1}^{N=\Delta\mathrm{T}/\mathrm{TS}} BC_{delta_{N2}} \quad (2.5.4);$$

$$PacketCount_{\Delta\mathrm{T}} = \sum_{N=1}^{N=\Delta\mathrm{T}/\mathrm{TS}} PC_{delta_{N2}} \quad (2.5.5);$$

Как уже было сказано выше, для решения вышеописанной задачи был проведен комплексный анализ математических методов классификации с учетом особенностей входных, анализируемых данных о потоках, а также особенности изначального требования – работы системы в режиме «на лету». В результате проведенного анализа был выбран подход использования нейронных сетей [45]. В соответствии с поставленной целью и особенностями входных данных $\mathrm{DataSet_{ML}}$была выбрана соответствующая архитектура нейронной сети.

2.5.2. Нейронная сеть и Deep Learning

На данный момент существует большая разновидность нейронных сетей. Одной из типовых задач, является классификация. Один из самых распространенных способов классификации – способ на основе описаний объектов с использованием признаков, в котором каждый объект характеризуется набором числовых или нечисловых признаков. Однако для некоторых типов данных открытые признаки не дают точности классификации, например, цвет точек изображений или цифровой звуковой сигнал. Причина заключается в том, что эти данные содержат скрытые признаки. Deep Learning представляет собой набор алгоритмов машинного обучения, которые позволяют моделировать высокоуровневые абстракции в данных, иными словами, выделять из данных скрытые признаки, при этом нейронные сети содержат более, чем 2 скрытых слоя. Поэтому, учитывая особенности объекта (трафик) и его признаков (числовые – статистические ряды, являющиеся метаданными) была выбрана нейронная сеть с Deep Learning.

Разработка и обучение нейронной сети проводились на основе высокоуровневого языка программирования Python с его различными библиотеками и фреймворками. В качестве искусственной нейронной сети была выбрана рекуррентная нейронная сеть с дополнительными слоями LSTM (Long Short-Term Memory, с англ. – долговременной краткосрочною памятью). Сеть LSTM является универсальной в том смысле, что с достаточным количеством нейронов она имеет возможность выполнять любые вычисления, которые может выполнять обычный компьютер. В данной работе глубинный модуль LSTM, как часть искусственной нейронной сети (ИНС) позволяет выявлять закономерности влияния данных предыдущих отсчетов на текущие с учетом большого разброса значений между ними. То есть, например, периодичность, что как раз может проявляется в трафике ИВ, самоподобность характера которого приводится во многих работах.

Так как выбранная архитектура нейронной сети реализует принцип «обучение с привлечением учителя», требуется составить обучающие наборы данных с маркированными данными, после чего сохранить состояние обученной сети. Для обучения нейронной сети входной $\mathrm{DataSet_{ML}}$ был преобразован в $\mathrm{DataSet_{ML}}_{train}$ путем добавления нового столбца данных, в каждой строчке которого стоял идентификатор статистической выборки. Соответственно, для обучения распознавания большего типа трафика данный обучающий набор данных требуется расширить, пометив соответствующую статистическую выборку меткой трафика, например - IoT. Таким образом, структура обучающего $\mathrm{DataSet_{ML}}_{train}$ выглядит следующим образом (формула 2.5.6):

$$\mathrm{DataSet_{ML}} = \begin{matrix} [\mathrm{TypeOfTraffic}] & [\mathrm{TimeStamp}] & [\mathrm{ByteCount}] & [\mathrm{PacketCount}] \\ \mathrm{IoT} & \mathrm{TS} & \mathrm{BC_{delta_{12}}} & \mathrm{PC_{delta_{13}}} \\ \mathrm{IoT} & \mathrm{TS} & \mathrm{BC_{delta_{22}}} & \mathrm{PC_{delta_{23}}} \\ \cdots & \cdots & \cdots & \cdots \\ \mathrm{Video} & \mathrm{TS} & \mathrm{BC_{delta_{N2}}} & \mathrm{PC_{delta_{N3}}} \\ \mathrm{others} & \mathrm{TS} & \mathrm{BC_{delta_{(N+1)2}}} & \mathrm{PC_{delta_{(N+1)3}}} \end{matrix} \quad (2.5.6)$$

Сеть LSTM получает данные фиксированной длины, поэтому данные делятся на сегменты по 200 строк или 10 секунд.

Теги типа трафика преобразуются в унитарный код с помощью, встроенных в библиотеку Python соответствующих функций. Данные делятся на тренировочные и тестовые наборы в отношении 8:2.

Архитектура сети содержит 2 полносвязанных глубинных уровня LSTM и 2 полносвязанных глубинных уровня RNN (Recurrent Neural Network, с англ. Рекуррентная нейронная сеть), каждый из которых содержит 7 скрытых нейронов.

Параметры тренировки ИНС:

- Оптимизатор: Adam;
- Количество эпох обучения: 40;

- Количество образцов на итерацию: 1024;
- Скорость обучения: 0,0025.

На рисунке 2.5.2.1 отображены входной слой, состоящий из трех нейронов, на вход которых подавались соответствующие данные из $\mathrm{DataSet}_{\mathrm{ML}_{train}}$. Входные три нейрона с первым вложенным уровнем, состоящим из нейронов типа LSTM. Второй вложенный уровень также состоит из нейронов типа LSTM и связан как с первым, так и с третьим полносвязанным принципом. Третий и четвертый вложенные уровни искусственной нейронной сети строятся на полносвязанных нейронах типа RNN.

Выходными нейронами на данный момент являются два нейрона, которые отображают результат работы нейронной сети. Соответственно, если составить $\mathrm{DataSet}_{\mathrm{ML}_{train}}$ с большим количеством типов помеченной статистики трафика, то количество выходных нейронов будет увеличиваться.

Стоит отметить, что существует определенный порог, превышая который, для обучения нейронной сети требуется увеличивать количество нейронов в каждом вложенном слое.

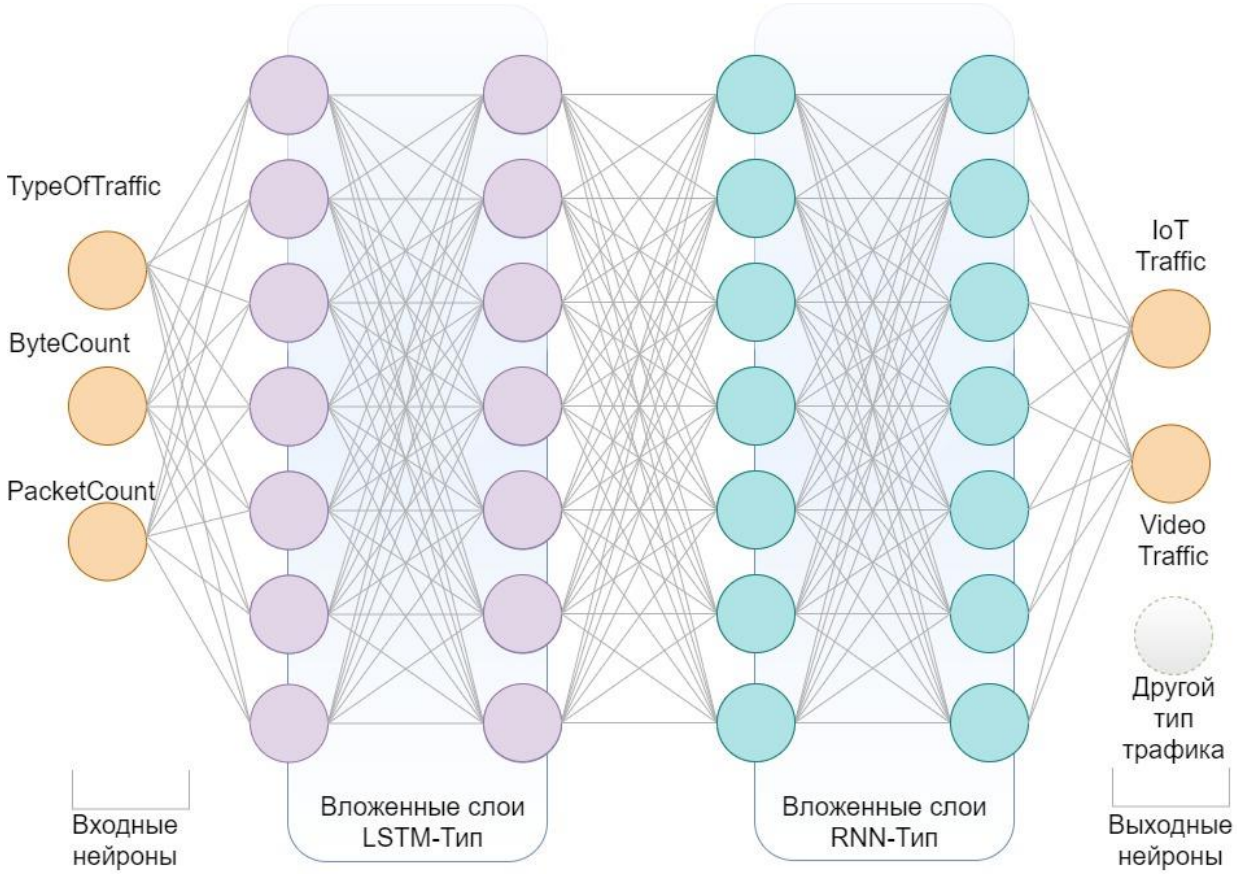

Рисунок 2.5.2.1 – Архитектура ИНС

2.5.3. Исследуемая модель. Общая архитектура сегмента

Исследуемый метод детектирования потоков реализован в виде программного модуля на языке программирования Python версии 2.7 для аналитической системы, реализованной на основе MVC паттерна. Данное программное обеспечение работает поверх серверного программного интерфейса API контроллера программно-конфигурируемой сети и оркестратора сети лаборатории.

Для формирования обучающих для Искусственной Нейронной Сети наборов данных были определены специальные условия:

- Весь любой другой трафик, не относящийся к проводимому исследованию, должен быть выключен и не проходить в SDN-сегмент;
- Сеть должна быть модифицирована так, чтобы была возможность однозначно идентифицировать поток с генерируемым трафиком в таблице потоков коммутаторов

После выполнения установленных условий процесс эксперимента выглядел следующим образом: с помощью графического интерфейса после старта генерации трафика обнаруживался необходимый поток (ID – идентификатор) в таблице потоков коммутаторов. Далее – приложению указывался точный номер потока в таблице потоков и приложение запускало соответствующий процесс по запросу статистических данных через северный интерфейс контроллера Программно-конфигурируемой сети и их дальнейшему сохранению для обучения ИНС. Таким же образом формировался обучающий набор данных для трафика «Video on Demand», при этом генераторы трафика ИВ были дезактивированы и сохранялись выше озвученные условия.

После сбора необходимых данных с сегмента Программно-конфигурируемой сети, аналитический программный модуль запускал соответствующий алгоритм по первичной обработке данных для подготовки их в качестве обучающего набора ИНС. После успешного обучения разработанная ИНС сохраняла свое состояние (получившаяся архитектура, веса, и другие параметры). На следующей стадии, при «боевом» распознавании трафика, данное сохраненное состояние ИНС может быть использовано другим ПО, которое формирует тестовый набор данных после указания номера исследуемого потока в таблице потоков. Общая архитектура сегмента и исследуемый сценарий, отображены на рисунке 2.5.3.1.

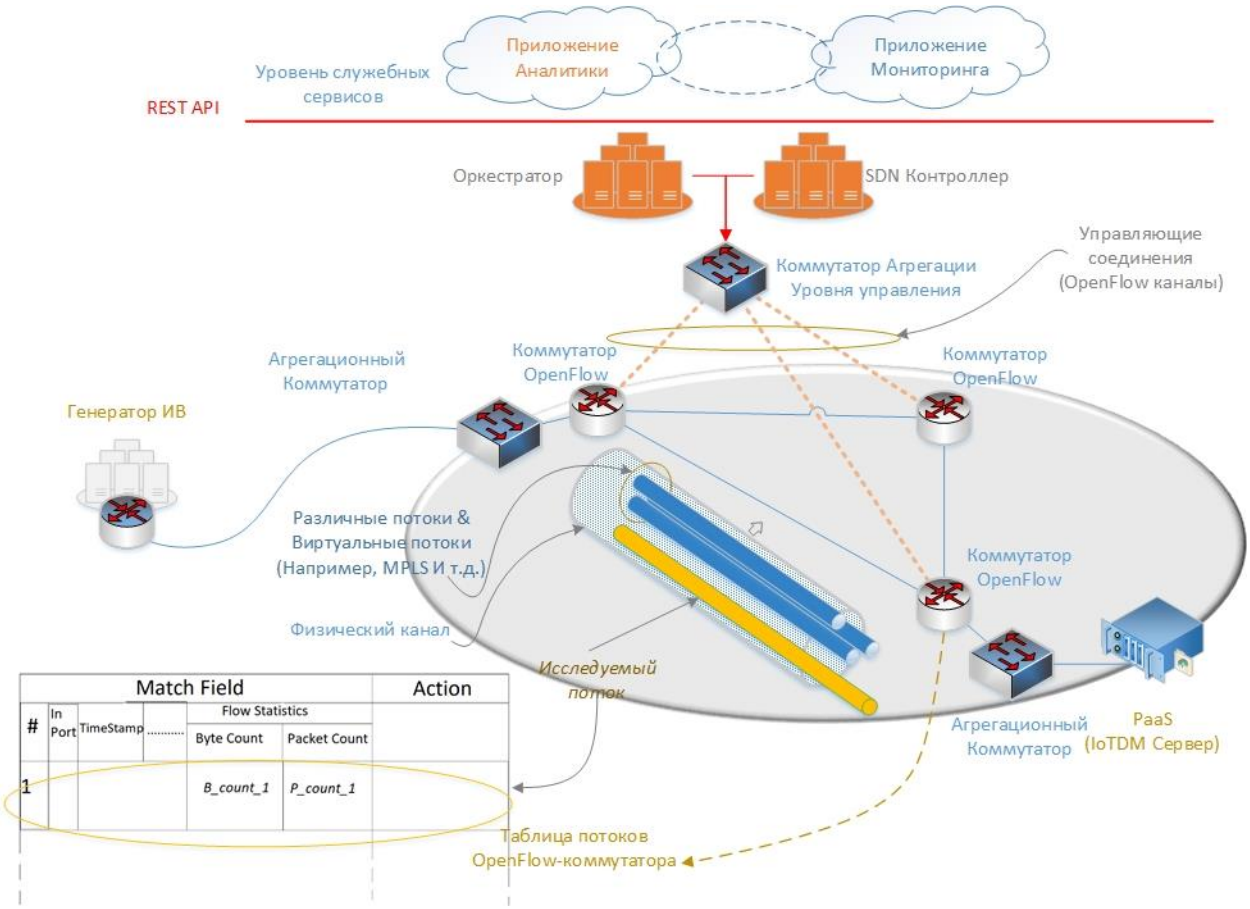

Рисунок 2.5.3.1 – Общая архитектура сегмента с исследуемым сценарием

На рисунке 2.5.3.1 отображены все главные элементы модельного стенда программно-конфигурируемой сети с серверами (генерации трафика, платформы ИВ), в том числе отображен исследуемый сценарий с объектом исследования и параметрами.

2.5.4. Результаты тестирования метода

Как уже ранее отмечалось, для проведения обучения ИНС были соблюдены определенные условия: весь сторонний трафик был выключен (дезактивирован), сеть модифицирована таким образом, чтобы была возможность однозначно детектировать

поток с трафиком ИВ или «Video on Demand» в таблице потоков SDN-коммутаторов. Генератор трафика ИВ моделировал работу 240 устройств Интернета Вещей. Каждое устройство генерировало HTTP 2.0 пакет и в теле пакета передавало значения датчиков. Далее, модулю программного обеспечения указывался точный номер потока в таблице потоков коммутаторов SDN, и приложение запускало процесс сохранения данных. Также формировался обучающий набор данных для трафика «Video-on-Demand», при этом использовалось уже программное обеспечение для генерации трафика по запросу (VLC).

В результате был получен $\mathrm{DataSet}_{\mathrm{ML}_{train}}$, который был сформирован из двух маркированных наборов данных (IoT, Video). Далее $\mathrm{DataSet}_{\mathrm{ML}_{train}}$ подавался на вход нейронной сети, конфигурация которой отображена выше.

По полученному $\mathrm{DataSet}_{\mathrm{ML}_{train}}$была построена диаграмма разброса значений. Диаграмма приведена на рисунке 2.5.4.1. На данной диаграмме визуально возможно выделить несколько кластеров распределения точек, в области которых превалирует соответствующее значение среднего значения длины пакета в потоке. Например, в потоке ИВ, в большинстве случаев (плотная область точек), средняя длина пакета составляет примерно 400 байт, при этом параметр может варьироваться. Также стоит заметить выброс на значение средней длины пакета 470 байт.

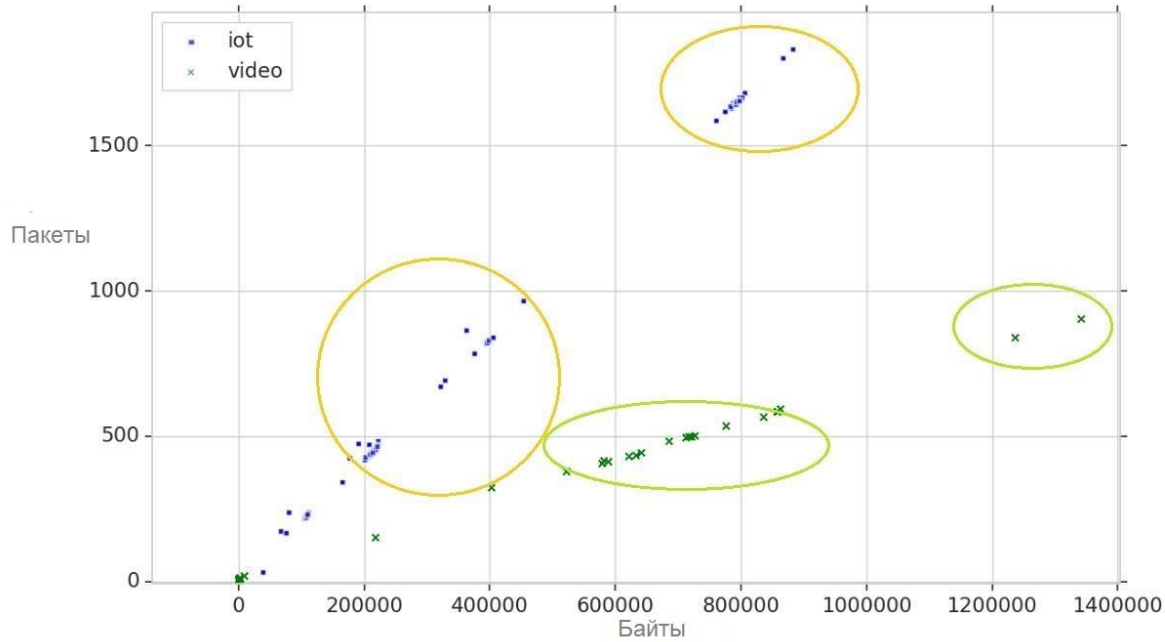

Рисунок 2.5.4.1 – Диаграмма разброса значений $\mathrm{DataSet}_{\mathrm{ML}_{train}}$

В дополнение, заметен выброс значений в статистическом ряду метаданных видео потока, при этом в большинстве случаев средняя длина пакета (в плотной области точек) составляет примерно 1450 байт, при этом данный критерий также варьируется, но незначительно.

После того, как обучающий набор данных был сформирован, согласно алгоритму, изложенному выше в данной диссертации, программный модуль разработанного приложения, реализующий Искусственную Нейронную Сеть, начинал работу. В течении процесса обучения нейронной сети приложением наблюдались следующие параметры:

- Train Accurancy (Точность обучения);
- Test Accurancy (Точность прохождения теста ИНС);
- Train loss (Ошибки во время обучения);
- Test loss (Ошибки во время прохождения теста ИНС);

График, отображающий данные параметры в процессе прохождения эпох обучения, приведен на рисунке 2.5.4.2. В дополнении, на рисунке 2.5.4.3 – приведено приближение (увеличен масштаб части графика на рисунке 2.5.4.2) для более детального отображения разницы между графиками.

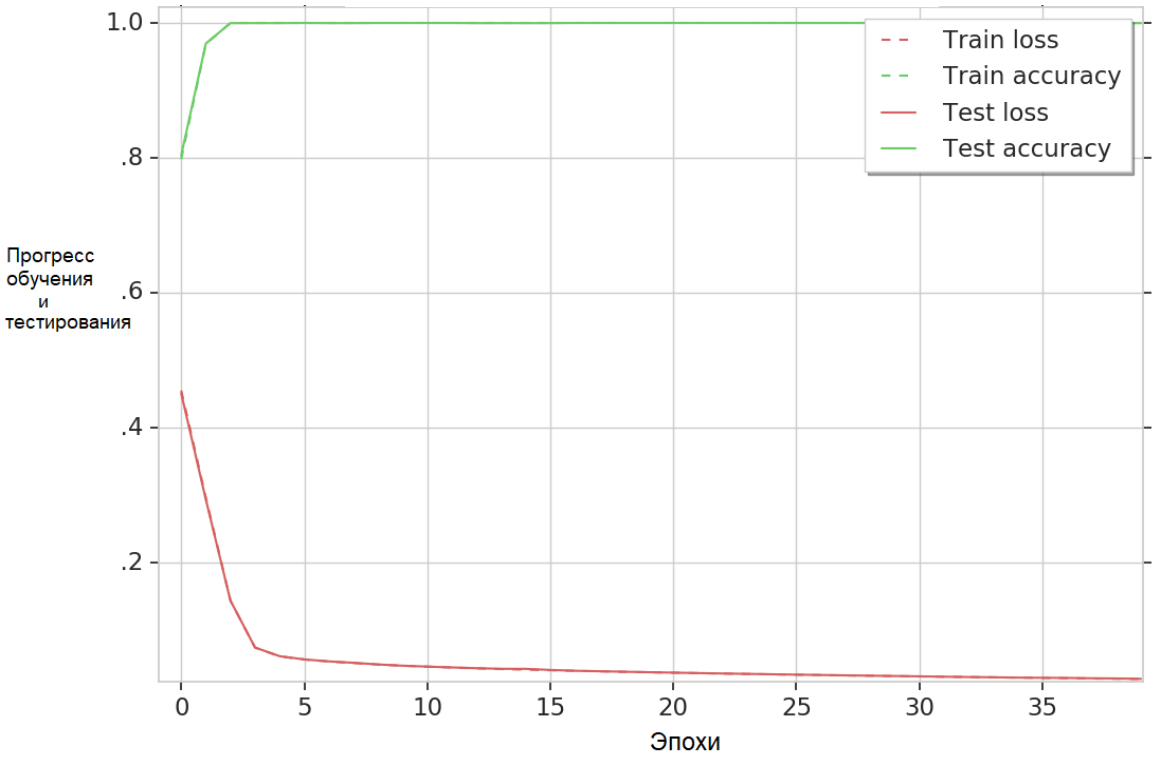

Рисунок 2.5.4.2 – График обучения и тестирования ИНС

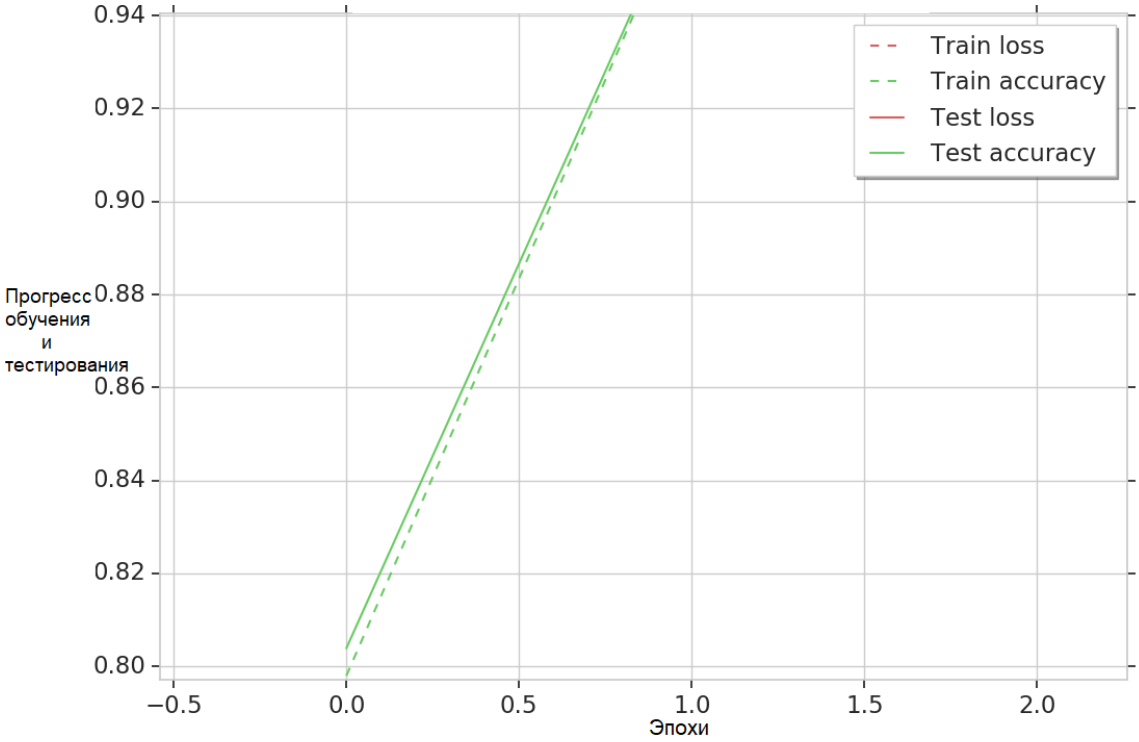

Рисунок 2.5.4.3 – График обучения и тестирования ИНС (масштаб)

Кроме выше приведенных графиков, приведенных на рисунках 2.5.4.2 и 2.5.4.3 – была рассчитана матрица противоречий ИНС (в терминологии ИНС с англ. – Confusion Matrix) обучения искусственной нейронной сети. Получившаяся матрица приведена на рисунке 2.5.4.4.

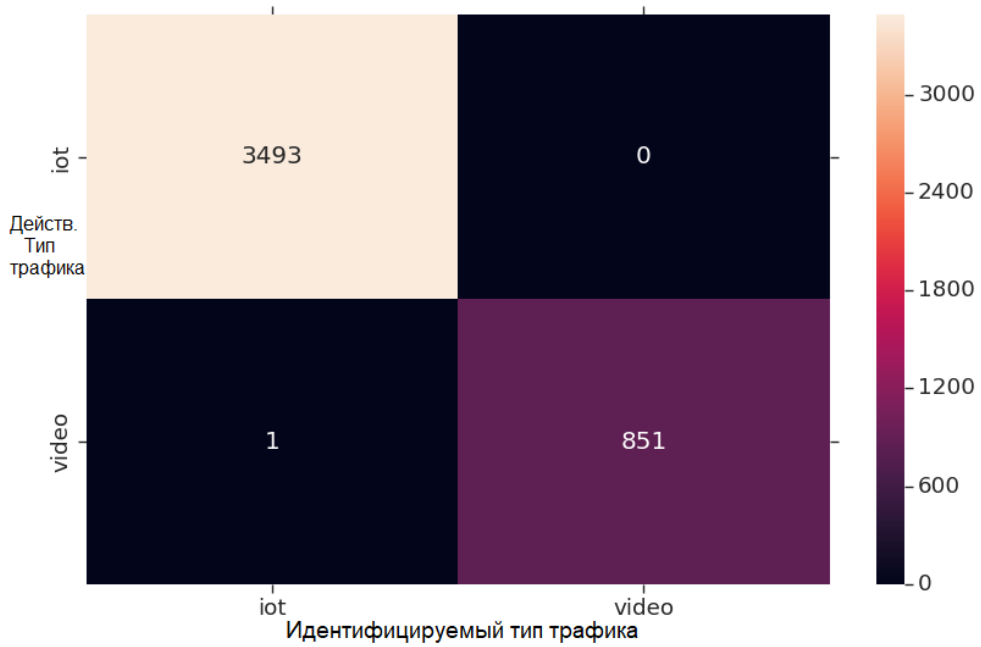

Рисунок 2.5.4.4 – Матрица противоречий (Confusion Matrix)

На рисунках 2.5.4.2 и 2.5.4.3, где отображен процесс обучения сети, хорошо видно, что для поставленной задачи разработанная искусственная нейронная сеть успешно прошла процесс обучения. В результате процесса обучения нейронной сети и проверки ее работы на тестовых наборах данных в обученном состоянии разработанная нейронная сеть может идентифицировать генерируемый поток Интернета Вещей с вероятностью 99,7 %. А также соответственно и генерируемый видео поток. С помощью матрицы противоречий (рисунок 2.5.4.4) можно заметить, что сеть один раз ошиблась, идентифицировав Видео поток, как поток Интернета Вещей. Стоит также отметить, что выбранная архитектура оказалась эффективной для решения поставленной задачи. При меньшем значении нейронов во вложенных уровнях, нейронная сеть работает не стабильно, то есть, совершает много ошибок, в том числе в процессе обучения, а не только конечной работы. При большем значении нейронов – сеть не эффективна, так как требует большего значения вычислительных

ресурсов. Стоит также отметить, что при преувеличении определенного значения нейронов во сложенном слое – нейронная сеть переходит в режим «переобученности».

В дополнение стоит отметить, что полученная вероятность распознавания потоков данных (ИВ и Видео) получилась достаточно высокой и при увеличении количества типов распознаваемого трафика (при увеличении – формируется новый, расширенный набор данных для обучения ИНС) данная вероятность может понизиться, однако находится в допустимых пределах (80-90%).

2.6. Выводы по Главе 2

1. Сети связи с ультрамалыми задержками и требования к ним задают вектор развития сетей и услуг в сторону их децентрализации. Данный класс сетей призван обеспечить работу новых типов услуг, таких как, Тактильный Интернет, Медицинские сети, Цифровые Аватары, беспилотный транспорт и так далее. Для достижения поставленных требований необходимы изменения не только в технологиях построения сетей связи и услуг, но и изменения в методах мониторинга и управления трафиком, в задачи которого включается в том числе идентификация типа трафика.

2. Учитывая возможность программируемости SDN сетей и функциональные возможности протокола управления OpenFlow, был разработан метод идентификации трафика в сетях связи на основе метаданных потоков и искусственной нейронной сети.

3. На базе модельной сети лаборатории «Программно-конфигурируемых сетей» было произведено обучение ИНС и последующее тестирование, результаты которых отображены в Главе 2.

4. Результат тестирования показал работоспособность разработанного метода по факту идентификации трафика и отсутствия внесения дополнительных задержек и какого-либо изменения структуры потоков.

ГЛАВА 3. СТРУКТУРА И МЕТОД ВЗАИМОДЕЙСТВИЯ ТУМАННЫХ И ГРАНИЧНЫХ ВЫЧИСЛЕНИЙ С ПОДДЕРЖКОЙ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ УСЛУГ

3.1. Необходимость перехода к децентрализации облачных вычислений

Разработка сетей и систем пятого поколения 5G/IMT-2020 сталкивается со многими новыми вызовами/задачами, в том числе с ультрамалыми задержками и сверхвысокой плотностью устройств. В решении данных задач предлагаются различные направления исследования и разработок. Одним из направлений исследований является организация распределенных вычислений. Таким образом, предлагается исключить возможные потери, а в основном – убрать сетевую составляющую круговой задержки. Одними из наиболее известных новых технологий ИКТ, которые могут разрешить озвученные выше вызовы через минимизацию сетевой части круговой задержки, являются пограничные системы с множественным доступом (с англ. MEC – Multi-access Edge Computing), Туманные вычисления (с англ. Fog computing) и взаимодействия устройство-устройство (с англ. D2D – Device-to-Device communications).

При объединении сетевых технологий SDN/NFV и инфраструктурных технологий распределенных вычислений (MEC/Fog) может быть получен положительный синергетический эффект, проявляющийся в следующих особенностях: быстрая масштабируемость сети и облаков, полный контроль потоков передачи данных, слайсинг ресурсов, устойчивость, самовосстановление, быстрая живая миграция услуг в любую точку сети, возможность разработки сервисов Искусственного интеллекта по модели SaaS и другие. Набор данных особенностей

обеспечивает возможность предоставления тех требований к качеству предоставляемых услуг, которые стали вызовом для всего мирового научного сообщества [50].

Освещая вопрос предоставления необходимых параметров качества услуг, стоит отметить немаловажный вопрос разработки и реализации самих услуг, которые представлены в виде программного обеспечения (ПО).

3.2. Анализ основных архитектурных подходов к построению программного обеспечения услуг

Программное обеспечение услуг, в современное время являются достаточно емкими (по объему) информационными системами, большая часть которых еще «хранит» достаточно старую архитектуру ПО и методы - простыми словами «монолит». Однако озвучивая данный вопрос, стоит обратить внимание на само понятие слова «Архитектура» относительно информационных и телекоммуникационных систем. Исходя из понятия Информационной системы ее архитектура с конструктивной точки зрения представляет собой набор ключевых решений, неизменных при изменении бизнес-технологии в рамках бизнес-видения (концепции). Отсюда вытекает следующий вывод: если возникает необходимость изменения ключевых решений в ИТ продукте при изменении бизнес-технологии (бизнес-модели/процессов/продуктов) в раках бизнес-видения (концепции), то резко возрастает стоимость владения системой. Как следствие, возникает понимание важности и необходимости принятия архитектуры системы как стандарта компании ввиду значимости архитектурных решений и их устойчивости по отношению к изменениям бизнес-технологий. Важно заметить, что вопрос архитектуры систем должен затрагиваться на уровне концептуального проектирования, а в некоторых

случаях на стадии технико-экономического обоснования системы [51]. Архитектура обычно определяется как «набор ответов» на следующие вопросы:

- Для выполнения какой цели предназначена система?
- Какие целевые функции системы?
- На какие части она разделяется?
- Каким образом данные части взаимодействуют?
- Где данные части размещены (виртуально/физически)?
- Какие основные принципы функционирования система реализует?

Таким образом, архитектура системы является логическим построением, моделью и полностью влияет на совокупную стоимость владения либо, если ИТ-продукт, то стоимостью доработки/изменения/усовершенствования, через набор связанных с ней решений по выбору средств реализации, операционной платформы, СУБД, БД, телекоммуникационных средств, архитектуры самого программного обеспечения (подходы/дизайна реализации ПО).

Любая услуга, предоставляемая в сети по факту, является Программным обеспечением (обычно высокоуровневым) – продуктом компании-бенефициаром. Соответственно, те же вопросы могут быть заданы не только к системе ИТ в целом, но и к Программному обеспечению (ПО), реализующему либо целый ИТ-продукт, либо его составную часть.

В современное время в мире разработки ПО было получено достаточное количество знаний в области разработки эффективного кода, методик, принципов и подходов к реализации ИТ-продукта (с точки зрения кода). В том числе существуют различные архитектурные решения. К ним можно отнести такие, как: монолитная, многоуровневая, управляемая событиями, бессерверная и другие, микросервисная. Например, бессерверный архитектурный стиль применим к приложениям, которые в качестве решения используют другие сервисы (третьи лица) для того, чтобы решить проблему загруженности бэкенда и серверов. Например, бессерверная архитектура

может помочь сократить время на работе с регулярными задачами сервера. Одним из известных примеров бессерверного API является сервис Amazon AWS «Lambda». Также к данному архитектурному стилю можем отнести и так называемое сервелет-программирование.

Однако особенно в последние 2-3 года активно развивается микросервисный архитектурный стиль разработки и реализации Программного обеспечения после его успешного применения в таких известных ИТ-продуктах, как Amazon, Netflix. Особенно данная архитектура важна для инфраструктурных решений, например платформы Интернета Вещей, Веб-платформы, реализующие больше, чем один продукт и другие. Стоит отметить, что в последнее время не только новые продукты разрабатывают в данном архитектурном стиле, но и существующие продукты некоторые компании буквально переписывают, либо пытаются найти тот уровень декомпозиции бывших монолитов, чтобы сохранить баланс стоимость разработки и перехода на новую архитектуру с сохранением всех функций, возможностей, и получением тех преимуществ, которые дает Микросервисная архитектура ПО.

Программное обеспечение представляет собой набор программных блогов – микросервисов и логику их взаимодействия. При этом, микросервис может быть представлен в виде целой подсистемы, реализующий набор функций услуги, или в виде достаточно атомарной структуры ПО – функции. Уровень «атомарности» и независимости микросервисов определяется командой разработки в каждом конкретном проекте, в том числе учитывая инструменты разработки ПО. Однако в идеале, микросервисы должны быть независимы, и при необходимости масштабирования функций системы, те или иные микросервисы должны быть размножены так, чтобы удовлетворять возросшим требованиям. При этом, при клонировании микросервисов, они должны на автоматическом уровне подключатся к общему «микросервисному организму» информационной системы так, чтобы при дальнейших процессах эти микросервисы без ошибок выполняли свои функции, взаимодействия с другими микросервисами, с клиентским ПО и так далее. Стоит

также отметить, что так называемое клонирование/размножение микросервисов обеспечивается в том числе принципами виртуализации программного обеспечения с помощью различных инструментов, в том числе такого как контейнеризация. Живая миграция же обеспечивается при помощи инструментов оркестрации вычислительных систем.

Таким образом, вышеупомянутая синергия инфраструктурных технологий может быть гармонично добавлена архитектурой разработки и принципами развертывания микросервисного программного обеспечения услуг. Особенно в данной тематике вызывает интерес объединения решений туманных вычислений и микросервисов. Данное объединение позволит достичь достаточно «плотного тумана» при надлежащей атомизации микросервисов услуг [50].

В данной главе диссертации предлагается новая структура так называемых динамических туманных вычислений, которая может учитывать спрос на услуги в той или иной географической области, обеспечивать миграцию необходимых микросервисов услуги для предоставления высоких требований к качеству услуг. Относительно практических исследований, в статье рассматриваются алгоритмы K-средних для определения центра скопления пользователей, а также алгоритм Роевого Интеллекта PSO (Practical Swarm Optimization) для определения устройства тумана, выполняющего необходимые требования к миграции на него соответствующего микросервиса или набора микросервисов услуги.

3.3. Разработка структуры и метода взаимодействия распределенных вычислений с микросервисной поддержкой

3.3.1. Предлагаемая структура/фреймворк

Как уже ранее было приведено, для достижения большего синергетического эффекта от технологий SDN, NFV, Fog, MEC, в настоящее время рассматривают их совместное использование. В рамках единой сетевой и вычислительной инфраструктуры необходимо обеспечить платформенные решения для необходимого абстрагирования и предоставления верхнеуровневым решениям (сервисам) необходимые функции. Если рассматривать данный вопрос с точки зрения всем знакомой иерархии IaaS/PaaS/SaaS, то данные решения реализуют уровень PaaS и обеспечивают необходимую логику взаимодействия всех сетевых и вычислительных ресурсов (так называемая оркестрация), а также в дополнение обеспечивают наложенную бизнес/системную логику решений (структуры или иначе – фреймворки, с англ. framework). Данные фреймворки отчасти реализуют системную логику услуг [50]. К примеру, международная рекомендация МСЭ-Т Q.3745 «Протокол для приложений Интернета Вещей с ограничением по времени поверх программно-конфигурируемых сетей» описывает кроме протокола и закладываемый фреймворк, который описывает структуру системы и соответствующие системные процессы, где уже используется сам протокол взаимодействия между сервером услуги Интернета Вещей и системным приложением программно-конфигурируемой сети.

На рисунке 3.3.1.1 приведен предлагаемый фреймворк/структура взаимодействия распределенных вычислений с микросервисной поддержкой. Данный фреймворк призван объединить облачные структуры многоуровневых пограничных вычислений и туманных вычислений, а также сетевую инфраструктуру, построенную на основе технологий SDN/NFV, учитывая в том числе уровень оркестрации. Таким образом, объединенная инфраструктура позволяет реализовывать новые подходы к распределению вычислений, в том числе на границе сети и также на самих конечных устройствах, в том числе устройствах Интернета Вещей.

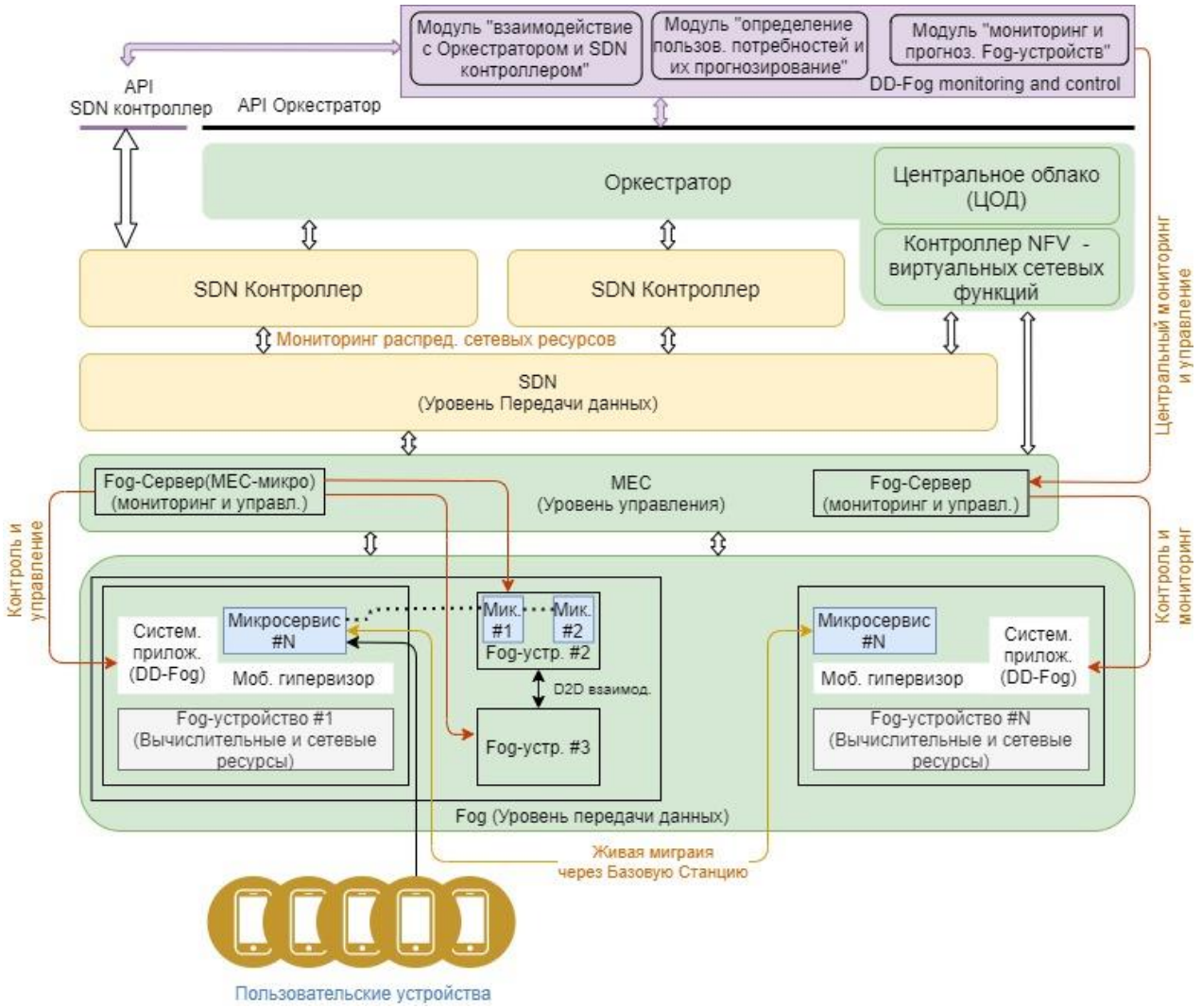

Рисунок 3.3.1.1 – Предлагаемый фреймворк/структура взаимодействия

В данном случае рассматривается подход к управлению через северный инфраструктурный интерфейс (уровень приложений оркестратора вычислительных ресурсов, уровень приложений контроллера(-ов) сети SDN/NFV. Данная возможность позволяет реализовывать бизнес-логику услуг как сетевого, так и сервис-оператора. Но, кроме того, благодаря заложенному уровню абстрагирования от физических ресурсов, есть возможность создания служебных приложений-сервисов, которые реализуют системную и бизнес-логику оператора по управлению устройствами подконтрольной инфраструктуры. В данном случае, как уже было оговорено в ряде научных статей [2-4, 11, 52, 53], а также документах международных

стандартизирующих организаций, путь автоматизации, а также интеллектуализации сети лежит через разработку сервисов, алгоритмы обработки данных которых принадлежат классу Искусственного Интеллекта.

Таким образом, система мониторинга и управления, являющаяся инфраструктурным приложением (программный комплекс), реализующая в себе в качестве алгоритмов алгоритмы машинного интеллекта, позволяет в определенной степени автоматизировать и более того сделать инфраструктуру интеллектуальной, не имеющей потребности в постоянном контроле и управлении со стороны людей, высококвалифицированных специалистов. Разрешение данной задачи позволит уменьшить такой показатель, как OPEX, путем сокращения штата квалифицированных администраторов сетей. А также позволит качественно изменить сети с точки зрения их реакции на действия с внешней стороны (рост трафика, изменение профиля трафика) - сеть будет подстраиваться под текущие потребности, перераспределять нагрузку, учитывая правила QoS. Кроме того, такая сеть будет учитывать ее мощностные характеристики, мониторить и предоставлять прогнозную аналитику по необходимости увеличения/уменьшения ее мощностных параметров (сетевые ресурсы, вычислительные ресурсы объединенной облачной инфраструктуры).

В рамках рассматриваемой архитектуры фреймворка (рисунок 3.3.1.1) выделены ряд элементов, реализующие определенные вычислительные функции. Структура MEC подразумевает иерархичность элементов с подчинением нижележащих вычислительных слоев облаков. При этом, в структуре туманных вычислений как таковой строгой иерархичности не наблюдается. При этом, если грамотно построить туманные структуры, учитывая технические характеристики и вычислительные способности каждого из Fog-устройства, их динамическое распределение в пространстве, а также технологии коммуникаций устройство-устройство (или на англ. D2D – Device-to-Device) возможно достичь большего синергетического эффекта. В данной работе с целью выделения «шлюза

коммуникации» с сетью Интернет предлагается использование сервера нижнего уровня структуры MEC – Micro Cloud (с англ. микро-облако). На рисунке 3.3.1.1 также отображены возможные взаимодействия. Например, взаимодействие D2D между Fog-устройствами, взаимодействие Fog-устройства и Микро-облака структуры MEC, выполняющий функцию сервера Fog-зоны.

Стоит напомнить, что в рамках данной структуры происходит динамическое распределение Fog-устройств, в результате чего происходит изменение ресурсоемкости каждой из Fog-зон. При этом, поверх этой динамической вычислительной инфраструктуры происходит миграция микросервисов приложения той или иной услуги. В рамках данного фреймворка реализуется следующая логика взаимодействия функциональных элементов при подключении нового Fog-устройства. Логика взаимодействия отображена в виде диаграмм сообщений на рисунке 3.3.1.2.

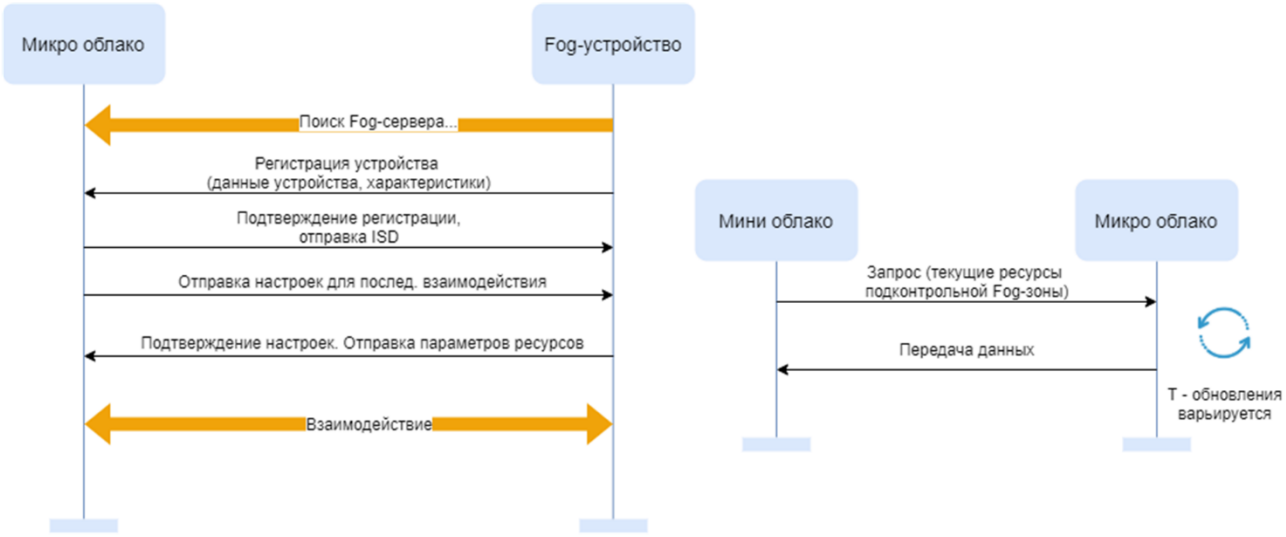

Рисунок 3.3.1.2 – Общая диаграмма взаимодействия основных элементов фреймворка

Как уже выше было обозначено, в рамках фреймворка предполагается динамическое перераспределение Fog-устройств. Для этого при переходе с одной Fog-зоны в другую, либо при первичном подключении, каждое Fog-устройство отправляет

широковещательный запрос с целью обнаружения ближайшего Micro Cloud структуры MEC. После обнаружения Fog-устройство направляет данные для регистрации в обнаруженной Fog-зоне. Для регистрации Fog-устройство передает следующие данные о своих вычислительных виртуальных ресурсах, выделенных для передачи их в аренду, как часть Fog-зоны:

- ЦПУ (количество ядер и тактовая частота ядра). Данный параметр на стороне сервера позволяет рассчитать производительность устройства (части выделенной мощности для выделенного виртуального пространства);
- Выделенное количество логического ОЗУ для виртуального пространства;
- Выделенное количество логического ПЗУ для виртуального пространства;
- Коды активных и доступных технически беспроводных технологий (например, 802.11ac, LTE, 5G, и т.д.);
- Разрешенная скорость передачи данных для сторонних сервисов (данный параметр может быть ограничен с целью сохранения качества основной связи для пользователя);
- Поддерживаемая система виртуализации и оркестрации;
- Поддерживаемый формат микросервисов;

После чего, главный сервер Fog-зоны определяет возможность подключения данного Fog-устройства с указанными параметрами. В случае положительного решения Fog-сервер направляет сообщение-подтверждение ожидающему подключения Fog-устройству. В данном сообщении Fog-сервер также передает уникальный сгенерированный ISC (DB, с англ.: внутренний системный код). Данный код необходим для последующих процессов взаимодействия Fog-устройства со статической облачной инфраструктурой MEC. В том числе для системы управления

миграцией микросервисов приложений, с целью однозначной идентификации каждого из Fog-устройств в Fog-зонах в условиях их физического перемещения.

После данного сообщения Fog-сервер производит передачу параметров соединения. В результате, подключаемое Fog-устройство отправляет сообщение-подтверждение Fog-серверу о готовности. Далее, Fog-сервер получает права на использование в своих целях (агрегирование) вычислительных ресурсов, предоставленных Fog-устройствами вычислительных и сетевых ресурсов. Стоит отметить, что далее происходит взаимодействие на более высоком уровне вычислительной архитектуры, а именно: периодическая передача данных Микро-облаку MEC о подконтрольной Fog-зоне по запросу от Мини облака структуры MEC. При этом периодичность запросов является переменной величиной и определяется использованием того или иного алгоритма обработки данных о Fog-зонах с целью мониторинга и управления вычислительными ресурсами. В результате, каждый из Мини-облаков вычислительной структуры MEC имеет полученную информацию о всех подконтрольных Fog-зонах. Данное решение позволяет разрабатывать и внедрять ряд алгоритмов мониторинга и управления с целью обеспечения эффективности работы вычислительной инфраструктуры в рамках динамичности Fog-устройств, а также миграции микросервисов предоставляемых услуг с помощью, заложенной оркестрации на уровне управления.

3.3.2. Пример микросервисной архитектуры ПО услуги

На данный момент, учитывая всеобъемлющий рост различного рода приложений, платформ (программных продуктов в сети Интернет) и их спроса со стороны пользователей, изменяются и предлагаются новые архитектуры разработки программного обеспечения. Основные причины изменения и сама потребность в

принципиально новых и эффективных архитектурах ПО заключаются в необходимости продукта достаточно в сжатые сроки иметь возможность измениться, предоставить новые функции для пользователей. В современное время изменилась парадигма создания конечных продуктов (для пользователя), которая в первую очередь строится на анализе потребностей пользователя. И если продукт будет не удобен пользователю, соответственно ему не будет места на рынке. В таком ключе, при изменении какой-либо бизнес логики доработка ПО должна быть минимальна и не затрагивать существующие (заложенные) процессы в системе. Кроме того, при сложности (в инженерном смысле слова) программных решений, существуют следующие требования перед ПО современных и перспективных услуг:

- гибкость в настройке;
- быстрая масштабируемость (как отдельных функций, так и в целом всей системы);
- устойчивость;
- переносимость (то есть независимость от среды развертывания и обслуживания);
- безопасность;
- скорость доработки ПО;
- возможность работы несколькими независимыми командами над разработкой/тестированием ПО;
- модульность.

На основе анализа современных архитектурных подходов к построению программного обеспечения, приведенного выше в настоящей диссертации, одним из самых активно развивающихся архитектурных подходов к разработке высоконагруженных приложений является микросервисная архитектура. Такая архитектура призвана решить задачу создания сложных систем путем декомпозиции на независимые модули. Уровень декомпозиции (сервера на модели MVC, с англ.

MVC- Model-View-Controller, модули серверов, классы, объекты, базы данных или даже отдельные функции-методы) определяется в каждом частном случае, проекте командой разработки. Однако каждый из выше определенных модулей может быть представлен в виде общепринятых интерпретируемых объектов: контейнер, виртуальная машина. Например, если декомпозиция является не столь низкоуровневой, то архитектуру программного обеспечения, которое разработано согласно микросервисной архитектуры можно представить в виде следующей схемы (рисунок 3.3.2.1).

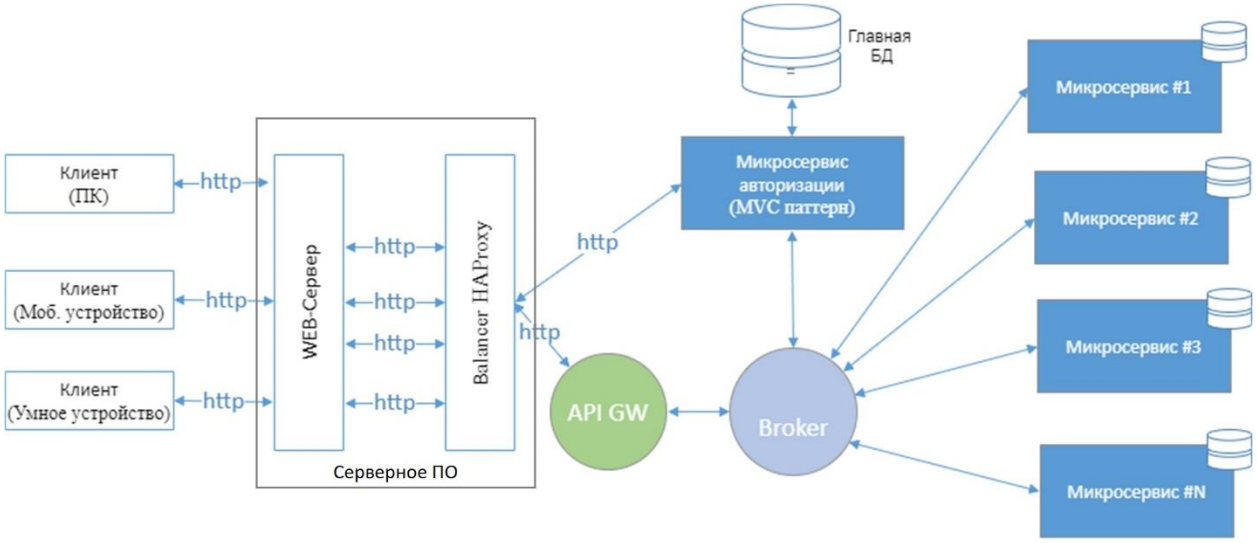

Рисунок 3.3.2.1 – Пример типовой микросервисной архитектуры ПО, высокого уровня декомпозиции

На рисунке 3.3.2.1 – приведены ключевые элементы типовой микросервисной архитектуры серверного программного обеспечения. Приведем краткие описания некоторых подсистем и их основные функции:

- Balancer HAProxy. Это программный модель (выполняет функции прокси-сервера), принимающий пользовательские запросы и выполняющий функции балансировки нагрузки TCP/HTTP, а также распределения запросов между подсистемами (например, программными модулями) на бэкенд-ПО. При этом запросы

могут быть сгенерированы различными пользовательскими устройствами (мобильное приложение, умное устройство, браузер и так далее).

Примечание: модуль «Balancer HAProxy» и WEB-сервер на практике обычно реализуются (для внешних программистов) в виде готового (единого) ПО WEB-сервера, на котором разрабатывается и будет разворачиваться решение. Например – сервер Nginx.

- API GW (с англ.- Application Programming interface Gateway). Данный программный модуль может быть отдельным микросервисом (а может быть программным модулем WEB-сервера, например, Nginx Gateway), который выступает единой точкой приема запросов от внешних пользователей API микросервисов. API GW получает запрос и на основании URL-адреса определяет, какой микросервис должен его получить и обработать. После этого передает сообщение через модуль «Broker» микросервису - получателю, дожидается ответа и возвращает его обратно. Второй важной функцией API GW является создание и ведение пользовательской сессии. Для создания пользовательской сессии, обычно применяется инструмент Токенов, который генерируется для каждого нового пользователя и передается в заголовке внутренних пакетов (либо теле сообщения в специальном поле JSON или XML). Токен позволяет отслеживать все данные сессии.
- MAIN DB – главная База Данных приложения/платформы. Данная БД хранит информацию о пользователях и связанной с ними информацией, а также другие метаданные приложения.
- Brocker – брокер сообщений. Данный модуль в первую очередь необходим для реализации асинхронного взаимодействия между модулями системы. В том числе, позволяет реализовать функции сервиса обнаружения новых микросервисов в системе. Одной из главных функций модуля является в том числе реализация очередей для приема запросов и ответов на полученные запросы. Каждый из микросервисов подключается к данному брокеру сообщений, к соответствующим

очередям. Брокер выполняет также функции маршрутизации сообщений на основании метаданных, передаваемых в заголовках сообщений (в том числе – данные о сессии пользователя, как уже ранее упоминалось – авторизационный токен). Таким образом, каждый экземпляр микросервиса определенного типа подписывается на прослушивание очереди запросов и выполняет подборку сообщений. В данном случае, стоит отметить, что при отправке же запросов в качестве получателя указывается не конкретный экземпляр микросервиса, а тип, к которому он относится. Так как в рамках микросервисного подхода реализация каждого из микросервисов может быть сделана таким образом, чтобы для увеличения производительности микросервиса требовалось только его копирование. Соответственно возможна живая миграция отдельных микросервисов, либо типов микросервисов и их совместная работа при географическом удалении друг от друга. Брокер данных берет на себя также функцию балансировки нагрузки. При этом балансировка производится неявно самими микросервисами, слушающими очередь запросов. В дополнении стоит отметить, что существующие решения в виде открытого кода (с англ. Open source) брокеров имеют дополнительные функции, с помощью которых возможно проводить мониторинг потоков данных проходящих между микросервисами.

Примечание: микросервисный архитектурный подход в каждом случае разработки реализуется по-разному, и брокер не является обязательным элементом. Однако в программном решении, где разрешаются дискретные задачи – построение архитектуры с учетом брокера, позволяет получить решение части задач взаимодействия между микросервисами «под ключ». Например, та же функция – асинхронного взаимодействия программных модулей или обнаружение новых микросервисов.

Стоит также отметить, что при разработке небольших систем, задачи брокера, программного шлюза, «балансировщика» могут разрешить дополнительные модули основного WEB-сервера, на котором разрабатывается и развертывается программный продукт, к примеру, такие возможности предоставляет последняя версия Nginx.

Однако для разрозненных, сложных, высокопроизводительных платформ и решений архитектура строится своя, часто объединяющая другие системы. При всем многообразии архитектурных подходов к реализации серверного программного обеспечения стоит отметить, что отдельные независимые программные модули могут быть реализованы в своем «изолированном» окружении и представлены в виде виртуальных машин или контейнеров. Что открывает возможность использования менее производительных устройств в качестве «вычислительного облака», а именно Fog-устройств. При этом, чем менее требовательный микросервис к вычислительным мощностям (например, уровень декомпозиции программного обеспечения на уровне лямбда-функций), тем менее производительное устройство Туманных вычислений может быть задействовано – вплоть до Умных устройств или нано-компьютеров (если говорить про концепцию Нано-сетей).

3.3.3. Пример использования приведенной структуры/фреймворка взаимодействия туманных и пограничных вычислений

На рисунке 3.3.3.1 приведен пример работы приведенного ранее фреймворка взаимодействия туманных и пограничных вычислений с микросервисной поддержкой.

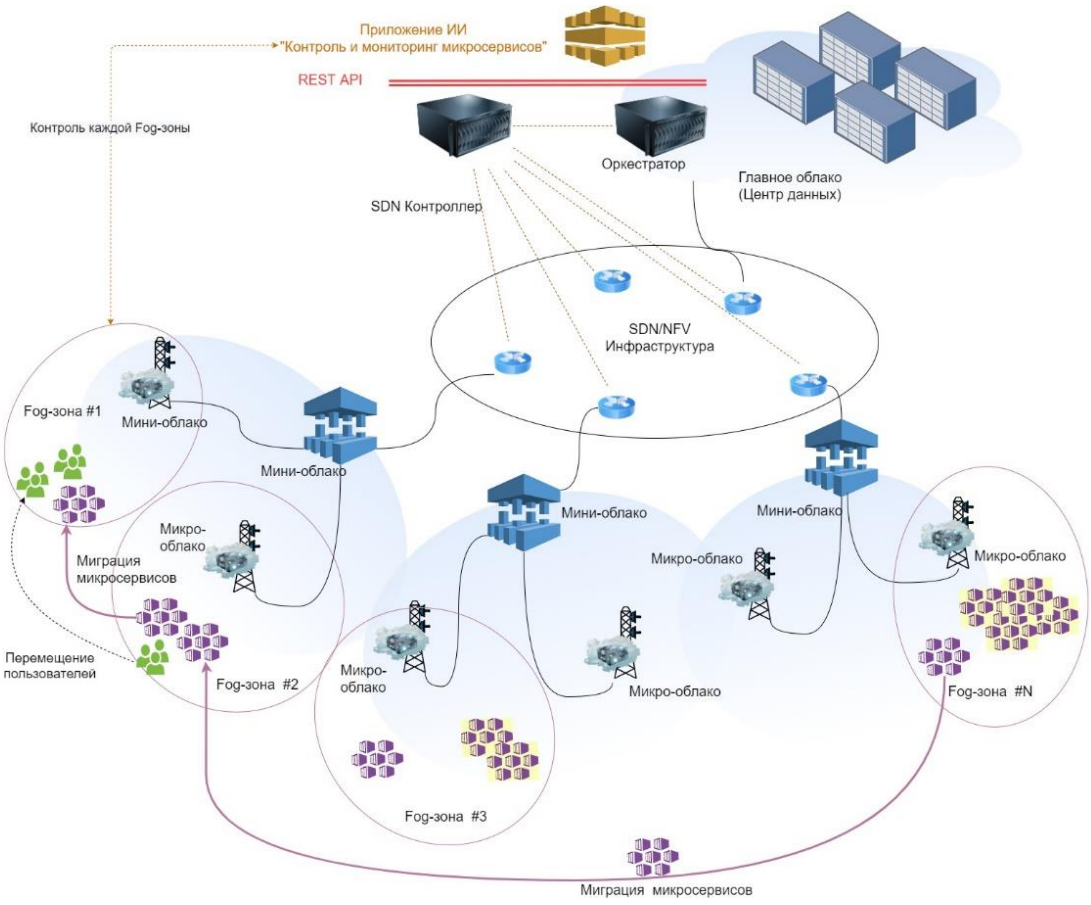

Рисунок 3.3.3.1 – Пример работы фреймворка в виде концептуальной схемы

На рисунке 3.3.3.1 группами сот отражены группы микросервисов. Разные цвета групп микросервисов обозначают различные услуги, которые они реализуют. К примеру, фиолетовые микросервисы – это программное обеспечение обработки фотографий с помощью искусственных нейронных сетей или кадров приложения дополненной реальности. При возникновении увеличения спроса на ту или иную услугу (например, сервис дополненной реальности), или при перемещении пользователей – микросервисы мигрируют в соответствующую Fog-зону, устройства которой способны принять данные микросервисы и предоставить необходимые вычислительные и сетевые возможности. Приведем пример в виде функциональной схемы услуги обработки фотографий пользователя с помощью Искусственных нейронных сетей с целью улучшения качества произведенных снимков на

пользовательские устройства. Функциональная схема приведена на рисунке 3.3.3.2.

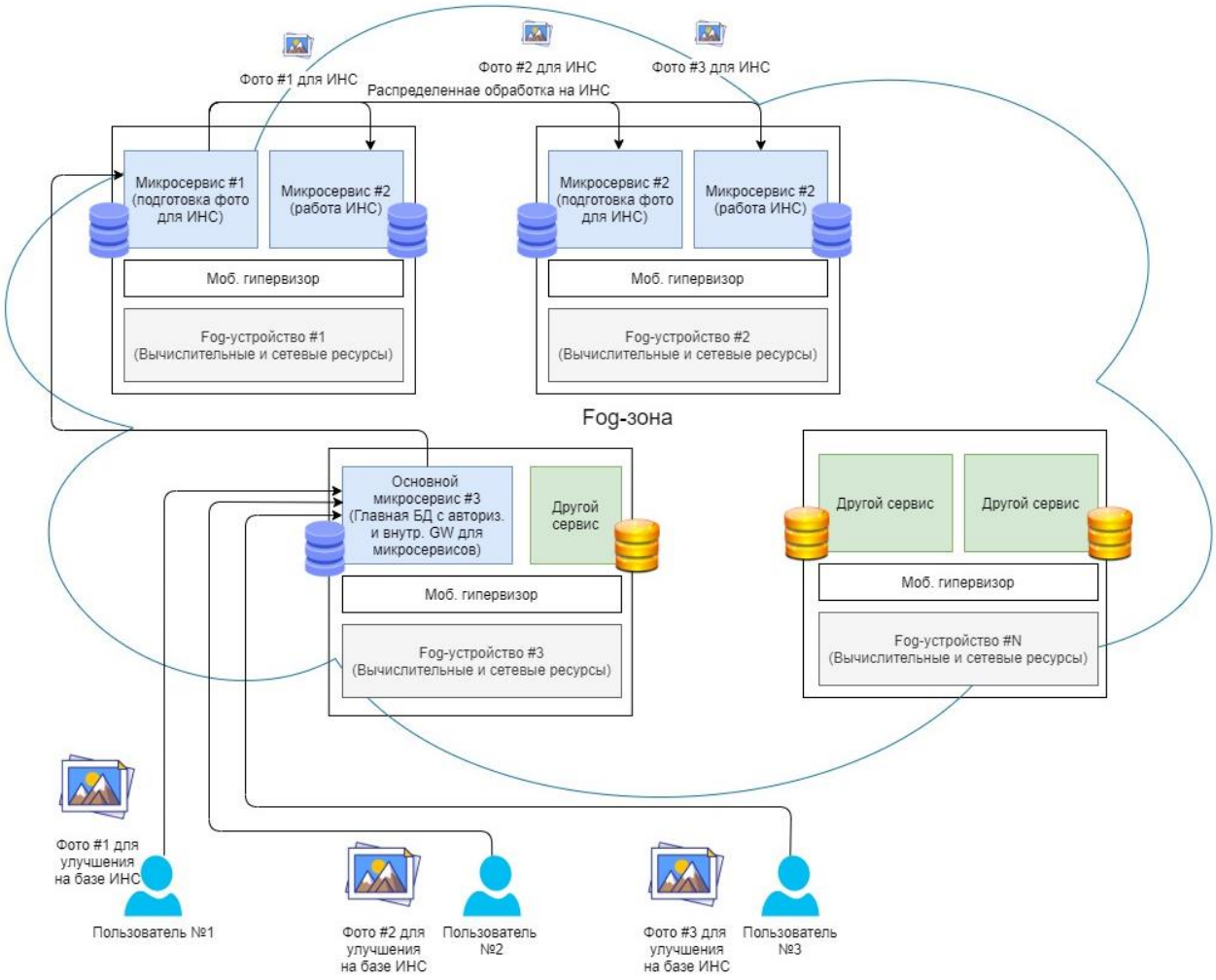

Рисунок 3.3.3.2 – Функциональная схема примера работы микросервисной услуги

На рисунке 3.3.3.2 приведен пример услуги, реализованной на микросервисом подходе, и предоставляющей функции улучшения снимков, сделанных на пользовательские устройства. Подобная услуга реализуется в рамках приложения Google Photos. После того, как пользователь сделал снимок на камеру мобильного устройства, данный снимок отправляется в ближайший микросервис, который направляет снимок на подготовку для обработки искусственной нейронной сетью (например – сверточного типа). Далее в обработанном виде снимок обрабатывается в микросервисе, где реализована ИНС. Данный пример приводит тривиальный сервис,

который может быть развернут в данном фреймворке. Например, более интересное и актуальное направление, если говорить про обработку видео/фото контента – это приложения дополненной реальности (AR – Augmented Reality) и/или виртуальной реальности (VR – Virtual Reality). В данном виде приложений требуется высококачественный контент высокого разрешения. В результате возникают требования по сетевым ресурсам для их передачи, а также вычислительным для их обработки – например, наслоения дополненного контента в реальность. В таких приложениях, в первую очередь необходимо определить маркер, к которому привязан виртуальный контент. И при этом маркером может служить окружающая обстановка – например, вход в музей и так далее. При этом для распознавания объектов на фотографии, для лучшего качества требуются решения машинного зрения на основе искусственных нейронных сетей. В результате можно определить, что для широкого внедрения приложений AR/VR в жизнь людей и ,возможно, в каждый вид деятельности человека требуются достаточно большие сетевые и вычислительные ресурсы. И распределенные туманные динамические вычисления могут помочь реализовать данное направление, предоставив колоссальные вычислительные возможности окружающего мира (суммы окружающих устройств) и сократив нагрузки на сетевую составляющую сетей связи и в тоже время минимизировав сетевую составляющую круговой задержки приложения.

3.3.4. Алгоритм для мониторинга и управления

В данной работе предлагается рассмотреть достаточно сложную, системную задачу по определению центра скопления пользователей (запросов от пользователей) какого-либо сервиса, а также одновременного определения вычислительного потенциала определенной среды туманных вычислений для определения устройства,

на которое будет совершаться миграция микросервиса. Для решения ряда подзадач в рамках вышеописанной задачи предлагается использовать набор эффективных алгоритмов обработки данных. Общий алгоритм, закладываемый в фреймворк, отображен на рисунке 3.3.4.1.

Рисунок 3.3.4.1 – Общий алгоритм фреймворка

В данной диссертации будут рассмотрены части алгоритма, выделенные оранжевым цветом. В рамках общего процесса, происходящего в системе, данные части представляют собой не менее сложные подпроцессы. Стоит отметить, что данный алгоритм действий реализует так называемую пост-аналитику данных. В развитии предлагаемого фреймворка, данный алгоритм планируется усложнить путем расчета значений на прогнозной аналитике данных. Что в свою очередь усложнит работу предлагаемого фреймворка, а с другой стороны, обеспечит актуальность выполнения вычислительных задач в рамках миграции микросервисов и динамики пользователей. Для решения задач, определенных в данном общем алгоритме

действий, предлагается исследовать применение эффективных алгоритмов Искусственного интеллекта. В исследуемых задачах, в рамках данной диссертации, будут рассмотрены алгоритмы кластеризации объектов и алгоритм роевого интеллекта.

3.3.5. Задача определения центра пользователей

Одной из задач, согласно алгоритму фреймворка, изображенного на рисунке 3.3.4.1 является периодическое определение центра скопления пользователей. Для осуществления поиска центра предлагается использовать алгоритм кластеризации K-средних (от англ. K-means). Данный алгоритм характеризуется простотой реализации и быстротой выполнения [36, 37, 50]. Априорными данными являются количество кластеров (задается до начала процесса кластеризации, и его значение обладает высоким влиянием на получение конечного результата). Относительно специфики применения алгоритма в текущей задаче, необходимо отметить, что на количество задаваемых кластеров влияют метрики, собираемые оператором мобильной сети (количество подключений, среднее количество подключений к Базовой станции в день и другие). Определение точной формулы, описывающей зависимость данных метрик с параметром количества кластеров, не является приоритетной задачей в данной исследовательской работе. Радиусы кластеров могут отличаться друг от друга [36], даже если кластера потребителей сервисов были сформированы в одной Fog-зоне в результате выполнения одного и того же процесса с заданным количеством кластеров.

В пределах одной Fog-зоны будут присутствовать области с разной плотностью скопления потребителей сервиса. Стоит отметить, что при этом размеры таких скоплений также не одинаковы. Если в качестве алгоритма кластеризации рассматривать алгоритм формального элемента FORELL с заданным заранее

размером кластера, то есть вероятность, что формируемый кластер не покроет такую область полностью (или зона покрытия будет существенно больше области скопления потребителей сервиса). Поэтому для поиска центра скопления пользователей применяется алгоритм кластеризации K-средних.

Одним из исходных условий в рамках данной диссертационной работы также является статичность устройств ИВ, Умных устройств, смартфонов и других устройств, образующих Fog-зоны. Радиусом кластера будет считаться расстояние между центром и самой удаленной точкой в кластере. В данной работе координаты x, y, z измеряются в метрах. Три плоскости позиционирования были выбраны для учета высоты нахождения пользовательского устройства с целью корректного формирования соответствующих кластеров. Также в реальных условиях координаты пользовательских устройств и Fog-устройств предлагается определять с помощью GPS. GPS приемник определяет координаты (широту и долготу) по расположению относительно друг друга трех спутников и расстояния между ними и приемником (определяется через время передачи радиосигнала от спутников к приемнику).

Работы выбранного алгоритма производится в следующие пять шагов:

1. Задаются входные данные: координаты x, y и z точек. Также задаются количество кластеров и их первоначальные центры.

Присвоение пользователей (пользовательских устройств) к ближайшим центрам кластеров. Для этого используется Евклидова метрика расстояния между точками в пространстве:$\sqrt{(x-x_k)^2+(y-y_k)^2+(z-z_k)^2}$, где x, y, z - координаты самой точки.

2. Определение центров масс: $C_{mx}=\frac{\sum_{i=0}^{l} x_i}{l}$, $C_{my}=\frac{\sum_{i=0}^{l} y_i}{l}$, $C_{mz}=\frac{\sum_{i=0}^{l} z_i}{l}$, где l - количество Fog-устройств, входящих в кластер.

3. Сравнение центров масс и предполагаемых центров кластеров.

4. Если центры масс и предполагаемые центры кластеров равны, то центры кластеров считаются окончательно определенными и все приписанные к ним

пользователи помечаются как элементы данного кластера. Если они не равны, то шаги 2-4 повторяются заново, но с предполагаемыми центрами кластеров равными центрам масс, определенным на данной итерации.

Стоит отметить, что в данной диссертационной работе (на текущем этапе проекта) при реализации математической модели алгоритма не учитывалась подвижность пользователей сервисов и Fog-устройств.

3.3.6. Задача определения устройства туманных вычислений для последующей живой миграции микросервисов

Второй исследуемой задачей в рамках данной диссертационной работы является задача определения тех Fog-устройств, которые имеют свободные необходимые вычислительные возможности и удовлетворяют условиям миграции на них микросервиса(-ов) с последующим развертыванием и включением в общую архитектуру. В рамках данного подпроцесса фреймворка необходимо решить задачу оптимизации: для заданной функции, описывающей состояние Fog-устройств через набор параметров среди всех возможных значений этой функции необходимо найти такое значение, при котором данная функция принимает максимальные (max) значения (min значения отбрасываются).

Существует множество алгоритмов, которые гарантированно находят экстремум функции, который является локальным минимумом вблизи заданной начальной точки x_0. К таким алгоритмам относятся, например, алгоритмы градиентного спуска. Однако, в данной задаче нужно найти глобальный максимум функции, которая в заданном диапазоне изменения параметров помимо одного глобального экстремума имеет множество локальных экстремумов. Градиентные алгоритмы не могут справиться с оптимизацией такой функции, потому что их решение сойдется к ближайшему экстремуму около начальной точки [33, 29]. Для

задач нахождения глобального максимума или минимума используют так называемые алгоритмы глобальной оптимизации. К одному из таких алгоритмов относятся алгоритмы роевого интеллекта – PSO (на англ. Practical Swarm Optimization). Алгоритм роевой оптимизации PSO был разработан на основании принципа поведения биологических организмов, в частности, способности групп некоторых видов животных работать как единое целое для определения желаемых позиций в заданной области, например, как птицы «стекаются» к источнику пищи.

Таким образом, данный алгоритм определяется следующим образом:

Каждая отдельная частица *i* состоит из трех векторов: её положение в D-мерном пространстве поиска $\overline{x_i} = (x_{i1}, x_{i2}, \dots, x_{iD})$, лучшая найденная позиция, $\overline{p_i} = (p_{i1}, p_{i2}, \dots, p_{iD})$, направленная скорость движения $\overline{v_i} = (v_{i1}, v_{i2}, \dots, v_{iD})$. При запуске алгоритма, частицы равномерно, случайным образом инициализируются по всему пространству поиска, при этом скорость частиц также инициализируется случайным образом. Сформированные частицы перемещаются по пространству поиска с помощью довольно простого набора уравнений обновления векторов частицы. Алгоритм обновляет весь рой на каждом временном шаге, обновляя скорость и положение каждой частицы в каждом измерении по следующим правилам:

$$v_{id} = v_{id} + c\varepsilon_1(p_{id} - x_{id}) + c\varepsilon_2\left(p_{gd} - x_{id}\right); \quad (3.3.6.1)$$

$$x_{id} = x_{id} + v_{id}; \quad (3.3.6.2)$$

где c – постоянная ускорения. ε_1 и ε_2 – случайные числа в пределах [0; 1]; p_{id} – лучшее положение из пройденных всеми частицами; p_{gd} – это положение, найденное любой соседней частицей. Процесс обновления кратко описан в алгоритме, представленном в виде таблицы 3.3.6.1 в формате псевдокода.

Таблица 3.3.6.1

Алгоритм обновления в PSO

for каждого шага *t* **do**
for каждой частицы *i* в рое **do**
обновить позицию *xt* используя выражения (3.3.6.1) и (3.3.6.2)
рассчитать фитнес-функцию для *xt* f (xt)
обновить *pi , pg*
end for
end for

Стоит отметить, что в алгоритме скорость частиц фиксируется на максимальном значении *vmax*. Без фиксации, алгоритм склонен не сойтись, где расчет значений (3.3.6.1) и (3.3.6.2) приводил бы к быстрому увеличению скорости и, следовательно, положения частиц, приближающиеся к бесконечности. Параметр *vmax* не позволяет системе войти в данное состояние, ограничивая скорость всех частиц.

В результате необходимо определить параметры, описывающие каждое из исследуемых Fog-устройств. При этом некоторые параметры оцениваются на уровне сравнения с предельными значениями, например, количество выделенной логической ОЗУ, необходимой для работы готовящегося к миграции микросервиса.

Для решения текущей задачи в настоящей диссертационной работе были определены параметры, описывающие Fog-узел с точки зрения обеспечения качества обслуживания. Глобально задача сформулирована следующим образом: необходимо поддерживать время обслуживания за счет выбора Fog-узла, на который необходимо мигрировать микросервис. Таким образом, минимизированная фитнес-функция выглядит следующим образом:

$$T = \sum_{i=1}^{n} W_i \cdot TimeSlot_i$$

, где:

- T – рассчитываемый параметр в [мс],
- W_i- вес соответствующего параметра,

- $TimeSlot_i$ – параметры, описывающие состояние Fog-устройства,
- n-количество таких параметров.

В данной диссертационной работе используются два параметра:

- $TimeSlot_1$ (задержка распространения),
- $TimeSlot_2$ (время обработки запроса микросервисом).

Так как оба параметра имеют одинаковое влияние на оцениваемый параметр, вес каждого из параметров (W_i) равен 0.5, при этом сумма весов не должна превышать 1. Данные параметры рассчитываются в [мс], таким образом, нет необходимости их приводить в одну область значений, как это требуется при оценке параметров с различных областей определения.

В результате, в рамках данной работы фитнес-функция выглядит следующим образом:

$$T = \sum_{i=1}^{2} W_i \cdot TimeSlot_i = 0.5 \cdot TimeSlot_1 + 0.5 \cdot TimeSlot_2$$

Первый параметр определяется через тайм-трекер на уровне фреймворка. А второй параметр отправляется устройством – время обработки задачи.

3.3.7. Результаты моделирования

Моделирование работы алгоритма поиска центра пользователей.

Для моделирования предложенного алгоритма, описанного в п.3.3.5, была разработана программная модель на основе языка программирования Python и соответствующих математических библиотек. На первой стадии работы, программная модель генерировала данные о пользовательских устройствах.

Следующие исходные данные были использованы для генерирования пользовательских данных:

1. Количество генерируемых скоплений пользователей не более 5-ти;

2. Предполагаемые центры были распределены случайным образом в соответствии с принципом - вероятнее возможных мест в пространстве скопления людей.

3. Таким образом, были сгенерированы координаты устройств исследуемых регионов (кластеров). Где в кластере №1 было сгенерировано 10 пользовательских устройств, во 2-ом кластере – 50, 3- ем 15, в 4-ом – 30, а в 5-ом – 50 пользовательских устройств.

Для наглядности предложенной модели, были также сгенерированы Fog-узлы. Полученные кластеры показаны на рисунках 3.3.7.1.1 и 3.3.7.1.2. Где на рисунке 3.3.7.1.1 выделены соответствующие кластеры и их номера.

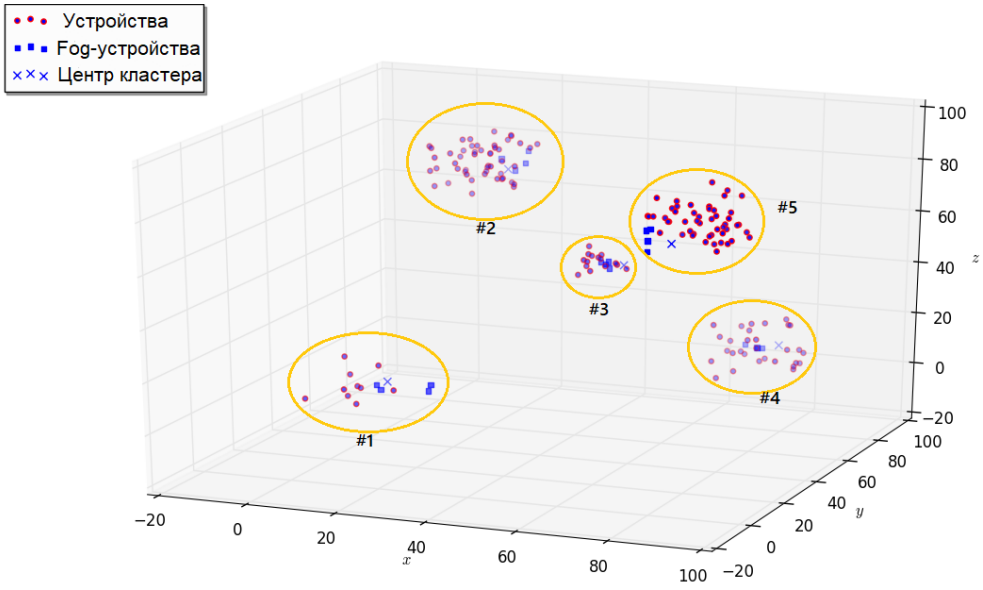

Рисунок 3.3.7.1.1 – Сгенерированные данные об устройствах (вид №1)

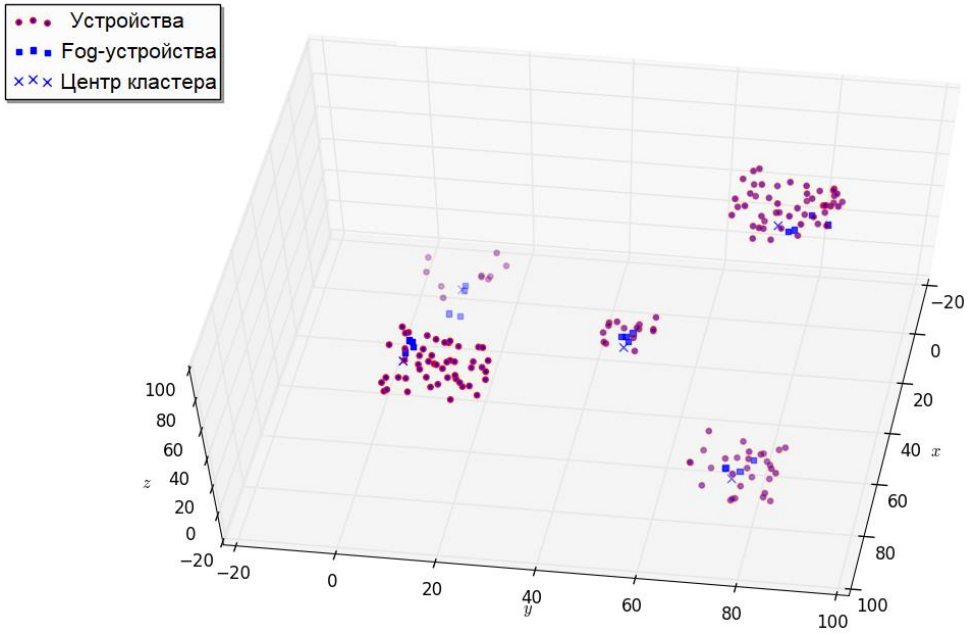

Рисунок 3.3.7.1.2 – Сгенерированные данные об устройствах (вид №2)

Опишем получившиеся данные: с точки зрения практической модели (реального мира), кластеры №1 и №4 отражают скопление пользователей в кафе быстрого питания на первых этажах здания, а также пользователей на следующих этажах офисного здания. Кластеры под номерами 2, 3 и 5 также отображают скопление пользователей на верхних этажах офисного здания, где, например, есть открытые рабочие пространства, что на практике означает более высокую плотность людей на квадратный метр. Либо торговые площади торгового центра/комплекса.

На стадии моделирования алгоритма K-средних, согласно теории, были также определены начальные центра пользователей (то есть возможные центры – стохастические точки возможного скопления пользователей). На следующей стадии моделирования алгоритм K-средних определяет центра масс (центр скоплений пользователей), а также радиус их разброса относительно их местоположения.

Результат работы алгоритма, математическая суть которого была определена ранее, представлен на рисунках 3.3.7.2.1, 3.3.7.2.2.

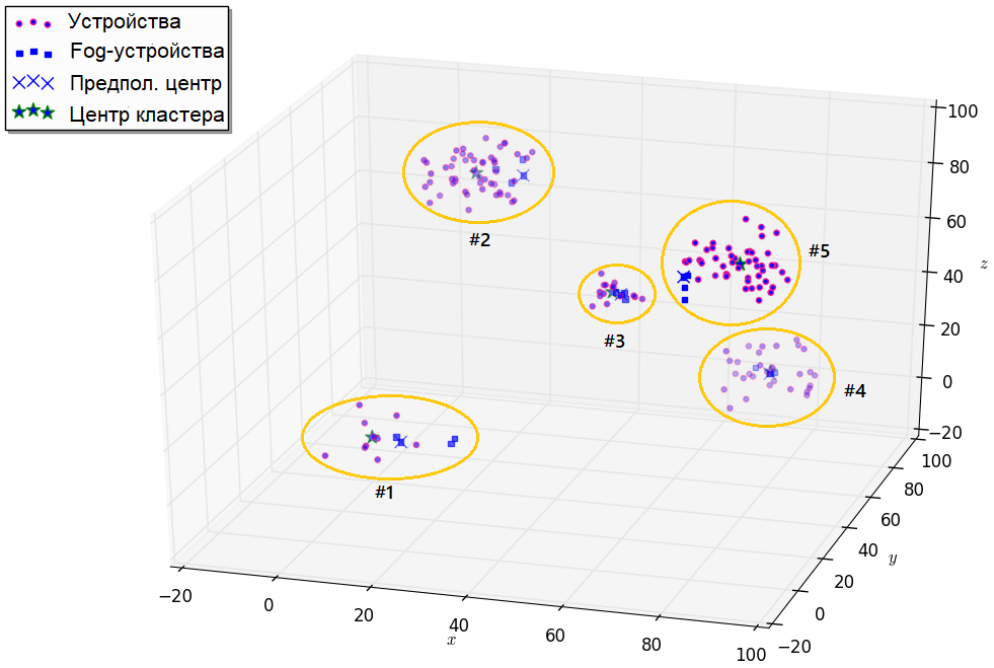

Рисунок 3.3.7.2.1 – Результат работы алгоритма К-средних (вид данных №1)

Центры кластеров, которые были рассчитаны с помощью алгоритма К-средних, обозначены на рисунках 3.3.7.2.1 и 3.3.7.2.2 в виде зеленых звездочек. Устройства пользователей в виде фиолетовых точек, а предполагаемые центры кластеров, которые были определены стохастическим образом при старте алгоритма – синими крестиками.

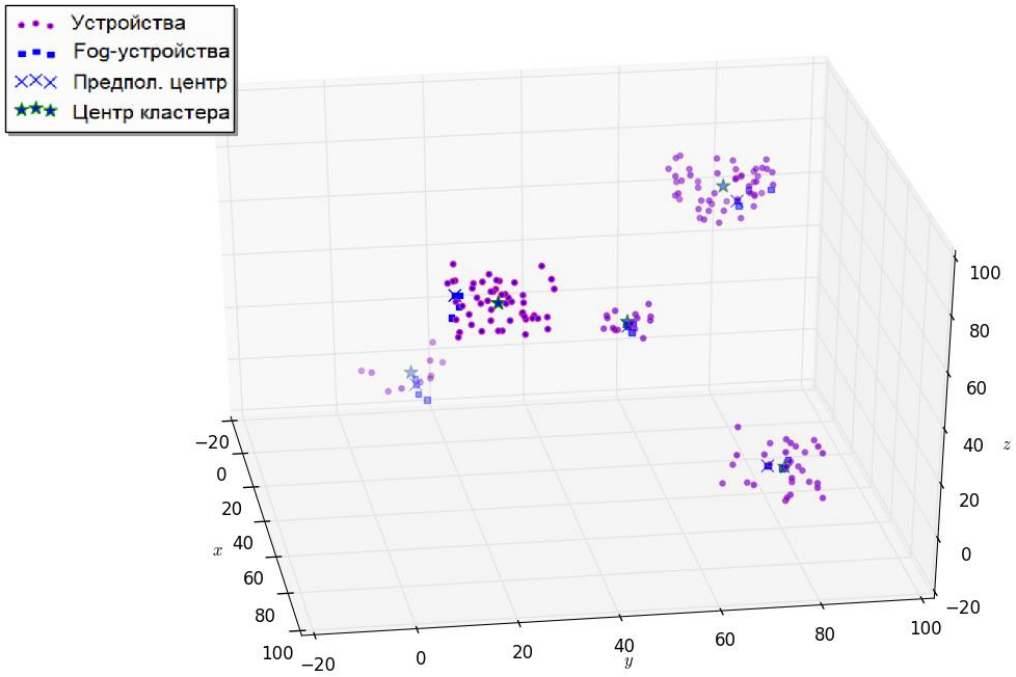

Рисунок 3.3.7.2.2 – Результат работы алгоритма К-средних (вид данных №2)

В результате моделирования были определены центры кластеров (рисунки 3.3.7.2.1, 3.3.7.2.2), а также их радиусы. Центры отображены в виде целеных звездочек. Данные о рассчитанных радиусах кластеров приведены на гистограмме (рисунок 3.3.7.3).

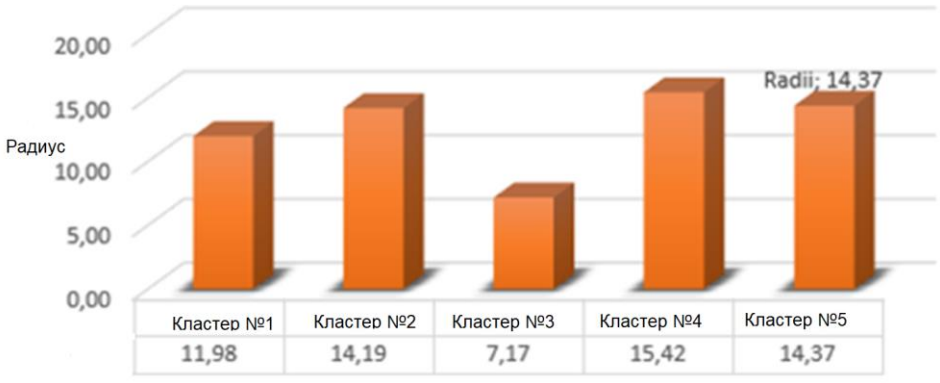

Рисунок 3.3.7.3 – Радиусы кластеров

На основе проведенных расчетов можем определить, что самым небольшим по размеру кластером является кластер №3, где находится 15 пользовательских

устройств (то есть самих пользователей), а самым большим – кластер №4, где находится 30 устройств. В данном случае, стоит заметить, что относительно плотности самих устройств на м^2 пространства, кластер №5 самый плотный на количество пользовательских устройств и при этом он находится достаточно близко к 4-му кластеру с самым большим относительно расстояния разбросом пользователей.

Моделирование второго алгоритма по определению Fog-устройства для последующей миграции микросервиса(-ов).

Для моделирования алгоритма роевой оптимизации (PSO) была разработана программная модель на языке программирования Python с соответствующими программными библиотеками и фреймворками, такими как: NumPy, Pandas, Matplotlib, Math и другие.

Исходные данные для генерирования информации о пользовательских устройствах были взяты из первой программной модели.

Согласно заданной фитнесс-функции и описанному алгоритму PSO были получены результаты для 5-го кластера, которые представлены на рисунке 3.3.7.4. В 5-ом кластере 50 устройств, каждое их которых характеризуется набором параметров, в том числе временных, которые были сгенерированы в следующих пределах:

$$TimeSlot_1 \in [0.5, 10]\ ms, \qquad TimeSlot_2 \in [0.2, 2]\ ms$$

Результат работы алгоритма PSO, показал, что 13-е Fog-устройство с показателями $TimeSlot_1 = 1.18\ [ms]$, $TimeSlot = 0.76\ [ms]$ и значением фитнесс-функции 0.97, имеет наименьший искомый показатель для дальнейшего размещения микросервиса на нем, а также удовлетворяет необходимым вычислительным и сетевым потенциалом.

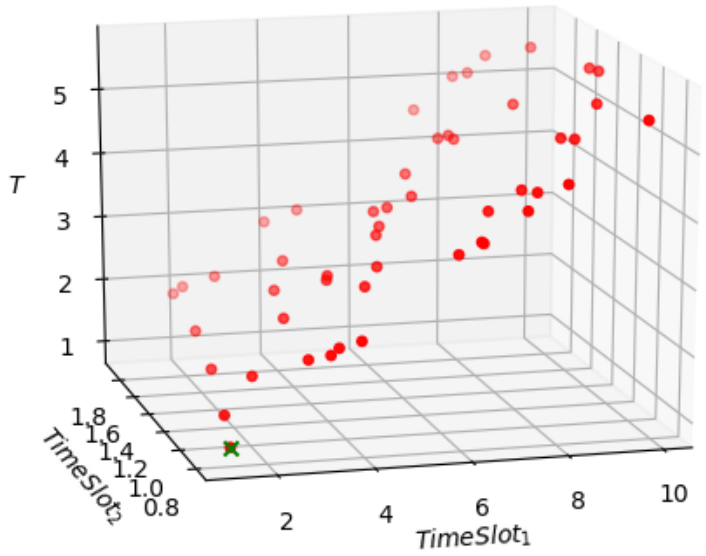

Рисунок 3.3.7.4 – визуализация данных расчета фитнесс-функции и результата моделирования алгоритма роевого интеллекта – PSO

Показатели данного устройства в виде точки отмечены зеленым крестиком на рисунке 3.3.7.4. Также данный график визуализирует данные, рассчитанные для каждого Fog-устройства тумана (сгенерированные данные в соответствующих пределах и соответствующее значение фитнесс-функции).

Как дополнение к полученным результатам, стоит отметить, что на основе значений фитнесс-функции, существует возможность разделения множества Fog-устройств одной Fog-зоны на микро-кластера. Каждый микро-кластер может удовлетворять требуемым показателям качества предоставляющих услуг. Пример такого разделения, для наглядности, представлен на рисунке 3.3.7.5 – где каждый из «микрокластеров» предоставляет необходимый уровень варьируемого качества в соответствующих пределах соответствующей услуги. Микрокластера отображены в виде бежевых овалов на рисунке 3.3.7.5.

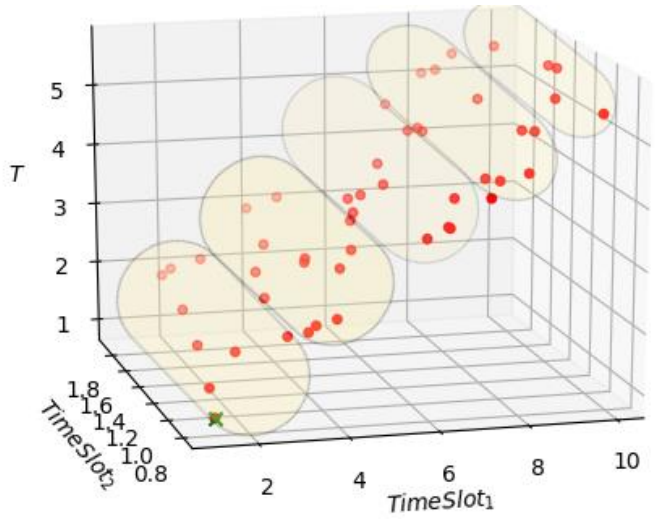

Рисунок 3.3.7.5 Разделение зоны Туманных вычислений на микрокластера услуг

Данное разделение является виртуальным для систем мониторинга и управления в рамках предлагаемого фреймворка с целью оперативной работы алгоритмов по выбору Fog-устройств в Fog-зоне для последующей миграции микросервисов с сохранением качества предоставляемой услуги.

Оценка эффекта применения алгоритма выбора Fog-устройства, относительно времени выполнения функции микросервисом.

Для оценки эффекта применения роевого алгоритма проведем следующие расчеты.

В пункте 3.3.6 приводятся параметры и описывается алгоритм для рационального выбора устройства Туманных вычислений с целью последующей миграции микросервиса. В качестве параметров принимаются следующие:

- $TimeSlot_1$ (задержка распространения);
- $TimeSlot_2$ (время обработки запроса микросервисом).

В результате моделирования было определено, что 13-е устройство имеет минимальные значения данных параметров: $TimeSlot_1 = 1.18\,[ms]$, $TimeSlot_2 = 0.76\,[ms]$, где время выполнения функции можем оценить суммой данных параметров.

$$T_{\text{функц}} = TimeSlot_1 + TmeSlot_2;$$

Таким образом, минимальное время выполнения функции на определенном Fog-устройстве:

$$T_{\text{функц.мин}} = 1.18 + 0.76 = 1.94 \cong 2[\text{мс}];$$

Если же осуществлять выбор Fog-устройства равновероятно из всего набора устройств (в рассматриваемой Fog-зоне - 50 устройств), то время выполнения функции может быть оценено средним значением $T_{\text{фукц.сред.}}$. Рассчитаем данный параметр по нижеследующей формуле:

$$T_{\text{фукц.сред.}} = \frac{1}{N}\sum_{i=1}^{N} TimeSlot_{1_i} + TimeSlot_{2_i} = \frac{1}{N}\sum_{i=1}^{N} T_{\text{функц}_i}$$

Учитывая сгенерированные данные в модели выше, среднее время выполнения функции микросервисом при равновероятном распределении функции на устройства Fog-зоны составляет $T_{\text{фукц.сред.}} = 6.43[\text{мс}]$.

Произведем точечную оценку эффекта применения алгоритма выбора Fog-устройства относительно времени выполнения функции микросервисом через отношение полученного минимального времени выполнения функции микросервисом $T_{\text{функц.мин}}$ к среднему значению выполнения, в случае равновероятного выбора устройства $T_{\text{фукц.сред.}}$

$$\frac{\left(T_{\text{функц.мин}} \cdot 100\%\right)}{T_{\text{фукц.сред.}}} = 30.90\%$$

Таким образом, можно сделать вывод, что применение алгоритма роевого интеллекта (PSO) в предложенном фреймворке позволяет уменьшить время выполнения функции микросервиса за счет рационального распределения ресурсов на величину до 70%.

3.4. Выводы по Главе 3

1. В данной главе приводится исследование причин и обоснование необходимости перехода к децентрализации облачных вычислений как части вычислительно-сетевой инфраструктуры сетей связи пятого и последующего поколения. Децентрализация вычислений является одним из направлений решения задачи по обеспечению требуемых характеристик сетей связи с ультрамалыми задержками при высокой плотности устройств. Данное направление преследует цель минимизации возможной вносимой сетевой составляющей задержки путем «выноса» серверных частей сервисов на «край» сети. В качестве технологий здесь рассматриваются граничные вычисления с множественным доступом (MEC) и туманные вычисления (Fog).

2. Одним из основных вопросов, которые были подняты в данной части диссертационной работы является вопрос современных архитектур программного обеспечения. Приводится необходимость разработки и проработки архитектуры на ранних стадиях создания программных продуктов, реализующих услуги в сетях связи. На основе анализа и современного тренда, приводится микросервисная архитектура программного обеспечения как одна из самых актуальных и реализующих необходимые требования к современному ПО. Как результат, предлагается использовать данный архитектурный подход при реализации современных и перспективных услуг сетей связи, особенно которые являются требовательными к критериям качества обслуживания и принадлежат к классу сетей связи с ультрамалыми задержками и сверхнадежностью.

3. Как результат анализа современных и перспективных технологий построения сетей связи пятого и последующего поколения (SDN/NFV и MEC/Fog) в данной главе предлагается решение в виде структуры/фреймворка взаимодействия распределенных вычислений с поддержкой микросервисов как модулей услуг, реализуемых сетью связи. Данный фреймворк учитывает микросервисное взаимодействие и нацелен на сокращения расстояния между пользователем и услугой до минимума путем размещения функций услуг в виде микросервисов на самых

«маленьких» Fog-устройствах с точки зрения сетевых и вычислительных способностей, к примеру – микроконтроллере устройства Интернет Вещей, либо смартфоне пользователя. Кроме архитектуры предлагаемого фреймворка приводится пример услуги и соответствующая функциональная схема с описанием сущностей и процессов взаимодействия.

4. В рамках данной главы и предложенного фреймворка взаимодействия распределенных туманных вычислений с микросервисной поддержкой рассматриваются два алгоритма, решающие две задачи в рамках данной диссертации, а именно: определение центра скопления пользователей, а также определение Fog-устройства, которое будет выбрано для живой миграции микросервиса услуги с целью сохранения качества обслуживания. В задачах были использованы алгоритмы К-средних и Роевого Интеллекта, которые в свою очередь входят в класс алгоритмов, относящихся к Искусственному Интеллекту.

5. Как результат, были приведены результаты моделирования данных алгоритмов и описанных моделей, где для моделирования был использован язык программирования Python с соответствующими библиотеками для работы с данными, типа Pandas, NumPy и другие. Полученные данные показывают работоспособность предложенных алгоритмов с точки зрения предложенного фреймворка и перспективность развития данного решения.

6. Было также определено, что применение алгоритма роевого интеллекта в предлагаемом фреймворке с целью рационального выбора Fog-устройства позволяет уменьшить время выполнения функции микросервиса на величину до 70% относительно равновероятного распределения ресурсов.

ГЛАВА 4. МЕТОД ПРОГНОЗИРОВАНИЯ НАГРУЗКИ НА КОНТРОЛЛЕРЫ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

4.1. Проблема мониторинга контроллеров Программно-конфигурируемой сети

С целю обеспечения достаточно высоких требований по качеству предоставляемых услуг, в том числе в таком классе сетей как сети связи с ультрамалыми задержками и сверхнадежностью, необходимо обеспечить устойчивость систем управления в сетях связи 5G/IMT-2020 и последующего поколения. Как уже ранее в Главе 1 настоящей диссертации было приведено, основными сетевыми технологиями опорных сетей, согласно рекомендациям Международного Союза Электросвязи, являются технологии Программно-конфигурируемых сетей (SDN) и виртуализации сетевых функций (NFV). В SDN управляющим главным функциональным элементом является контроллер, реализующий логику и протоколы взаимодействия. В том числе такие протоколы, как OF-Config и OpenFlow для управления SDN-коммутаторами. В системах виртуализации сетевых функций таким элементом является оркестратор NFV. В рамках данной диссертации будет особенно исследован вопрос мониторинга и прогнозирования нагрузки на контроллеры Программно-конфигурируемых сетей.

За время развития концепции программно-конфигурируемых сетей было выполнено много исследовательских работ (в том числе коммерческими компаниями, разрабатывающими решения SDN, либо занимающиеся их внедрением) по различному тестированию решений, в частности – стрессовые тесты. Стоит отметить, что в большинстве случаев целю исследований в такого рода работах является определение предела работы контроллера SDN по параллельно обслуживаемым

потокам OpenFlow, оценка преимущества одной программной архитектуры контроллера перед другой, а также нахождение самого быстродействующего контроллера, имеющего более широкий функционал и так далее. Таким образом знание о нагрузках контроллера Программно-конфигурируемой сети позволяет оценить работоспособность сети в целом.

Учитывая данный факт, прогнозирование поведения контроллера программно-конфигурируемой сети является одной из первостепенных задач. В данном направлении исследования одним из подходов является разработка специального программного обеспечения под операционную систему, на которой развернут контроллер SDN. К примеру – Linux Debian/Ubuntu и так далее. Данное ПО обращается к операционной системе через вызовы системных функций (регламентированы в рамках соответствующей ОС и ее версии) и запрашивает значения параметров аппаратной части (потоки в процессоре и их нагрузка, используемая оперативная память, используемая постоянная память, активность ядер процессора и так далее). Полученные данные возможно первично обработать и передать далее в аналитическую систему/платформу по сетевому API, которая может быть расположена удаленно, относительно самой ОС с развернутым контроллером Программно-конфигурируемой сети. Однако, в данном случае стоит отметить ряд недостатков такой архитектуры решения:

- зависимость от аппаратной части контроллера SDN;
- зависимость от типа и версии ОС, так называемая «вендорская игла» от производителя ПО и зависимых библиотек и фреймворков;
- зависимость от обновлений системных утилит, которые достаточно часто обновляются разработчиками ОС с целью обеспечения безопасности, устойчивости и так далее;
- внесение дополнительных процессов для процессора АПК контроллера SDN;

- сложность в переносимости на другие решения и быстрого развертывания данного мониторингового ПО;
- в случае стороннего ПО (типа Zabbix и так далее) существует зависимость от его функций в первую очередь, а во-вторых, данное ПО изначально не учитывает специфику работы контроллера SDN. А также, учитывая вектор развития технологий ИИ в системах управления сетями связи, мониторинговое ПО должно выполнять и требования к модулям такого класса служебных приложений сетей связи и, в частности, SDN;
- Стоит также отметить, повышение вероятности вывода контроллера Программно-конфигурируемой сети из строя при некорректной работе данного мониторингового ПО, в случае его размещения на одном уровне с ОС контроллера.

В результате, учитывая вышеприведенные недостатки такого метода организации мониторинга контроллера Программно-конфигурируемой сети, существует задача по разработке нового метода по анализу и прогнозированию нагрузки на контроллеры программно-конфигурируемой сети. В данной диссертационной работе предлагается новый метод мониторинга и прогнозирования нагрузки на контроллер SDN, основанный на проведении аналитики метаданных только суммы служебных потоков, поступающих от коммутаторов OpenFlow на контроллер Программно-конфигурируемой сети. Такой метод позволит разрешить выше озвученные проблемы/недостатки и в том числе проблему зависимости от аппаратной части и операционной системы, на которых развернут контроллер программно-конфигурируемой сети.

Учитывая особенности получаемых статистических данных протокола OpenFlow, в данной диссертационной работе был произведен анализ существующих математических методов по прогнозированию данных, также был проведен многопараметрический корреляционный анализ, результат которого показывает наличие зависимости активности служебных потоков протокола OpenFlow и нагрузки аппаратной части контроллера SDN. В процессе данной работы был также разработан

дополнительный программный модуль к серверу мониторинга (Глава 2 настоящей диссертации) и предиктивного анализа служебных потоков в программно-конфигурируемой сети, на основе языка программирования Python и соответствующих библиотек обработки данных типа Pandas, NamPy и библиотек Искусственных Нейронных Сетей, типа Tensorflow. Стоит отметить, что полученные результаты подтверждают реализуемость предложенного метода прогнозирования нагрузки контроллера SDN в сетях связи 5G/IMT-2020 и следующих поколений сетей связи 2030.

4.2. Разработка метода прогнозирования нагрузки на контроллеры программно-конфигурируемых сетей на основе технологий Искусственного Интеллекта

4.2.1. Описание метода и исследуемая модель

К текущему моменту развития технологий Программно-конфигурируемых сетей существует немало работ, направленных на исследование и разработку различных методов тестирования контроллера, в том числе так называемое стрессовое тестирование. Как уже было выше отмечено, в данной диссертационной работе рассматривается возможность реализации мониторинга нагрузки контроллера SDN при помощи мониторинга и интеллектуальной аналитики метаданных только группы служебных потоков OpenFlow. Для проверки работоспособности предложенного подхода использовалась модельная программно-конфигурируемая сеть, архитектура которой приводилась в Главе 1, с работающим сервисом «Video-on-Demand» на уровне передачи данных сети SDN.

Аналитическая система (далее – служебное приложение сети), частью которой является модуль прогнозирования активности служебных потоков OpenFlow, разработана на языке программирования Python в виде WEB-сервера, работающего на основе модели проектирования MVC (Model-View-Controller). Данное ПО работает поверх северного программного интерфейса API контроллера SDN модельной сети лаборатории.

В качестве контроллера сети SDN, в данной модельной сети используется OpenDaylight Beryllium SR4. Уровень передачи данных построен на основе коммутаторов Mikrotik с поддержкой протокола OpenFlow версии 1.0. Общая архитектура сегмента отображена на рисунке 4.2.1.1. На данном рисунке также кроме главных элементов модельной сети отображены объекты исследования и их параметры (метаданные на основе структуры OpenFlow – таблиц).

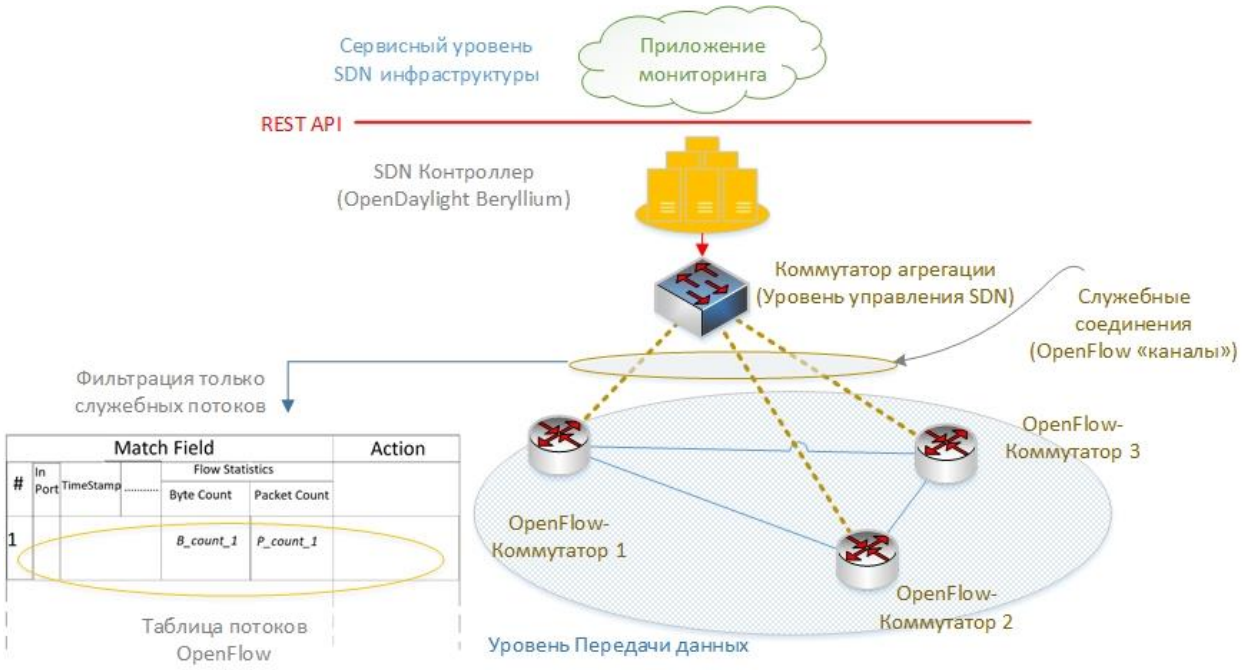

Рисунок 4.2.1.1 – Архитектура сегмента с объектами исследования

Для формирования исследуемых наборов данных (с англ. – Data Sets), приложение мониторинга отправляло REST-запросы каждую секунду на контроллер

программно-конфигурируемой сети через северный API. Следующей стадией была дополнительная фильтрация полученных таблиц потоков таким образом, чтобы далее на обработку отправлялись данные только по служебным потокам OpenFlow. Далее обработанный набор данных проходил необходимую дополнительную обработку для формирования наборов данных с целью построения аналитических моделей, описание которых приведено далее в настоящей диссертационной работе.

Ранее (Глава 2) была подмечена возможность составления метаданных потоков на основе двух глобальных частей таблицы потоков SDN-коммутатора, а именно: Match Field и Actions (см. рисунок 4.2.1.1). Одной из важных особенностей этих данных является то, что на основе счетчиков «Byte Count» и «Packet Count» нельзя точно определить точную длину пакета в потоке. Так как за один момент времени счетчики могут быть равны: «Byte Count» -1500, «Packet Count» - 3. Соответственно, на основе этих данных нельзя точно определить длину каждого из пакетов, зарегистрированных в потоке за промежуток времени: $\Delta T = 1$ [c].

Стоит также отметить, что счетчики отображают суммарное значение параметров [Byte Count», «Packet Count»]. Однако кроме данных счетчиков, в таблице потоков SDN-коммутаторов существует еще один параметр «Time Stamp», который позволяет оценить в каждый момент времени мгновенное значение [ByteCount_delta] и [PacketCount_delta]. Таким образом, за произвольный период времени ΔT, имея отсчеты значений [Byte Count], [Packet Count], [TimeStamp] возможно составить набор данных с установленной структурой данных, где каждый отсчет отображает мгновенное значение [ByteCount_delta] и [PacketCount_delta]. Отсчеты значений [Byte Count], [Packet Count], [TimeStamp] – формируются путем ежесекундных запросов на контроллер сети SDN через программный интерфейс REST API. Структура формируемого на основании запросов $\mathrm{DataSet_{RQ}}$ с «сырыми» данными (4.2.1.1) и формула (4.2.1.2) его преобразования в требуемый формат $\mathrm{DataSet_{ML}}$ с мгновенными значениями (4.2.1.3) приведены ниже.

Пусть, PacketCount_delta – $\mathrm{PC_{delta}}$,

ByteCount_delta – BC_{delta}, а

TimeStamp_deltas – TS = 1 [sec.] = const, тогда:

$$DataSet_{RQ} = \begin{matrix} [TimeStamp] & [ByteCount] & [PacketCount] \\ TimeStamp_{11} & Byte\_Count_{12} & Packet\_Count_{13} \\ TimeStamp_{21} & Byte\_Count_{22} & Packet\_Count_{23} \\ \dots & \dots & \dots \\ TimeStamp_{N1} & Byte\_Count_{N2} & Packet\_Count_{N3} \end{matrix} \quad (4.2.1.1);$$

$$\begin{cases} BC\_delta_{N2} = Byte\_Count_{N2} - Byte\_Count_{(N-1)2}, & \text{if } N \geq 1 \\ PC\_delta_{N2} = Packet\_Count_{N2} - Packet\_Count_{(N-1)2}, & \text{if } N \geq 1 \end{cases} \quad (4.2.1.2);$$

$$DataSet_{ML} = \begin{matrix} [TimeStamp] & [ByteCount] & [PacketCount] \\ TS & BC\_delta_{12} & PC\_delta_{13} \\ TS & BC\_delta_{22} & PC\_delta_{23} \\ \dots & \dots & \dots \\ TS & BC\_delta_{N2} & PC\_delta_{N3} \end{matrix} \quad (4.2.1.3);$$

При этом, расчет суммарных значений параметров за установленный промежуток времени, производится по следующим формулам (4.2.1.4, 4.2.1.5):

$$ByteCount_{\Delta T} = \sum_{N=1}^{N=\Delta T/TS} BC\_delta_{N2} \quad (4.2.1.4);$$

$$PacketCount_{\Delta T} = \sum_{N=1}^{N=\Delta T/TS} PC\_delta_{N2} \quad (4.2.1.5);$$

Отчасти данные структуры данных, отображенные выше, схожи с теми, которые были приведены в рамках метода идентификации трафика в Главе 2 настоящей диссертационной работы, однако текущие структуры описывают потоки уровня управления в Программно-конфигурируемых сетях. В то время, как в Главе 2 исследовались потока на уровне передачи пользовательских данных.

4.2.2. Исследуемая модель и обоснование работоспособности метода через многопараметрический анализ

В рамках данных исследований необходимо было изначально проверить наличие зависимости между активностью потоков OpenFlow (изменения метаданных) и изменением нагрузки аппаратной части контроллера SDN.

На рисунке 4.2.2.1 отображена принципиальная схема стенда.

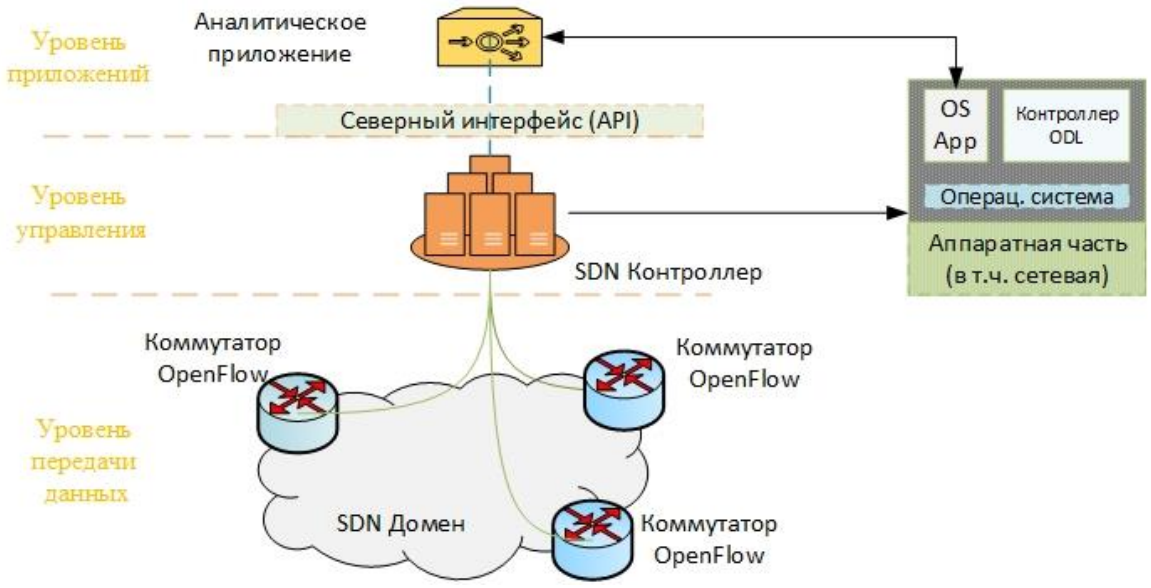

Рисунок 4.2.2.1 – принципиальная схема стенда

Для проверки зависимости между изменением активности служебных потоков OpenFlow (изменения метаданных) и нагрузки аппаратной части SDN контроллера было разработано два приложения «OS App» и «Аналитическое приложение». «OS App» - приложение-сервер для операционной системы Linux, на которой развернут SDN-контроллер OpenDaylight Beryllium SR4. Данное приложение запрашивает значения параметров аппаратной части (ядра процессора, оперативная память) у операционной системы Linux Ubuntu через системную утилиту.

Значения показателей доступны через REST API этого приложения сторонним приложениям по служебной сети. «Аналитическое приложение» - данное приложение

разработано также в виде сервера, но уже работающего поверх северного интерфейса контроллера REST API. Данное приложение в том числе запрашивает значения параметров у «OS App» через его API по служебной сети уровня управления, и в конечном итоге формирует набор данных параметров для оценки их зависимости.

Проведя анализ существующих математических методов для оценки зависимости между параметрами, было предложено использовать многопараметрический корреляционный анализ.

Согласно теории многопараметрического корреляционного анализа определяется ряд данных для сравнения и формируется многомерная матрица данных (X), которая в последующем приводится к типовой U-матрице. Стоит отметить, что строки такой матрицы (не приведенной), соответствуют результатам регистрации всех наблюдаемых параметров объектов (поток OpenFlow и аппаратная часть SDN-контроллера) в одном эксперименте, а столбцы содержат результаты наблюдений за одним параметром (фактором) во всех экспериментах. Для определения рядов обозначим количество параметров через m, где (m>1), а количество наблюдений – через n. В матрице элемент xij соответствует значению j-ого параметра в i-ом наблюдении. При этом допускается наличие пустых значений некоторых элементов, например, которые могут возникнуть из-за пропусков в регистрации значений параметров. В контексте данной исследовательской задачи регистрация параметров происходит каждую секунду. Однако для многомерного анализа желательно устранить пропущенные значения.

Для этого существует два подхода: вычеркивание соответствующих строк матрицы или занесение средних значений вместо отсутствующих. В данной диссертационной работе входные ряды данных подвергаются нормированию перед непосредственным корреляционным анализом на основе механизма занесения средних значений вместо отсутствующих.

Дальнейшие методы обработки матрицы X основаны на следующем предположении: если объект исследования подвергнуть новому обследованию и

получить другую матрицу данных, то после ее обработки с помощью тех же методов будут получены результаты, близкие к результатам первой матрицы. Данное предположение основано на статистической гипотезе формирования матрицы.

Таким образом, объектом исследования в многомерном анализе является многомерная случайная величина, представленная выборкой конечного объема. Стоит также отметить, что параметры, характеризующие объект исследования, имеют разный физический смысл, и матрица данных существенно изменяется, если изменяются шкалы, в которых измеряются выбранные параметры. Соответственно матрица данных приводится к стандартному виду, то есть стандартизируются значения параметров (вариант). Как уже отмечалось ранее, стандартизированную матрицу будем обозначать через U.

Матрица X, согласно ряду параметров, построенная на основании матрица $\mathrm{DataSet}_{\mathrm{ML}}$, выглядит следующим образом:

$$X = \begin{matrix} ByteCount & PacketCount & CPU & RAM \\ x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{i1} & x_{i2} & x_{i3} & x_{i4} \end{matrix} \quad (4.2.2.1);$$

Преобразование матрицы X, сформированной на основе выбранных параметров (4.2.1.1) потока и дополнительных данных о нагрузке контроллера в процентном значении каждого, к стандартизированной матрице U происходит следующим образом:

По каждому исследуемому параметру j = 1, 2, ... ,4 вычисляются взвешенные оценки по следующей формуле:

$$u_{ij} = \frac{\left(x_{ij} - \mu_1(x_j)\right)}{\delta(x_j)}, \text{при i} = 1, 2, \ldots, \text{n; j} = 1, 2, \ldots, 4 \quad (4.2.2.2);$$

В формуле (4.2.2.2) используются следующие два параметра, а именно математическое ожидание $\mu_1(x_j)$ и дисперсия $\delta^2(x_j)$, которые рассчитываются по следующим формулам:

$$\mu_1(x_j) = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \quad (4.2.2.3);$$

$$\mu_2(x_j) = \delta^2(x_j) = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \mu_1(x_j))^2 \quad (4.2.2.4);$$

Таким образом рассчитав элементы "u_{ij}" составляется матрица-U, которая становится последующим объектом обработки.

Стоит отметить, что воздействие общих факторов, а также наличие объективных закономерностей в поведении исследуемых объектов приводят лишь к появлению так называемой статической зависимости. Статической называют зависимость, при которой изменение одной из величин влечет изменение распределения других, и эти величины, как известно, принимают некоторые значения с определенными вероятностями.

Существует также частный случай статической зависимости, которая более подходит к рассматриваемой в данной диссертационной работе модели, а именно корреляционной зависимости, которая в свою очередь характеризует взаимосвязь значений одних случайных величин со средним значением других. Стоит отметить, что корреляционная зависимость описывает обычно причинную зависимость между значениями используемых параметров исследуемой аналитической модели.

Таким образом, корреляционная зависимость определяется различными параметрами, среди которых наибольшее распространение получили показатели, характеризующие взаимосвязь двух случайных величин (так называемые – парные показатели):

- корреляционный момент,
- коэффициент корреляции.

Оценка *корреляционного момента (коэффициента ковариации*) двух вариант x_j и x_k вычисляется по исходной матрице-X**:**

$$\xi_{jk} = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \mu_1(x_j))(x_{ik} - \mu_1(x_k)) \text{ (4.2.2.5);}$$

Коэффициент ковариации ρ_{jk} нормированных случайных величин называют *коэффициентом корреляции,* и его оценка рассчитывается по следующей формуле:

$$\rho_{jk} = \frac{1}{n}\sum_{i=1}^{n}(u_{ij}x_{ik}) \text{ (4.2.2.6);}$$

При этом, коэффициент корреляции зависит не от значений случайных величин, а от их вариаций, так если значение величины увеличивается на порядок, то коэффициент не изменится. При этом, значение коэффициента корреляции лежит в пределах от – 1 до +1. Если случайные величины U_j и U_k независимы, то коэффициент ρ_{jk} обязательно равен нулю, однако обратное утверждение неверно. Коэффициент корреляции ρ_{jk} характеризует значимость линейной связи между случайными величинами (параметрами):

- при ρ_{jk} = 1 значения u_{ij} и u_{ik} полностью совпадают, т.е. значения параметров принимают одинаковые значения. Иначе говоря, имеет место функциональная зависимость, то есть зная значение одного параметра, можно однозначно указать значение другого параметра;
- при ρ_{jk}= –1 величины u_{ij} и u_{ik} принимают противоположные значения. И в этом случае также имеет место функциональная зависимость;
- при ρ_{jk}=0 величины u_{ij} и u_{ik} практически не связаны друг с другом линейным соотношением. Однако в таком случае, не означает отсутствия каких-то других (например, нелинейных) связей между параметрами;

- при $|\rho_{jk}|>0$ и $|\rho_{jk}|<1$ однозначной линейной связи величин u_{ij} и u_{ik} нет. И чем меньше абсолютная величина коэффициента корреляции, тем в меньшей степени по значениям одного параметра можно предсказать значение другого.

Ранее была упомянута так называемая *функциональная зависимость*, которая, по сути, отображает связь рассматриваемой величины с одной или множеством других величин, если эта величина зависит только от этого множества факторов. Однако в данном случае стоит учитывать то, что функциональные связи являются больше математическими абстракциями, так как в реальных ситуациях существует бесконечно большое количество свойств самого объекта и внешней среды, влияющих друг на друга. Поэтому, говоря о возможной функциональной зависимости стоит учитывать то, что выбранный ряд параметров, на основе которых производится непосредственный корреляционный анализ, является ограниченным. При этом ряд параметров определяется в каждой конкретной практической задаче, например, как в текущей исследовательской задаче, обоснованный в данной статье ряд параметров, выбранных для проведения анализа, ограничен только четырьмя, описывающими метаданные суммы служебных потоков и нагрузки контроллера SDN.

Таким образом, в данной задаче (изучение зависимости между активностью служебных потоков и нагрузки контроллера SDN), аналитическим приложением производится расчет следующих показателей.

Примем следующие условные обозначения:

- ByteCount – BtCt;
- PacketCount – PcCt;
- CPU value – Cpu;
- RAM value – Ram;

$$\xi_{(1x4)_X} = \begin{vmatrix} BtCt|PcCt & BtCp|Cpu & BtCp|Ram & PcCt|Ram \\ \xi_{1x1} & \xi_{1x2} & \xi_{1x3} & \xi_{1x4} \end{vmatrix} \quad (4.2.2.7);$$

$$\rho_{(1x4)_U} = \begin{vmatrix} BtCt|PcCt & BtCp|Cpu & BtCp|Ram & PcCt|Ram \\ \rho_{1x1} & \rho_{1x2} & \rho_{1x3} & \rho_{1x4} \end{vmatrix} \text{ (4.2.2.8);}$$

В матрицах (4.2.2.7, 4.2.2.8) в первой строчке отображается отношение параметров, между которыми рассчитан соответствующий показатель (корреляционный момент или коэффициент).

4.2.3. Применение Искусственной Нейронной Сети для прогнозирования нагрузки на контроллер SDN

Анализ актуальности применения инструментов Искусственного Интеллекта в современных и перспективных сетях связи был достаточно объемно приведен ранее в данной диссертации. При этом Искусственная Нейронная Сеть была предложена в разработанном методе идентификации трафика в сетях связи на основе метаданных потоков на уровне передачи данных, в рамках 2 Главы настоящей диссертационной работы. В последнее время ИНС достаточно широко используется для решения различных задач из различных областей жизнедеятельности человека. Например, такие задачи, как распознавание текста, речи, прогнозирование сложных моделей в большинстве своем на данный момент решают с помощью ИНС, достигая при этом высоких результатов. Широко в обиходе используются и голосовые помощники, которые стали составлять новый вид интерфейса между человеком и ИТ-системой. На данный момент существует большая разновидность нейронных сетей. В данной задаче используется нейронная сеть для прогнозирования нагрузки.

Архитектура разработанной ИНС отображена на рисунке 4.2.3.1.

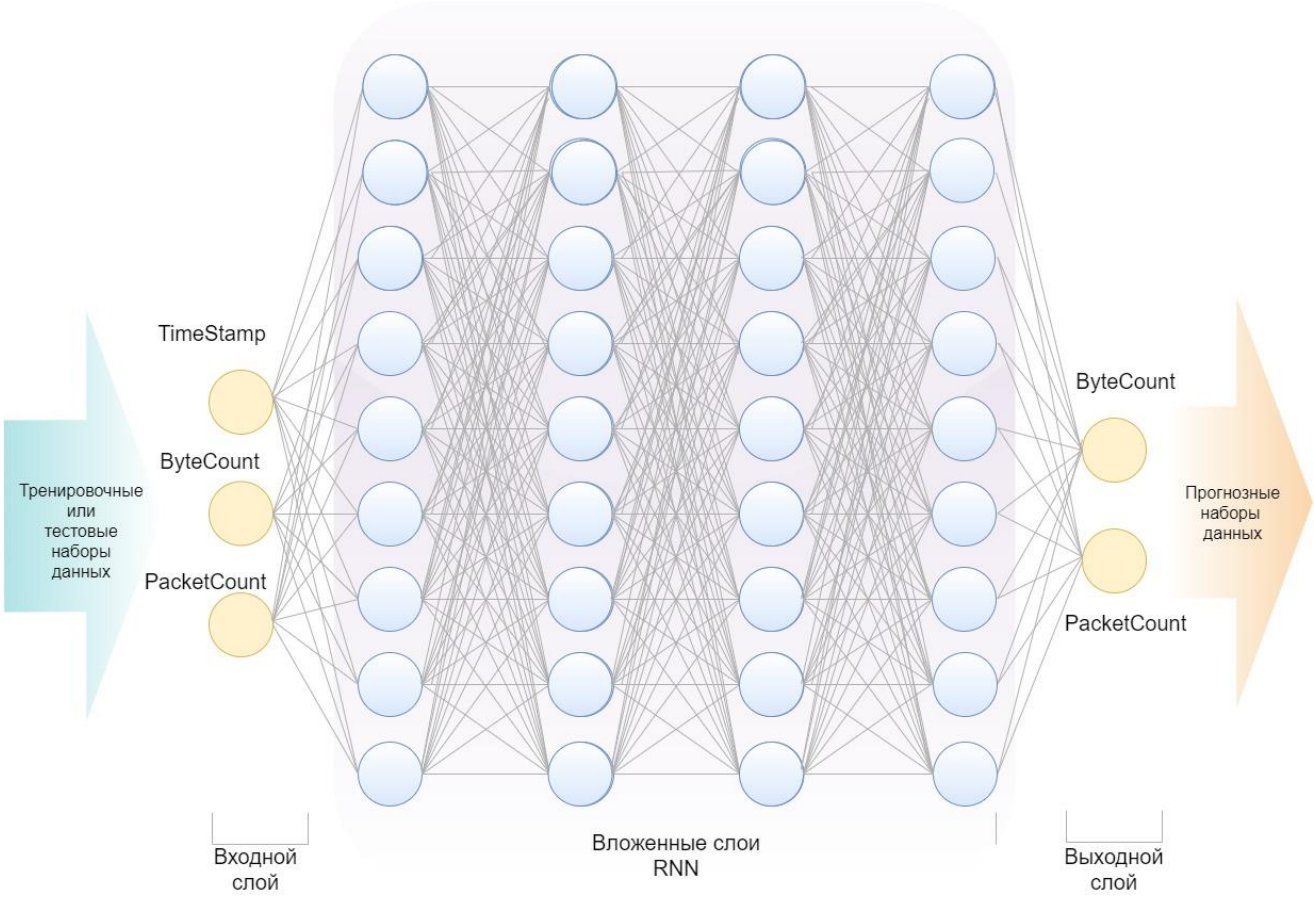

Рисунок 4.2.3.1 – Архитектура ИНС

Работа ИНС: на входной слой нейронов (так называемые плейсхолдеры) подается поток данных с сформированного $\mathrm{DataSet}_{\mathrm{ML}}$. Плейсхолдеры соединены с первым слоем нейронной сети полносвязанной структурой. Выходными являются два нейрона, которые генерируют ряд (набор данных) предсказанных значений.

Нейронная сеть получает на вход данные фиксированной длины, для этого изначальный набор данных делиться на сегменты по 200 строк. Также исходный набор данных разделяется на два набора данных: обучающий и практический (тестовый), в соотношении 8:2. Архитектура нейронной сети является полностью рекуррентной и содержит 4 полносвязанных вложенных уровня нейронов, каждый из которых состоит из 10 нейронов (рисунок 4.2.3.1).

Параметры обучения Искусственной Нейронной Сети:

- Оптимизатор: Adam
- Количество эпох: 20
- Количество образцов на итерацию: 1024
- Скорость обучения: 0.0025.

4.2.4. Результаты тестирования

Практические испытания предложенного метода мониторинга и прогнозирования нагрузки на контроллер программно-конфигурируемой сети разделены на две стадии. Первой стадией является проведение исследования с целью подтверждения установленной гипотезы о прямом влиянии активности потока (изменения метаданных) служебных сообщений OpenFlow на контроллер от коммутаторов SDN и соответственно на его аппаратную часть. Второй же частью практических испытаний, при полученных положительных результатах первой части, является разработка и обучение Искусственной Нейронной Сети с целью мониторинга и построения прогнозов активности служебных потоков OpenFlow.

Для проведения практических испытаний был собран стенд, структура и описание которого приведены в пункте 4.2.1 настоящей диссертационной работы.

Для расчета показателей зависимости между исследуемыми параметрами было разработано программное обеспечение (ПО) на языке программирования Python. Данное ПО сначала формирует набор данных, структура которого так отображена в пункте 4.2.2 настоящей диссертационной работы, после чего производит расчет соответствующих показателей ξ_X и ρ_U.

Как уже было отмечено ранее при обосновании выбора данного математического метода, корреляционный момент имеет определенную особенность,

а именно зависит от единиц измерения независимых величин, что не дает в полной мере оценить уровень корреляции двух величин.

Для этого изначальная матрица приводится к виду нормированной по формулам, указанным выше. На основе данной матрицы был построен также точечный график распределения взвешенных оценок относительно друг друга. Полученный график строился по выборке 100 значений из общего набора данных и отображен на рисунке 4.2.4.1.

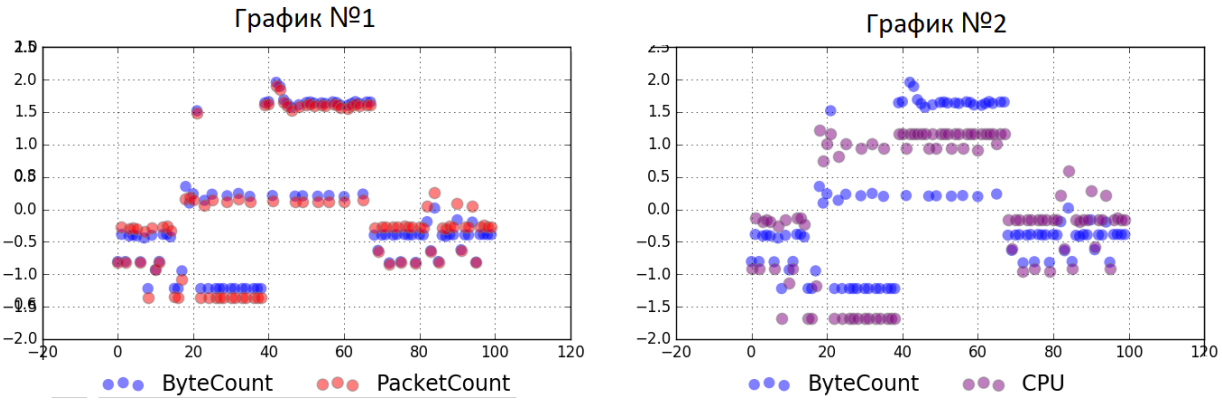

Рисунок 4.2.4.1 Точечный график распределения взвешенных оценок

На рисунке 4.2.4.1 отображены наиболее интересующие результаты относительно взаимосвязи ByteCount и нагрузки CPU. На графике 1 отображена естественная зависимость между двумя параметрами, которые напрямую связаны, для примера оценки параметров. На графике 2 рисунка 4.2.4.1 отображены распределения взвешенных оценок уже различных параметров, которые необходимо оценить с точки зрения зависимости изменений и можно сделать вывод, что согласно графику №2, существует зависимость между изменениями параметра ByteCount суммы служебных потоков и нагрузки на центральный процессор контроллера Программно-конфигурируемой сети.

Для оценки уровня коррелированности исследуемых величин рассчитывается коэффициент корреляции. В таблице 1 приведены рассчитанные показатели корреляции ρ_U между величинами, отображенными на графиках рисунка 4.2.4.1.

Таблица 4.2.4.1 – Матрица коэффициентов корреляции

ρ	$BtCt\|PcCt$	$BtCp\|Cpu$
значение	0.98	**0.89**

Для анализа полученных данных, отображенных в таблице 4.2.4.1 обратимся к свойствам показателя коэффициента корреляции, которые были отображены ранее при описании метода. Все полученные значения коэффициентов корреляции удовлетворяют следующему свойству $|\rho_{jk}|>0$ и $|\rho_{jk}|<1$. На основе данного свойства возможно сделать следующий вывод: существует связь между исследуемыми параметрами, и при этом чем больше величина коэффициента корреляции, тем в большей степени по значениям одного параметра возможно построить прогноз значения другого. Из полученных значений в данной работе интересует значение коэффициента корреляции между $BtCt|Cpu$ равный 0,89 соответственно (см. таблица 4.2.4.1). Значение данного показателя позволяет сделать вывод о существовании зависимости между активностью служебных потоков OpenFlow и нагрузкой аппаратной части SDN-контроллера. При этом, стоит отметить, что явной линейной зависимости между ними нет. При этом полученный результат позволяет сделать вывод о том, что предложенный метод мониторинга и прогнозирования нагрузки на котроллер может эффективно работать. И соответственно, для оценки работы/нагрузки контроллера Программно-конфигурируемой сети может быть достаточно построение прогнозной аналитической модели суммы служебных потоков OpenFlow, где суммарные счетчики группы служебных потоков (метаданные) является прогнозируемым параметром.

Для проведения второй части практического исследования, стенд был модернизирован относительно первой части практических испытаний. Вторая часть практических испытаний заключалась в обучении разработанной Искусственной

Нейронной Сети, архитектура и параметры которой приведены в пункте 4.2.3 настоящей диссертационной работы.

По полученному набору данных $\mathrm{DataSet_{ML}}$ была построена диаграмма разброса значений. Диаграмма приведена на рисунке 4.2.4.2. На данной диаграмме визуально возможно выделить несколько областей (кластеров) распределения точек, в области которых превалирует соответствующее значение среднего значения длины пакета в потоке (выделены овалами).

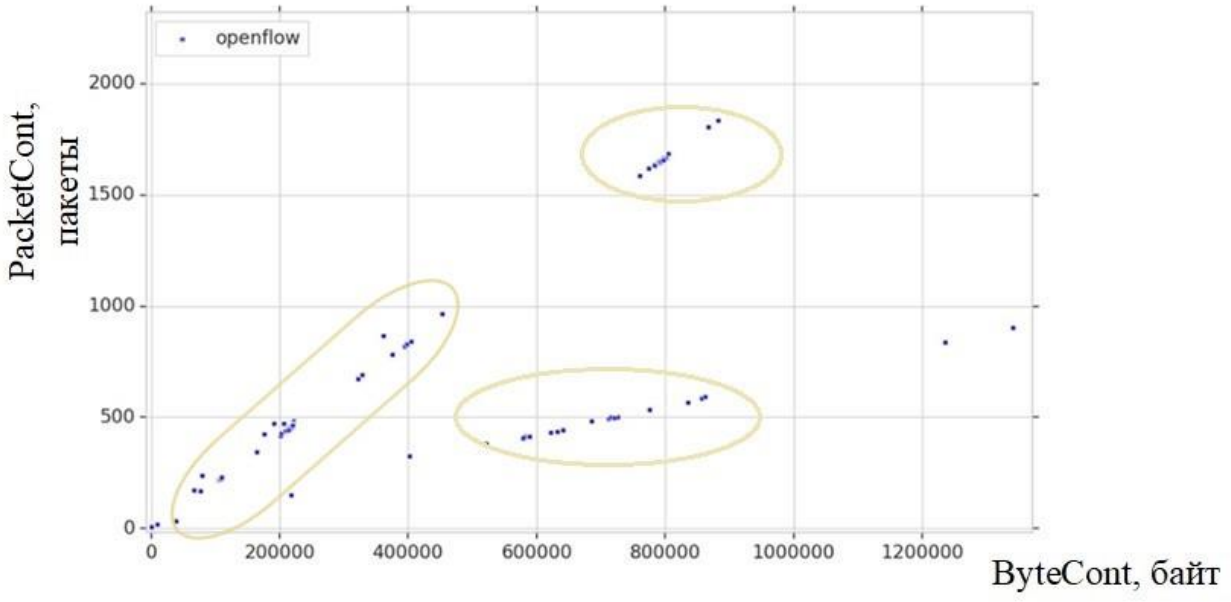

Рисунок 4.2.4.2 – Точечная диаграмм разброса значений набора данных $DataSet_{ML}$

В течении обучения Искусственной Нейронной Сети (оценки точности прогнозирования нейронной сети), в качестве параметра оценки работы, наблюдался параметр MSE - Mean Square Error (с англ. среднеквадратическая ошибка). Изменение значения параметра MSE приведено на рисунке 4.2.4.3.

Где MSE оценивается по следующей формуле:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

где Y-вектор наблюдаемых значений прогнозируемой переменной, $\hat{Y}_i$ – вектор спрогнозированных значений. Другими словами, MSE это среднее квадратов ошибок

прогноза. Данная оценка является простой и достаточно часто применяемой в подобного вида задачах.

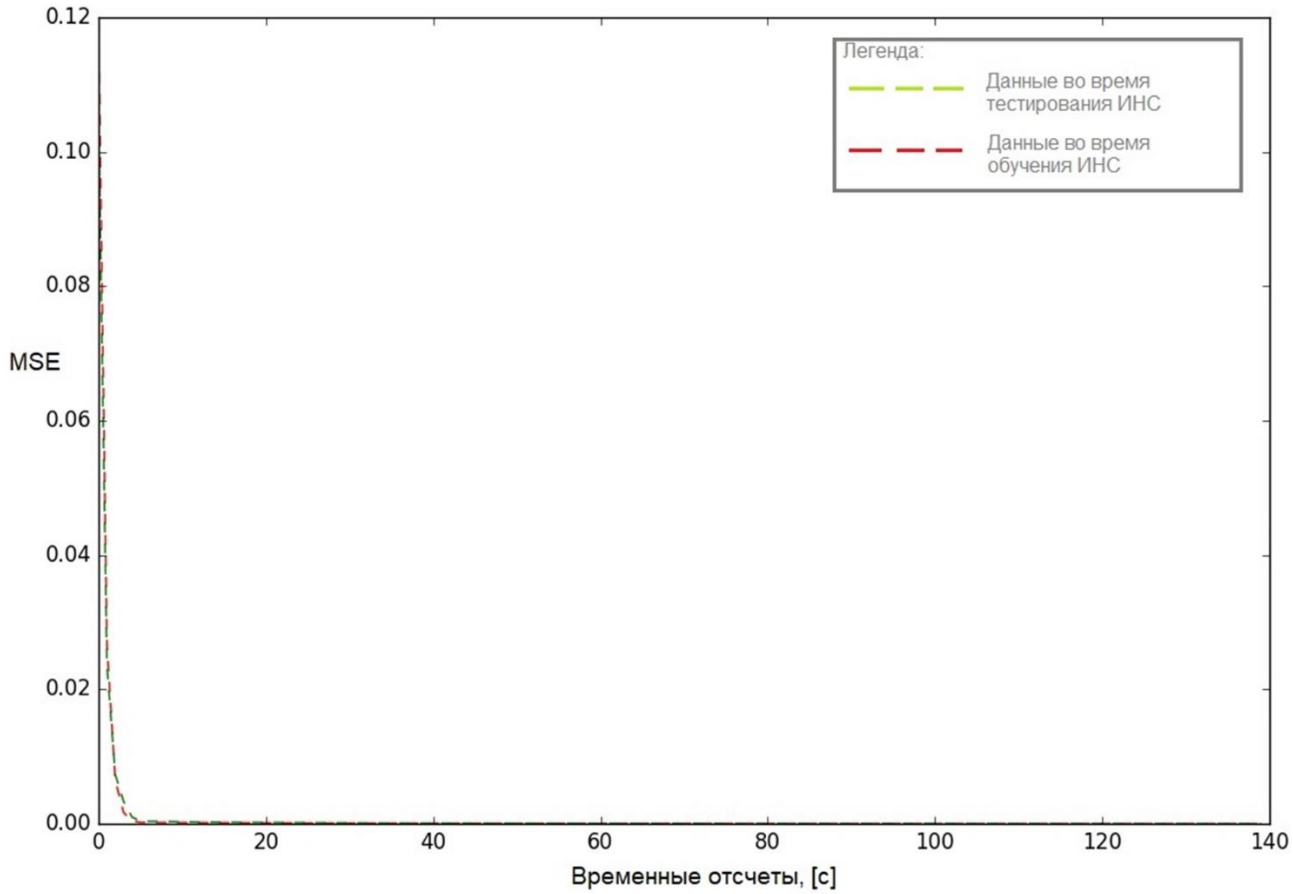

Рисунок 4.2.4.3 – Процесс обучения и тестирования ИНС

На графике рисунке 4.2.4.3 приведено два графика, которые идут близко друг к другу. Красной пунктирной линией отображается изменение параметра MSE при работе сети на обучающем наборе данных, зеленым отображается изменение параметра MSE на рабочем (реальном) наборе данных.

По графику видно, что выбранная архитектура и параметры нейронной сети удовлетворяют для формирования прогнозирования нагрузки. В конечном итоге, получились следующие параметры:

$$MSE_{Train} = 4.54 * 10^{-6},$$

$$MSE_{TEST} = 1.5 * 10^{-5}.$$

Также, в процессе обучения нейронной сети, отслеживался процесс ее обучения на формирование прогнозов. Для наглядности процесса, строился график реальных и спрогнозированных значений нейронной сети на промежутке данных в 20 тыс. отсчетов. На рисунке 4.2.4.4 приведены несколько снимков экрана в процессе обучения нейронной сети, отображающие ее прогресс.

В каждой из областей (графики №1,2,3,4), приведенных на рисунке 4.2.4.4, есть два графика синего и зеленого цвета, отображающие реальные значения и спрогнозированные.

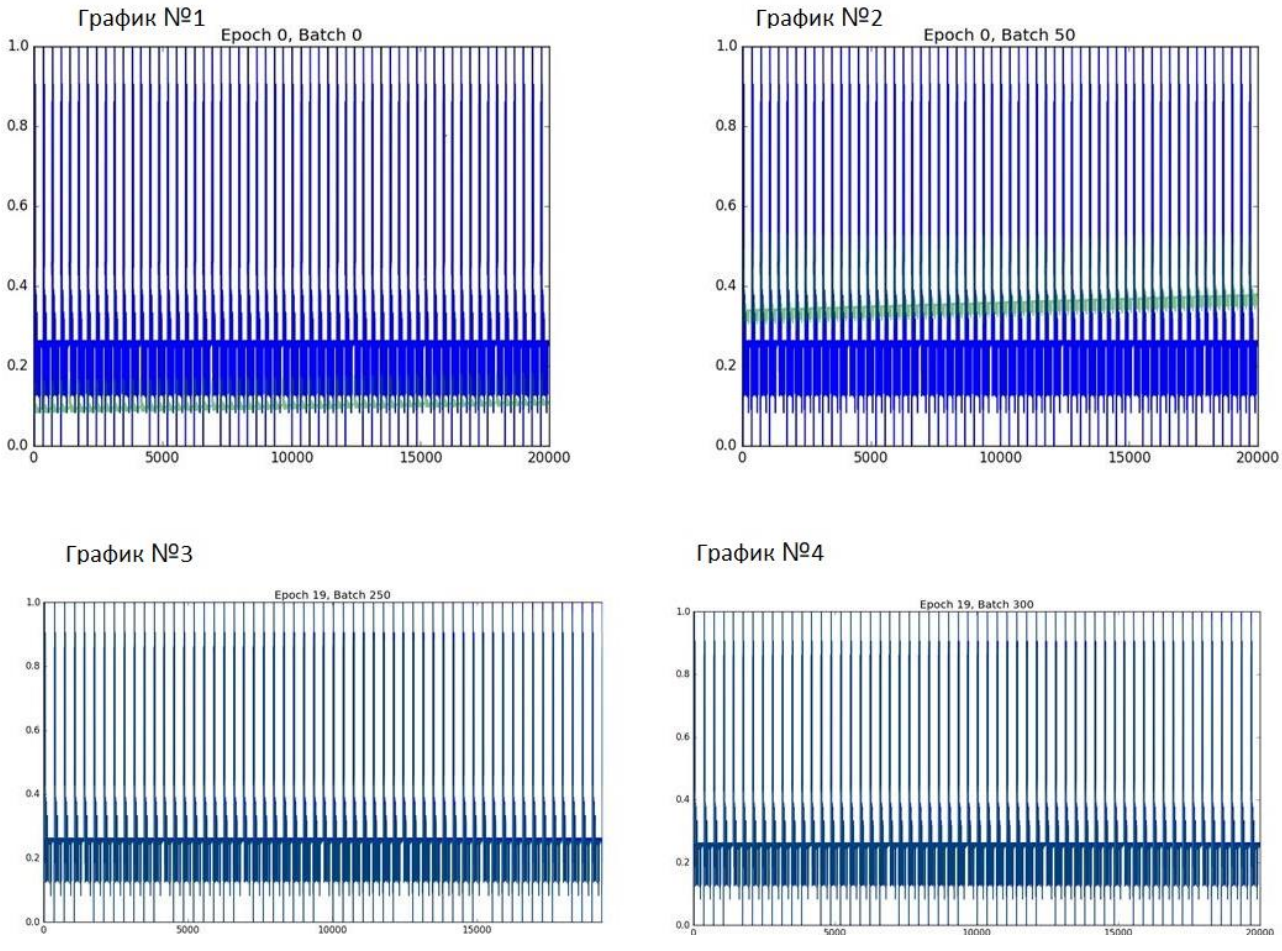

Рисунок 4.2.4.4 – Процесс обучения Искусственной Нейронной сети

График №1 на первой линии рисунка 4.2.4.4 с подписью (Epoch 0, Batch 0) отображает реальный и первый прогноз. На графике видно, что спрогнозированные значения сильно отличаются от реальных. То есть график зеленого цвета сильно ниже основного – синего графика.

Правый график №2 на первой линии рисунка 4.2.4.4 с подписью (Epoch 0, Batch 50) отображает следующий прогноз нейронной сети, после первого (график №1). По графику видно (зеленый), что теперь спрогнозированные значения превышают реальные, при этом график достаточно сильно завышен.

Нижние два графика (график №3 и график №4), с подписями (Epoch 19, Batch 250), (Epoch 19, Batch 300) соответственно отображают последние шаги обучения сети. По данным графикам видно, что Искусственная Нейронная Сеть обучилась и теперь может корректно прогнозировать активность суммы служебных потоков OpenFlow в реальной программно-конфигурируемой сети на уровне управления. Видно, что на последнем (нижнем правом графике №4), реальный график визуально совпадает с прогнозированным.

В результате после того, как Искусственная Нейронная Сеть обучилась, программное обеспечение сохраняет ее состояние архитектуры и параметров так, чтобы из другого программного модуля сервера «Аналитическое приложение» было возможно загрузить ее состояние. Далее, программный модуль, который загрузит состояние обученной нейронной сети сможет формировать прогнозные данные на поступающие тестовые данные, которые ИНС получает на входные нейроны (плейсхолдеры).

4.3. Выводы по Главе №4

1. В данной главе приводится исследование проблемы мониторинга контроллеров Программно-конфигурируемых сетей как систем управления в сетях связи 5G/IMT-2020 и последующих поколений (сетей связи 2030). Приводится вывод о необходимости проведения мониторинга и прогнозирования нагрузки на контроллеры SDN с целью обеспечения устойчивости системы в условиях жестких рамок качества обслуживания предоставляемых услуг, в то время как контроллеры SDN являются самой уязвимой частью данного типа сетей. Даже учитывая системы мульти контроллерного взаимодействия и резервирования подобных систем, прогнозная аналитика с целью проведения предупреждающих действий с целью сохранения устойчивости управляющей системы является одной из самых актуальных и важных задач в концепции внедрения технологий Искусственного Интеллекта в сетях связи. В рамках исследования данного вопроса в текущей главе диссертации был раскрыт вопрос архитектурных подходов к организации мониторинга серверных систем, к числу которых относится и контроллер SDN. В результате были приведены отрицательные стороны «стандартного» подхода (размещения специализированного ПО параллельно в ОС с ПО контролера SDN и его работа через обращение к системным ОС-проприетарным утилитам) к мониторингу нагрузки сервера, если проводить оценку через призму развертывания и сопровождения работы контроллера Программно-конфигурируемой сети.

2. Для разрешения проблемных вопросов при рассмотренном обычном подходе к мониторингу нагрузки контроллера SDN через системный уровень получения данных, в данной Главе был предложен метод мониторинга и прогнозирования нагрузки на контроллер SDN путем аналитики метаданных суммы служебных потоков OpenFlow уровня управления.

3. С целью подтверждения гипотезы о возможности прогнозирования нагрузки на вычислительные ресурсы контроллера SDN на основе метаданных суммы служебных потоков уровня управления был предложен метод проверки через применение многопараметрического корреляционного анализа.

4. Для апробации был разработан стенд модельной программно-конфигурируемой сети, где на первой стадии был реализован метод многопараметрического корреляционного анализа с целью проверки выдвинутой гипотезы. Для этого были параллельно собраны данные о нагрузке вычислительных ресурсов контроллера с помощью системных утилит ОС и метаданные служебных потоков OpenFlow. Расчет оценочных характеристик показал, что действительно изменение активности служебных потоков напрямую, однако нелинейно, влияет на нагрузку вычислительных ресурсов контроллера SDN.

5. Для прогнозирования нагрузки на контроллер Программно-конфигурируемой сети было предложено использовать рекуррентную Искусственную Нейронную сеть с 4-мя вложенными уровнями. Данный метод позволяет получать достаточно качественный прогноз нагрузки при корректном подборе параметров обучения и архитектуры ИНС.

6. Для апробации предложенного метода было разработано программное обеспечение на языке программирования Python в виде программного модуля Аналитического приложения, работающего с контроллером сети через северный интерфейс REST API.

7. Результат апробации показал работоспособность предложенного метода прогнозирования, где в качестве оценочной характеристики был использован критерий среднеквадратической ошибки MSE. В результате обучения и тестирования разработанной ИНС, значение MSE в тестовом варианте составил $MSE_{TEST} = 1.5 * 10^{-5}$, что является достаточно высоким показателем.

8. В рамках данной главы был исследован достаточно актуальный и важный вопрос в рамках реализации сетей связи с ультрамалыми задержками. Такие сети должны быть устойчивы к различного вида нагрузкам. При этом7 с целью эффективного предоставления вычислительных и сетевых ресурсов для систем управления необходимо иметь прогнозные данные о нагрузке на системы управления для проведения предупреждающих действий с целью обеспечения устойчивости

системы. В данном направлении также исследуются вопросы мульти контроллерного взаимодействия и распределения нагрузки на несколько контроллеров в Программно-конфигурируемой сети, однако для совершения управляющего действия по переносу задач на другой контроллер необходимо оценить грядущую нагрузку. Для этого необходимо развивать класс служебных приложений программно-конфигурируемой сети, способных с помощью технологий Искусственного Интеллекта предоставлять необходимую информацию как другим приложениям/системам, в рамках ранее озвученного в данной диссертации, конвейера машинного обучения (согласно рекомендации МСЭ-Т) так и администратору сети. При этом данные приложения должны удовлетворять требованиям к современному ПО, например, таким как: гибкое изменение, масштабируемость, переносимость, мультивендорность, скорость работы и так далее. В рамках данного класса задач, стоит еще раз отметить те возможности «программирования ресурсов», которые предоставляют Программно-конфигурируемые сети, что в свою очередь позволяет производить автоматизацию и интеллектуализацию сетей связи 5G/IMT-2020 и последующего поколения.

9. Таким образом, решая различные вопросы мониторинга и управления в сетях связи возможно прийти к единому, интеллектуальному ядру сети, которое будет подстраиваться и изменяться самостоятельно, без вмешательства человека, для соответствующего трафика. С учетом особенностей трафика Умных сервисов и жестких критериев качества к их реализации, только умная, интеллектуальная сеть сможет их предоставить и при этом обеспечить высокую гарантию работы инфраструктуры.

ЗАКЛЮЧЕНИЕ

В диссертационной работе были получены следующе основные результаты:

1. Был проведен анализ концепций современных и перспективных сетей связи, в том числе учитывая долгосрочные до 2030 года перспективы. Было установлено, что сети 5G/IMT-2020 и последующего поколения – сети связи 2030, обладают такими новыми характеристиками, как «ультрамалые задержки», «сверхнадежная связь», «сверхплотные сети», что требует разработки новых методов их построения.

2. Так как определена необходимость обеспечения в характеристиках, в указанных в п.1, требуется рассмотреть новые технологии и методы построения сетей связи и предоставления услуг, таких как: Программно-конфигурируемые сети, Виртуализация сетевых функций, граничные вычисления с множественным доступом (MEC), туманные вычисления (Fog), а также методы разработки и реализации программного обеспечения услуг, в частности – микросервисный архитектурный подход.

3. Кроме озвученных в п.2 методов и технологий построения сетей связи пятого и последующих поколений, требуется изменение принципов управления сетью и услугами через применение технологий Искусственного Интеллекта. В данном направлении установлено, что особенно актуальными вопросами являются: однозначная идентификация трафика на уровне передачи данных, прогнозирования данного трафика, прогнозирование нагрузки на контроллеры Программно-конфигурируемых сетей, и эффективное распределение вычислений на уровне глубоко интегрированной туманной инфраструктуры и пограничных вычислений с множественным доступом. В данной диссертационной работе предложены методы по идентификации трафика на основе метаданных потоков в SDN-сетях, прогнозирования нагрузки на контроллеры, основываясь на принципе аналитики

метаданных и инструментов ИИ, а также предложена структура взаимодействия интегрированных туманных и граничных вычислений с поддержкой микросервисных услуг.

4. Было установлено, что сети связи с ультрамалыми задержками и требования к ним формируют вектор развития сетей и услуг в сторону их децентрализации.

5. В данной диссертации на основе возможностей программируемости SDN-сетей и функциональных возможностей протокола OpenFlow, был разработан и предложен метод идентификации трафика в сетях связи на основе метаданных потоков и инструмента ИИ – искусственной нейронной сети.

6. Для апробации предложенного метода на базе модельной SDN сети лаборатории «Программно-конфигурируемых сетей» было проведено обучение ИНС и последующее тестирование. Результат тестирования показал работоспособность разработанного метода по факту идентификации трафика, позволяющий исключить внесение дополнительных задержек и изменение структуры потоков на уровне передачи данных.

7. Было проведено исследование причин и обоснование необходимости перехода к децентрализации облачных вычислений, как части вычислительно-сетевой инфраструктуры сетей связи пятого и последующего поколения. Установлено, что в качестве ключевых технологий в данном направлении являются технологии пограничных вычислений с множественным доступом (MEC) и туманных вычисления (Fog).

8. В данной диссертации было проведено исследование современных архитектур относительно построения перспективных услуг в сетях связи. Было определено, что микросервисный архитектурный подход является одним из самых актуальных и реализующих необходимые требования к современному программному обеспечению, в том числе к программному обеспечению современных и

перспективных услуг связи. Особенно обращается внимание на услуги сетей связи с ультрамалыми задержками и сверхнадежностью.

9. Была предложена структура/фреймворк взаимодействия распределенных туманных вычислений и пограничных вычислений с поддержкой микросервисных услуг.

10. В рамках предложенного фреймворка были рассмотрены две задачи, а именно: определение центра скопления пользователей, а также определение Fog-устройства, которое будет выбрано для живой миграции микросервиса услуги. На основе анализа были определены математические алгоритмы решения поставленных задач, такие как: К-средних и Роевого Интеллекта (PSO), которые в свою очередь входят в класс алгоритмов, относящихся к Искусственному Интеллекту. Полученные данные в результате программного моделирования показывают работоспособность предложенных алгоритмов с точки зрения фреймворка и перспективность развития данного решения.

11. Было также определено, что применение алгоритма роевого интеллекта в предлагаемом фреймворке, с целью рационального выбора Fog-устройства, позволяет уменьшить время выполнения функции микросервиса на величину до 70% относительно равновероятного распределения ресурсов.

12. В данной диссертации было также определена необходимость мониторинга и прогнозирования нагрузки управляющих систем сетей связи пятого и последующих поколений. Были приведены отрицательные стороны мониторинга с помощью системных программ и утилит, с точки зрения требований к устойчивости и ультрамалым задержкам в перспективных сетях связи.

13. В результате был предложен метод мониторинга и нагрузки контроллеров Программно-конфигурируемых сетей путем аналитики метаданных суммы служебных потоков OpenFlow уровня управления. С целью подтверждения гипотезы о возможности прогнозирования нагрузки на основе принципа аналитики метаданных

был предложен метод проверки через применение многопараметрического корреляционного анализа.

14. Для апробации предложенного метода прогнозирования был разработан стенд модельной программно-конфигурируемой сети, где на первой стадии был реализован метод многопараметрического корреляционного анализа с целью проверки гипотезы об использовании метаданных служебных потоков, результат тестирования которого подтвердил работоспособность предложенного метода. На второй стадии был реализован метод прогнозирования потоков служебного трафика в программно-конфигурируемой сети, позволяющий оценить нагрузку на контроллеры SDN. В результате обучения и тестирования разработанной ИНС, значение MSE в тестовом варианте составил $MSE_{TEST} = 1.5 \cdot 10^{-5}$, что является достаточно высоким показателем. Стоит также отметить, что предложенный метод позволяет исключить зависимость мониторингового ПО от особенностей АПК-решений производителей.

15. В данной диссертации были подняты вопросы внедрения технологий Искусственного Интеллекта, с целью обеспечения новых требований к современным и перспективным услугам связи, появившимся в свое время на основе технологий Интернета Вещей. Особенно было обращено внимание на сети связи с ультрамалыми задержками, которые на данный момент задают вектор развития сетей связи, и возможность их реализации через планомерное внедрение функций Искусственного Интеллекта в сети связи. В результате был определен ряд актуальных задач в области ИИ в сетях связи. В частности, в качестве прикладных исследований в диссертации были рассмотрены вопросы методов мониторинга и идентификации трафика на уровне передачи данных, методов построения вычислительной инфраструктуры с поддержкой микросервисных услуг, а также методов мониторинга и прогнозирования нагрузки на контроллеры программно-конфигурируемых сетей. В результате можно сделать вывод, что в процессе разрешения задач мониторинга и управления в современных и перспективных сетях связи возможно прийти к единому, интеллектуальному ядру сети, которое, в свою очередь, будет подстраиваться и

изменяться самостоятельно, без вмешательства администраторов, для соответствующего трафика, услуги. Стоит также обратить внимание на необходимость выстраивать распределенный Искусственный Интеллект в сетях связи, для обеспечения более эффективного использования вычислительных ресурсов и обеспечения устойчивости управляющих систем.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

АПК	Аппаратно-программный комплекс
АП	Ассоциативная Память
АЗУ	Ассоциативное Запоминающее Устройство
БД	База Данных
ДР	Дерево Решений
ИВ	Интернет Вещей
ИИ	Искусственный Интеллект
ИК	Исследовательская комиссия МСЭ
ИКТ	Инфокоммуникационные Технологии
ИНС	Искусственная Нейронная Сеть
ИТ	Информационные Технологии
M2M	Machine-to-Machine Communications
МСЭ	Международный Союз Электросвязи
МСЭ-Т	Международный Союз Электросвязи, сектор стандартизации телекоммуникаций
МСЭ-Р	Международный Союз Электросвязи, сектор стандартизации радиосвязи
ОЗУ	Оперативное Запоминающееся Устройство
ОС	Операционная Система
ООН	Организация Объединенных Наций
ПД	Уровень Передачи Данных
ПКС	Программно-Конфигурируемые Сети
ПО	Программное Обеспечение
СПбГУТ	Санкт-Петербургский Государственный Университет Телекоммуникаций имени проф. М.А. Бонч-Бруевича
СПД	Сети Передачи Данных

ССиПД	Сети связи и Передача Данных – кафедра СПбГУТ
СУБД	Системы Управления Базами Данных
ТИ	Тактильный Интернет
ТФоП	Телефонные Сети Связи общего Пользования
ЦОД	Центр Обработки Данных
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
ASF	Authentication Server Function
AF	Application Function
BPTT	Backpropagation Through Time
CAPEX	Capital Expenditure
CEF	Capability exposure function
D2D	Device-to-Device Communications
DD-Fog	Distributed Dynamic Fog Computing Framework
DB	Data Base
DPI	Deep Packet Inspection
DWDM	Dense-Wavelength Division Multiplexing
EB	Enhanced Mobile Broadband
ETSI	European Telecommunications Standards Institute
FG	Focus Group
GPS	Global Positioning System
GUI	Graphical User Interface
GW	Gateway
HMM	Hidden Markov Model
HTTP	Hyper Text Transfer Protocol

ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
ISO/IEC JTC 1	International Organization for Standardization / International Electrotechnical Commission Joint Technical Committee
ISC	Internal System Code
IoT	Internet of Things
IoTDM	Internet of Things Data Management
IaaS	Infrastructure as a Service
IMT	International Mobile Telecommunications
IP	Internet Protocol
KPI	Key Performance Indicators
LSTM	Long Short-Term Memory
LTE	Long-Term Evolution
ML	Machine Learning
MEC	Mobile Edge Computing
MSE	Mean Square Error
MMC	Massive machine type communications
MLFO	Machine Learning Function Orchestrator
MPLS	Multi-Protocol label switching
MVC	Model-View-Controller
NACF	Network Access Control Function
NB-API	NorthBound API
NSSF	Network Slice Selection Function
NFV	Network Function Virtualization
NFR	Network function registry function
OAM	Operations, administration and maintenance
OS	Operation System

ONF	Open Network Foundation
OPEX	Operating Expenditure
PaaS	Platform as a Service
PCF	Policy control function
PSO	Practical Swarm Optimization
QoS	Quality of Service
RAM	Random Access Memory
RBM	Restricted Boltzmann Machine
REST	Representational State Transfer
RNN	Recurrent Neural Networks
RRC	Radio Resource Control
RSVP-TE	Resource ReSerVation Protocol - Traffic Engineering
SaaS	Software as a Service
SDN	Software-Defined Networking
SMF	Session Management Function
SVM	Support Vector Machine
SOM	Self-Organizing Map
TCAM	Ternary Content Addressable Memory
TE	Traffic Engineering
ToS	Type of Service
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-reliable and Low Latency communications
USM	Unified Subscription Management function
VM	Virtual Machine
VR	Virtual Reality

VM	Virtual Machine
VR	Virtual Reality

СПИСОК ЛИТЕРАТУРЫ

1. Владыко А.Г., Мутханна А.С., Киричек Р.В., Волков А.Н., Маколкина М.А., Парамонов А.И. Программируемые сети SDN. Санкт-Петербург, 2019.

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

3. Бородин А.С. Искусственный интеллект в сетях связи пятого и последующих поколений. А.Н. Волков, А.С.Мутханна, А.Е. Кучерявый. Электросвязь №1, 2021, с.17-22

4. Волков А. Н., Мутханна А. С. А., Кучерявый А. Е. Сети связи пятого поколения: на пути к сетям 2030 // Информационные технологии и телекоммуникации. 2020. Том 8. № 2. С. 32–43. DOI 10.31854/2307-1303-2020-8-2-32-43.

5. Recommendation M.2083-0 IMT-Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. ITU-R, Geneva. – 2015.

6. Recommendation Y.3300 Framework of Software-defined networking. ITU-T, Geneva. – June 2014.

7. Кучерявый, А.Е. Тактильный Интернет. Сети связи со сверхмалыми задержками / А.Е. Кучерявый, М.А. Маколкина, Р.В. Киричек // Электросвязь. – 2016. – № 1. – С. 44-46.

8. Выборнова, А.И. Тактильный интернет: новые возможности и задачи / Выборнова, А.И.; Кучерявый, А.Е. // В сборнике: Проблемы техники и технологий телекоммуникаций ПТиТТ-2016 Первый научный форум "Телекоммуникации: теория и технологии" 3Т-2016. 2016, Самара, С. 133-134.

9. Recommendation Y.2060/Y.4000. Overview of Internet of Things. ITU-T, Geneva. – February 2012.

10. Technical Specification. FG-NET2030 – Focus Group on Technologies for Network 2030. Network 2030 Architecture Framework. ITU-T, Geneva. – June 2020.

11. Кучерявый А. Е. Сети связи с ультрамалыми задержками // Труды НИИР. 2019. № 1. С. 69–74.

12. Кучерявый, А.Е. Тактильный Интернет. Сети связи со сверхмалыми задержками / А.Е. Кучерявый, М.А. Маколкина, Р.В. Киричек // Электросвязь. – 2016. – № 1. – С. 44-46.

13. Recommendation Y.3172. Architectural framework for machine learning in future networks including IMT-2020. ITU-T, Geneva. – June 2019.

14. Recommendation Y.3170. Requirements for machine learning-based quality of service assurance for the IMT-2020 network. ITU-T, Geneva. – September 2018.

15. Recommendation Y.3104. Architecture of the IMT-2020 network. ITU-T, Geneva. – December 2018.

16. Recommendation Y.3174. Framework for data handling to enable machine learning in future networks including IMT-2020. ITU-T, Geneva. – February 2020.

17. Recommendation Y.3176. Machine learning marketplace integration in future networks including IMT-2020. ITU-T, Geneva. – September 2020.

18. Technical Specification ETSI TS 123 01 v16.6.0 Release 16. 5G. System architecture for the 5G System (5GS). ETSI, France. – October 2020.

19. Recommendation Y.3102. Framework of the IMT-2020 network. ITU-T, Geneva. – May 2018.

20. Russell, S., Norvig, P.: ‘Artificial Intelligence (A Modern Approach)’. 3rd ed. New Jersey: Prentice Hall, 1995. 1152 p.

21. Huawei’s Global Industry Vision (Report GIV 2025).

22. История машинного обучения. [Электронный ресурс] – Режим доступа. – URL: https://abv24.com/istoriya-mashinnogo-obucheniya.

23. Mohammadi, M., Al-Fuqaha, A., Sorour, S. Guizani, M.: ‘Deep Learning for IoT Big Data and Streaming Analytics: A Survey’, IEEE Communications Surveys & Tutorials.,

24. Tang, T., Zaidi, S.A.R., McLernon, D., Mhamdi, L. Ghogho, M.: ‘Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks’. In 2018 IEEE International Conference on Network Softwarization (NetSoft 2018), Montreal, Canada, Jun 2018.

25. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

26. Convolutional Neural Networks (LeNet) – Deep Learning 0.1 documentation. Deep Learning 0.1. LISA Lab. Дата обращения 20 апреля 2019.

27. Zhang, Q., Yang, L.T., Chen, Z. and Li, P.: ‘A survey on deep learning for big data’, Information Fusion, 2018,42, pp.146-157.

28. S. Zhongfu and Y.S.D. Keming,“Development Trend of Internet of Things and Perspective of Its Application in Agriculture”, Agriculture Network Information, vol. 5, (2010),pp. 5-8.

29. Negnevitsky, M.: ‘Artificial Intelligence - A Guide to Intelligent Systems’. 2nd ed. Essex: Addison-Wesley, 2005. 415 p.

30. Russell, S., Norvig, P.: ‘Artificial Intelligence (A Modern Approach)’. 3rd ed. New Jersey: Prentice Hall, 1995. 1152 p.,Rokach, L.: ‘Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography’, Computational Statistics & Data Analysis, 2008,53, (12), pp. 4046-4072.

31. Rokach, L.: ‘Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography’, Computational Statistics & Data Analysis, 2008,53, (12), pp. 4046-4072.

32. LeCun, Y., Bengio, Y., Hinton, G.: ‘Deep learning’, Nature, 2016, 521, (7553), pp. 436-444.

33. Deng, L.: ‘A tutorial survey of architectures, algorithms, and applications for deep learning’, APSIPA Transactions on Signal and Information Processing, 2014, 3, (e2),1-19.

34. Mohammadi, M., Al-Fuqaha, A., Sorour, S. Guizani, M.: ‘Deep Learning for IoT Big Data and Streaming Analytics: A Survey’, IEEE Communications Surveys & Tutorials.

35. Mac Queen, J.B.: ‘Some Methods for classification and Analysis of Multivariate Observations’. In Proc. of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, USA, 1967, pp. 281-297.

36. Khan, S.S., Ahmad, A.: ‘Cluster center initialization algorithm for K-means clustering’, Pattern recognition letters,2004, 25, (11), pp. 1293-1302.

37. Pal, N.R, Bezdek, J.C: ‘On cluster validity for the fuzzy c-means model’, IEEE Trans Fuzzy Syst, 1995,3 (3), pp. 370-379.

38. Phan, T.V., Bao, N.K., Park, M.: ‘Distributed-SOM: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks’, Journal of Network and Computer Applications, 2017,91, pp. 14-25.

39. Phan, T.V., Bao, N.K., Park, M.: ‘Distributed-SOM: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks’, Journal of Network and Computer Applications, 2017,91, pp. 14-25.

40. Fan, Z., Xiao, Y., Nayak, A., Tan, C.: ‘An improved network security situation assessment approach in software defined networks’, Peer-to-Peer Networking and Applications, 2017.

41. Kohonen, T.: ‘The self-organizing map’, Proceedings of the IEEE, 1990,78, (9), pp. 1464-1480.

42. Fan, Z., Xiao, Y., Nayak, A., Tan, C.: ‘An improved network security situation assessment approach in software defined networks’, Peer-to-Peer Networking and Applications, 2017.

43. Watkins, C.J., Dayan, P.: ‘Q-learning’, Machine learning, 1992,8, (3-4), pp. 279-292.

44. Koucheryvy, A., Muthanna, A., Volkov, A.: AI/machine learning for ultra-reliable low-latency communication, ITU News Magazine, No.5, December 2020. pp.62-65

45. Волков, А.Н. Идентификация трафика сервисов в сетях связи IMT-2020 и последующего поколения на основе метаданных потоков и алгоритмов машинного обучения / А.Н.Волков, А.Е. Кучерявый // Электросвязь. – 2020. – № 11. – С. 21-28.

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

47. Маколкина, М.А. Метод выгрузки трафика приложений дополненной реальности в многоуровневой системе граничных вычислений / М.А. Маколкина, А.А. Атея, А.С.А. Мутханна, А.Е. Кучерявый // Электросвязь. – 2019. – № 6. – С. 36-42.

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

49. Volkov A., Khakimov A., Muthanna A., Kirichek R., Vladyko A., and Koucheryavy A. Interaction of the IoT traffic generated by a Smart city segment with SDN core network // WWIC 2017 International Conference on Wired/Wireless Internet Communication, pp. 115–126. 0302-9743 eISSN: 1611-3349 Springer-Verlag GmbH (Heidelberg).

50. Волков А.Н. Структура распределенной динамической вычислительной системы туманных вычислений для микросервисов (DD-Fog) / А.Н. Волков // Электросвязь. – 2021. № 7. – С. 34-43.

51. Галимянов А.Ф., Галимянов Ф.А. Архитектура информационных систем / А. Ф. Галимянов, Ф. А. Галимянов. – Казань: Казан. ун-т, 2019. – 117 с

52. Tran T.X. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. Hajisami, A., Pompili, D. IEEE Communications Magazine 55, 2017, 54–61. doi:10.1109/MCOM.2017.1600863.

53. Атея А.А. Интеллектуальное ядро для сетей связи 5G и тактильного интернета на базе программно-конфигурируемых сетей / А.А. Атея, А.С. Мутханна, А.Е. Кучерявый // Электросвязь. – 2019. – № 3. – С. 34-40.

54. Волков, А.Н. Метод прогнозирования нагрузки на контроллеры SDN с помощью технологий Искусственного Интеллекта / А.Н.Волков, А.Е. Кучерявый // Электросвязь. – 2021. – № 2. – С. 31-38.

Приложение А. ЛИСТИНГ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОБРАБОТКИ ДАННЫХ С СЕВЕРНОГО ИНТЕРФЕЙСА КОНТРОЛЛЕРА ПРОГРАММНО-КОНФИГУРИРУЕМОЙ СЕТИ

Модуль обработки данных с SDN контроллера, получаемые через программный модуль API (access_data.py) разработанного Web-сервера, из класса NodeConnectionSwitch. В свою очередь программный модуль API реализует взаимодействие с контроллером SDN по северному программному интерфейсу.

```
#!env/bin/python
# -*- coding: ascii -*-
#import libraries
import json
import sys
import csv
#import parrent directory
sys.path.insert(0,'..')

from app.access_data import switches
from app.access_data import switchTablSize
from app.access_data import switchBuffSize
from app.access_data import switchTopoPort
from app.access_data import switchTopoID
from app.access_data import nodesPriority
from app.access_data import nodesStat
from app.access_data import nodesFlowTable
from app.access_data import SwReqIdent
from app.access_data import FlowTableInternalProc
import pandas
from pandas import DataFrame

newDataByte = []
GlobaldataByte = []
newDataPacket = []
GlobaldataPacket = []
newDataByteControlPlane = []
GlobaldataByteControlPlane = []
newDataPacketControlPlane = []
GlobaldataPacketControlPlane = []

class PrepareData(object):
                def __init__(self, counter):
                        super(PrepareData, self).__init__()
                        self.arg = counter
                def FilterFunc_OF_table(self):
                        '''
                                        Developer Note's: This Function is the filter for outgoing from API controller OF-tab
Removed next rows in table:
                                                - if in row colloums values "Packet count" & "Byte count" are zero
```

```
                                        - // if in row "Action" value is only to CONTROLLER
                                        - if in row "Ethernet Type protocol" are 0x88cc and 0x8809

Input list -- Input Data = [[switchID, IDTable, priority, portIngress, sourceMAC, destMAC,
                                                                                TypeEth,
ipSource, ipDest, "*", packCount, byteCount, timeDuration, actions], [..], ...]
                        '''
                        InputData  = FlowTableInternalProc()
                        data       = []
                        dataFilter = []

                        # Filtering on: packet & byte counts, ethernet type protocol
                        i = 0 #set count for cycle
                        while i < len(InputData):
                                try:
                                        if int(InputData[i][11])!=0 and int(InputData[i][12])!=0 and
InputData[i][6]!='0x88cc' and InputData[i][6]!='0x8809':
                                                data.append(InputData[i])
                                        i = i + 1
                                except IndexError:
                                        break
                        i = 0
                        while i < len(data) :
                                try:
                                        if 'CONTROLLER' in data[i][13]:
                                                i = i + 1
                                        else:
                                                dataFilter.append(data[i])
                                                i = i + 1
                                except IndexError:
                                        break
                        #print str(dataFilter)
                        return dataFilter

                def FilterFunc_OF_table_controlPlane(self):
                        '''Developer Note's: This Function is the filter for outgoing from API controller OF-table
data Removed next rows in table:
                                        - if in row colloums values "Packet count" & "Byte count" are zero
                                        - if in row "Action" value is haven't to CONTROLLER

Input list -- Input Data = [[switchID, IDTable, priority, portIngress, sourceMAC, destMAC, TypeEth, ipSource,
ipDest, "*", packCount, byteCount, timeDuration, actions], [..], ...] '''
                        InputData  = FlowTableInternalProc()
                        data       = []
                        dataFilter = []

                        i = 0 #set count for cycle
                        while i < len(InputData):
                                try:
                                        if int(InputData[i][11])!=0 and int(InputData[i][12])!=0:
                                                data.append(InputData[i])
                                        i = i + 1
                                except IndexError:
```

```
                                        break
                        i = 0
                        while i < len(data) :
                                try:
                                        if 'CONTROLLER' not in data[i][13]:
                                                i = i + 1
                                        else:
                                                dataFilter.append(data[i])
                                                i = i + 1
                                except IndexError:
                                        break

                        return dataFilter

                def FuncDevidedSW_OF_table(self):
                        FilteredFlow = self.FilterFunc_OF_table()

                        structOutMass = []
                        i = 0
                        while i < len(switches):
                                try:
                                        structOutMass.append([switches[i]])
                                        i = i + 1
                                except IndexError:
                                        break
                        i = 0
                        while i < len(structOutMass):
                                try:
                                        j = 0
                                        while j < len(FilteredFlow):
                                                try:
                                                        if structOutMass[i][0] == FilteredFlow[j][0]:
                                                        structOutMass[i].append(FilteredFlow[j])
                                                                j = j + 1
                                                        else:
                                                                j = j + 1
                                                except IndexError:
                                                        break
                                        i = i + 1
                                except IndexError:
                                        break
                        #print str(structOutMass[0]) + '\n' + str(structOutMass[1]) + '\n' + str(structOutMass[2])
                        return structOutMass

                def __del__(self):
                        print "deling", self


class CreateDataSets(object):
                        def __init__(self, FlowTable, IdSw,IdFl):
                        super(CreateDataSets, self).__init__()
                        self.FlowTable = FlowTable
                        self.IdFlow = IdFl
```

```
                self.IdSwitch = IdSw

        def CreateDataSet(self):
                '''
                This function for create Data Set.
                Input parameters are:
                        - Devided Flow Tables
                        - ID of switch for choose OF table from first parameter
                        - ID of Flow for choose Flow from Table
                        - Counter - for define number of rows in Data Set
                '''
                DevFlowTable = self.FlowTable
                IdSwitch     = self.IdSwitch
                IdFlow       = self.IdFlow

                TimeStamp = []; ByteCount = []; PacketCount = [];

                i = 0
                while i < len(DevFlowTable):
                        try:
                                if IdSwitch == DevFlowTable[i][0]:
                                        j = 1
                                        while j < len(DevFlowTable[i]):
                                                try:
                                                        if IdFlow == DevFlowTable[i][j][1]:
                                                                #print str(DevFlowTable[i][j])

ByteCount.append(DevFlowTable[i][j][11])

PacketCount.append(DevFlowTable[i][j][10])

TimeStamp.append(DevFlowTable[i][j][12])
                                                                        break
                                                        else: j = j + 1
                                                except IndexError:
                                                        break
                                        break
                                else: i = i + 1
                        except IndexError:
                                break

                return TimeStamp, ByteCount, PacketCount


        def calculation(self):

                TimeStamp   = []
                ByteCount   = []
                PacketCount = []

                InputData   = self.CreateDataSet()
                TimeStamp   = int(InputData[0][0])
                ByteCount   = InputData[1]
```

```
                PacketCount = InputData[2]

                #print str(InputData) + '\n'

                global newDataByte
                global GlobaldataByte

                global newDataPacket
                global GlobaldataPacket

                if (len(GlobaldataByte) == len(ByteCount)):
                        try:
                                print 'start calculation'
                                newDataByte   = int(ByteCount[0]) - int(GlobaldataByte[0])
                                newDataPacket = int(PacketCount[0]) - int(GlobaldataPacket[0])
                                GlobaldataByte[0] = int(ByteCount[0])
                                GlobaldataPacket[0] = int(PacketCount[0])
                        except IndexError:
                                        print 'Select another Flow, no data'

                elif (len(GlobaldataByte) != len(ByteCount)):
                        #this part of function working correctly
                        print 'during calculation'
                        GlobaldataByte = []
                        GlobaldataPacket = []
                        GlobaldataPacket.append(0)
                        GlobaldataByte.append(0)
                        newDataPacket = GlobaldataByte
                        newDataByte = GlobaldataByte

                #print 'Gloabal Data byte  == ' + str(GlobaldataByte)
                #print 'Global Data packet == ' + str(GlobaldataPacket)

                return str(TimeStamp), str(newDataByte), str(newDataPacket)


        def CreateCSV(self):
                '''
                        Function return data in CSV format:
                        .....
                '''
                InputData       = self.calculation()
                TimeStamp       = InputData[0]
                ByteCount       = InputData[1]
                PacketCount = InputData[2]

                with open('app/analytics/DataSets/DataSet.csv', mode='a') as csv_file:
                        fieldnames = ['TypeOfTraffic', 'TimeStamp', 'ByteCount', 'PacketCount']
                        writer = csv.DictWriter(csv_file, fieldnames=fieldnames)

                        writer.writerow({'TypeOfTraffic': 'iot', 'TimeStamp': TimeStamp, 'ByteCount':
ByteCount, 'PacketCount':PacketCount})
```

```
                print 'File written successfully'


        def WriteToDataBase():
                '''
                        This Function for write data set to DataBase
                        .....
                '''
                pass
        def ToFrontend():
                '''
                        This Function for formated data specialy for fronted. Send on request from view.py
(qjuery)
                        .....
                '''
        def __del__(self):
                print "deling", self

class CreateDataSetsControlPlane(object):
        def __init__(self, FlowTable):
                super(CreateDataSetsControlPlane, self).__init__()
                self.FlowTable = FlowTable

        def CreateDataSet(self):
                '''
                This function for create Data Set.
                Input parameters are:
                        - summarize counters of group's Flows in one time set. Group of Flows were
created with filtered method in class 'PrepareData'
                '''
                InputFlowTable = self.FlowTable
                #ByteCount = []; PacketCount = [];
                ByteCount = 0
                PacketCount = 0
                i = 0
                while i < len(InputFlowTable):
                        try:
                                ByteCount += int(InputFlowTable[i][11])
                                PacketCount += int(InputFlowTable[i][10])
                                i = i + 1
                        except IndexError:
                                break

                return [ByteCount], [PacketCount]
        def calculation(self):
                ByteCount   = []
                PacketCount = []
                InputData   = self.CreateDataSet()
                ByteCount   = InputData[0]; print str(ByteCount) + "\n"
                PacketCount = InputData[1]; print str(PacketCount) + "\n"
                global newDataByteControlPlane
                global GlobaldataByteControlPlane
                global newDataPacketControlPlane
```

```
                        global GlobaldataPacketControlPlane
                        if (len(GlobaldataByteControlPlane) == len(ByteCount)):
                                try:
                                        print 'start calculation for control plane'
                                        newDataByteControlPlane   = int(ByteCount[0]) -
int(GlobaldataByteControlPlane[0])
                                        newDataPacketControlPlane = int(PacketCount[0]) -
int(GlobaldataPacketControlPlane[0])
                                        GlobaldataByteControlPlane[0] = int(ByteCount[0])
                                        GlobaldataPacketControlPlane[0] = int(PacketCount[0])
                                except IndexError:
                                                print 'Select another Flow, no data'

                        elif (len(GlobaldataByteControlPlane) != len(ByteCount)):
                                GlobaldataByteControlPlane = []
                                GlobaldataPacketControlPlane = []
                                GlobaldataPacketControlPlane.append(0)
                                GlobaldataByteControlPlane.append(0)
                                newDataPacketControlPlane = GlobaldataPacketControlPlane
                                newDataByteControlPlane = GlobaldataByteControlPlane

                        return str(newDataByteControlPlane), str(newDataPacketControlPlane)

                def CreateCSV(self):
                        '''
                        InputData       = self.calculation()
                        ByteCount       = InputData[0]
                        PacketCount = InputData[1]

                        with open('app/analytics/DataSets/DataSet.csv', mode='a') as csv_file:
                                fieldnames = ['TypeOfTraffic', 'TimeStamp', 'ByteCount', 'PacketCount']
                                writer = csv.DictWriter(csv_file, fieldnames=fieldnames)

                                writer.writerow({'TypeOfTraffic': 'iot', 'TimeStamp': '1', 'ByteCount': ByteCount,
'PacketCount':PacketCount})
```

Приложение Б. ЛИСТИНГ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МНОГОПАРАМЕТРИЧЕСКОГО КОРРЕЛЯЦИОННОГО АНАЛИЗА

```
        #____coding: utf8___

        import csv
        import sys
        import math  as m
        from numpy import var, std, mean, arange
        import matplotlib.pyplot as plt; plt.rcdefaults()

        def FileCSVProcessing():
```

```
        #This Function Read new File and on the next stage formated to Internal Format for next stage
- processing
        filePath = 'DataSets/DataSetControlPlaneMAN.csv'
        fileData = open(filePath, 'rb') #open csv file
        reader = csv.reader(fileData, quoting=csv.QUOTE_NONNUMERIC, delimiter=',', quotechar
='"')
        #headers = reader.next() #skip a headers
        #print(headers)
        A1 = []; A2 = []; A3 = []; A4 = []; A5 = []; A6 = []; # There are collumns of Matrix

        for row in reader:
                A1.append(row[0]);  # write data from the 1'st collumn "Type of Traffic"
                A2.append(row[1]);  # write data from the 2'd collumn "TimeStamp"
                A3.append(row[2]);  # write data from the 3'd collumn "ByteCount"
                A4.append(row[3]);  # write data from the 4'th collumn "PacketCount"
                A5.append(row[4]);  # write data from the 5'd collumn "CPU parameter of controller
hardware"
                A6.append(row[5]);  # write data from the 5'd collumn "RAM parameter of controller
hardware"

        fileData.close() #clousing file


        return A3[:100], A4[:100], A5[:100], A6[:100]

      def MatrixTransform(InputMatrix):
        ArrayMatExp = []        # The array of mathematical expection after settlement
        ArrayMatDev = []        # The array of  mathematical deviation after settlement
        OutputMatrixStandart = [] # The output array which are transformated to Standart Type

        if len(InputMatrix[0])!=0:
                # Check Matrix Size
                for j in xrange(0, 3):
                        MatExpNumerator = 0
                        MatExp = mean(InputMatrix[j])
                        ArrayMatExp.append(MatExp)
        else:
                # Write a algorithm, which will align matrix
                print "Some troubles with matrix, hasn't data"

        # Calculate mathematical deviation
        if len(ArrayMatExp)!=0:
                # Check Matrix Size
                for j in xrange(0, 3):
                        MatVarNumerator = 0
                        MatVar = std(InputMatrix[j])
                        ArrayMatDev.append(MatVar)
        else:
                # Write a algorithm, which will align matrix
                print "Some troubles with matrix, hasn't mathematical expection"
        # Assessment of the elements of the matrix and reduction to the standard form
        OutputMatrixStandart = InputMatrix
        for j in xrange(0, 3):
```

```
                ## u_ij - new elements of standart type matrix "U" (based on theory)
                u_ij = 0
                for i in xrange(len(InputMatrix[0])):
                        u_ij = (InputMatrix[j][i] - ArrayMatExp[j])/ArrayMatDev[j]
                        OutputMatrixStandart[j][i] = u_ij

        return OutputMatrixStandart

    def CorrelationMomentFunction(InputMatrix):
      #Multuparameter correlation analyze based on Standart matrix type and returned correlation
Moment (KSI)
        ArrayMatExp = []        # The array of mathematical expection after settlement
        ArrayKSI = []           # The array of correlation Moment (KSI) between thirst (A) parameter
and other parameters

        if len(InputMatrix[0])!=0:
                # Check Matrix Size
                for j in xrange(len(InputMatrix)):
                        MatExpNumerator = 0
                        MatExp = mean(InputMatrix[j])
                        ArrayMatExp.append(MatExp)
        else:
                # Write a algorithm, which will align matrix
                print "Some troubles with matrix, hasn't data"

        for j in xrange(len(ArrayMatExp)):
                ## ksi_ik - parameter KSI (correlation Moment) between (J:K)-> [0:1; 0:2; 0:3; 1:3]
                ## [0-ByteCount, 1-PacketCount, 2-CPU, 3-RAM]
                ksi_ik = 1
                #print str(j) + "\n"
                for i in xrange(len(InputMatrix[0])):
                        if j < 3 :
                                ksi_ik += ((InputMatrix[0][i]-ArrayMatExp[0])*(InputMatrix[j+1][i]-
ArrayMatExp[j+1]))
                                #print str(ksi_ik)
                        else:
                                ksi_ik += ((InputMatrix[1][i]-ArrayMatExp[1])*(InputMatrix[2][i]-
ArrayMatExp[2]))
                ArrayKSI.append(ksi_ik/len(InputMatrix[0]))
        return ArrayKSI

    def CoeffCorrelationFunction(StandartMatrix):
      ArrayRU = [] # The array of correlation coefficient (RU) between thirst (A) parameter and
other parameters

        for j in xrange(len(StandartMatrix)):
                ## ru_ik - parameter KSI (correlation Moment) between (J:K)-> [0:1; 0:2; 0:3; 1:3]
                ## [0-ByteCount, 1-PacketCount, 2-CPU, 3-RAM]
                ru_ik = 0
                for i in xrange(len(StandartMatrix[0])):
                        if j < 3 :
                                ru_ik += StandartMatrix[0][i]*StandartMatrix[j+1][i]
                                #print str(ru_ik)
```

```
                        else:
                                ru_ik += StandartMatrix[1][i]*StandartMatrix[2][i]

                ArrayRU.append(ru_ik/len(StandartMatrix[0]))

        return ArrayRU

    def DrawPlotFunctionCoeff(InputMatrix):
        print str(InputMatrix)
        #objects = ('A / X_r', 'A / K_id', 'A / Pr_sl', 'A / m_b', 'Pr_sl / m_b')
        objects = ('ByteCount./\nPacketCount', 'ByteCount./\n/ CPU Value.', 'ByteCount./\nRAM
Value.', 'PacketCount./\nRamValue.')
        y_pos = arange(len(objects))
        performance = InputMatrix
        plt.bar(y_pos, performance, align='center', alpha=0.5)
        plt.xticks(y_pos, objects)
        plt.ylabel('Correlation Coeff.')
        plt.title('Correlation Coeff. between research parameters')
        plt.grid(True)
        plt.show()
    def DrawScatterPlot(StandartMatrix):
        import numpy as np
        import matplotlib.pyplot as plt
        #np.random.seed(19680801)
    # N = 50
        x = arange(0, len(StandartMatrix[0]), 1)
        A =     StandartMatrix[0]
        Xr =    StandartMatrix[1]
        K_id =  StandartMatrix[2]
        Pr_sl = StandartMatrix[3]
        m_b =   StandartMatrix[4]

    '''
        #plt.subplot(321)
        #plt.scatter(x, y, s=80, edgecolors='none', alpha=0.5)

        #plt.subplot(322)
        #plt.scatter(x, y, s=80, c='r', edgecolors='red', alpha=0.5)
        fig = plt.figure()
        plt.title("The Distributions")

        ax1 = fig.add_subplot(231)
        ax1.scatter(x, A, s=80, c='b', edgecolors='none', alpha=0.5, label='A')
        ax1.scatter(x, Xr, s=80, c='r', edgecolors='gray', alpha = 0.5, label='Xr')
        ax1.grid(True)
        plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})

        ax2 = fig.add_subplot(232)
        ax2.scatter(x, A, s=80, c='b', edgecolors='none', alpha = 0.5, label='A')
        ax2.scatter(x, K_id, s=80, c='purple', edgecolors='gray', alpha = 0.5, label='K_id')
        ax2.grid(True)
```

```
          plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})

          ax3 = fig.add_subplot(233)
          ax3.scatter(x, A, s=80, c='b', edgecolors='none', alpha = 0.5, label = 'A')
          ax3.scatter(x, Pr_sl, s=80, c='g', edgecolors = 'gray', alpha = 0.5, label = 'Pr_sl')
          ax3.grid(True)
          plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})

          ax4 = fig.add_subplot(234)
          ax4.scatter(x, A, s=80, c='b', edgecolors='none', alpha = 0.5, label = 'A')
          ax4.scatter(x, m_b, s=80, c='y', edgecolors='gray', alpha=0.5, label = 'm_b')
          ax4.grid(True)
          plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})

          ax5 = fig.add_subplot(235)
          ax5.scatter(x, Pr_sl, s=80, c='b', edgecolors='none', alpha = 0.5, label = 'Pr_sl')
          ax5.scatter(x, m_b, s=80, c='black', edgecolors='gray', alpha=0.5, label='m_b')
          ax5.grid(True)
          plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})
          plt.show()
        '''

        def DrawScatterPlot(StandartMatrix):
          import numpy as np
          import matplotlib.pyplot as plt
          #np.random.seed(19680801)
                  #N = 50
          #[0-ByteCount, 1-PacketCount, 2-CPU, 3-RAM]
          x = arange(0, len(StandartMatrix[0]), 1)
          ByteCount =     StandartMatrix[0]
          PacketCount =    StandartMatrix[1]
          CPU =  StandartMatrix[2]
          RAM = StandartMatrix[3]
          fig = plt.figure()
          plt.title("The Distributions")
          ax1 = fig.add_subplot(221)
          ax1.scatter(x, ByteCount, s=80, c='b', edgecolors='none', alpha=0.5, label='ByteCount')
          ax1.scatter(x, PacketCount, s=80, c='r', edgecolors='gray', alpha = 0.5, label='PacketCount')
          ax1.grid(True)
          plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})
          ax2 = fig.add_subplot(222)
          ax2.scatter(x, ByteCount, s=80, c='b', edgecolors='none', alpha = 0.5, label='ByteCount')
          ax2.scatter(x, CPU, s=80, c='purple', edgecolors='gray', alpha = 0.5, label='CPU Value')
          ax2.grid(True)
          plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})
          ax3 = fig.add_subplot(224)
          ax3.scatter(x, ByteCount, s=80, c='b', edgecolors='none', alpha = 0.5, label = 'ByteCount')
```

```
            ax3.scatter(x, RAM, s=80, c='g', edgecolors = 'gray', alpha = 0.5, label = 'RAM Value')
            ax3.grid(True)
            plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})
            ax4 = fig.add_subplot(223)
            ax4.scatter(x, PacketCount, s=80, c='b', edgecolors='none', alpha = 0.5, label = 'PacketCount')
            ax4.scatter(x, RAM, s=80, c='y', edgecolors='gray', alpha=0.5, label = 'RAM Value')
            ax4.grid(True)
            plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05), fancybox=True, shadow=True,
ncol=5,  prop={'size': 17})
            plt.show()
        print str(CorrelationMomentFunction(FileCSVProcessing()))
        DrawPlotFunctionCoeff(CoeffCorrelationFunction(MatrixTransform(FileCSVProcessing())))
        DrawScatterPlot(MatrixTransform(FileCSVProcessing()))
```

Приложение В. АКТЫ ВНЕДРЕНИЯ

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

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

14.07.2021 № 1340/54
на №_____ от_____

Утверждаю
И.о. проректора по научной работе
А.Г.Владыко

Акт

о внедрении научных результатов,

полученных в диссертационной работе Артема Николаевича Волкова

“Исследование и разработка методов построения инфраструктуры и предоставления услуг сетей связи на основе технологий искусственного интеллекта”

Комиссия в составе декана факультета Инфокоммуникационных сетей и систем к.т.н., доцента Д.В.Окуневой, профессора кафедры сетей связи и передачи данных д.т.н., доцента М.А.Маколкиной и заведующей лаборатории кафедры сетей связи и передачи данных О.И.Ворожейкиной составила настоящий акт в том, что научные результаты, полученные Артемом Николаевичем Волковым в диссертации "Исследование и разработка методов построения инфраструктуры и предоставления услуг сетей связи на основе технологий искусственного интеллекта”, использованы:

1. При чтении лекций и проведении практических занятий для бакалавров по дисциплине «Интернет вещей и самоорганизующиеся сети» (Рабочая Программа регистрационный номер №20.05/534-Д), разделы Программы:

– Сети связи шестого поколения,

- Сети связи 2030.

2. При чтении лекций и проведении практических занятий для бакалавров по дисциплине «Искусственный интеллект в сетях и системах связи» (Рабочая Программа регистрационный номер №405-Д), разделы Программы:

- Машинное обучение,

- Приложения искусственного интеллекта в сетях связи.

3. При чтении лекций и проведении практических занятий для магистров по дисциплине «Интернет вещей» (Рабочая Программа регистрационный номер 20.05/371-Д), разделы Программы:

- Интеллектуальные транспортные сети,

- Облачные сервисы для подключения Интернет вещей.

4. При чтении лекций и проведении практических занятий для аспирантов по дисциплине «Системы, сети и устройства телекоммуникаций» (Рабочая Программа регистрационный номер №20.05/712-Д), раздел Программы:

- Основные задачи построения и эксплуатации систем, сетей и устройств телекоммуникаций.

В указанных дисциплинах используются следующие новые научные результаты, полученные Артемом Николаевичем Волковым в диссертационной работе:

– Метод мониторинга, идентификации трафика услуг в сетях связи пятого и последующих поколений, основанный на аналитике метаданных потоков и алгоритмах Машинного обучения, позволяющий исключить внесение дополнительных задержек и изменение структуры потоков.

- Структура и метод взаимодействия туманных и граничных вычислений, основанные на алгоритмах больших данных, обеспечивающие функционирование микросервисной архитектуры с возможностью живой миграции, позволяющий уменьшить время выполнения функции микросервиса за счет рационального распределения ресурсов на величину до 70%.

- Метод прогнозирования нагрузки на контроллеры Программно-конфигурируемых сетей на основе технологий Искусственного интеллекта, использующий для оценки нагрузки анализ метаданных служебных потоков, что позволяет исключить зависимость программного обеспечения мониторинга от особенностей АПК производителя.

Кроме того, при выполнении НИР по теме “Исследование проблемных вопросов сетевой поддержки перспективных услуг сетей связи 2030, включая

телеприсутствие, и путей их решения, в том числе на основе технологий искусственного интеллекта, при подготовке отраслевых кадров" как составной части прикладного научного исследования СПбГУТ, использован следующий новый научный результат диссертации Артема Николаевича Волкова "Исследование и разработка методов построения инфраструктуры и предоставления услуг сетей связи на основе технологий искусственного интеллекта":

- Структура и метод взаимодействия туманных и граничных вычислений, основанные на алгоритмах больших данных, обеспечивающие функционирование микросервисной архитектуры с возможностью живой миграции, позволяющий уменьшить время выполнения функции микросервиса за счет рационального распределения ресурсов на величину до 70%.

Декан факультета ИКСС,
к.т.н., доцент Д.В.Окунева

Профессор кафедры ССиПД,
д.т.н., доцент М.А.Маколкина

Заведующая лабораторией
кафедры ССиПД О.И.Ворожейкина

Экз. 1 из 3

МИНИСТЕРСТВО
ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ
И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
УНИТАРНОЕ ПРЕДПРИЯТИЕ
«Ордена Трудового Красного Знамени
Российский научно-исследовательский
институт радио имени М.И. Кривошеева»
(ФГУП НИИР)
Казакова ул., д. 16, Москва, 105064
Телефон: (495) 647-18-30, для справок: (499) 261-63-70,
Факс: (499) 261-00-90, E-mail: info@niir.ru
http://www.niir.ru
ОКПО 01181481, ОГРН 1027700120766
ИНН/КПП 7709025230/770901001

_____№_____

«УТВЕРЖДАЮ»

Первый заместитель генерального директора ФГУП НИИР, кандидат технических наук

_____ М.Ю. Сподобаев

«__» _____ 2021 г.

АКТ

внедрения результатов диссертационной работы Волкова Артёма Николаевича на тему «Исследование и разработка методов построения инфраструктуры и предоставления услуг сетей связи на основе технологий Искусственного Интеллекта», представленной на соискание ученой степени кандидата технических наук по специальности 2.2.15– Системы, сети и устройства телекоммуникаций

Комиссия в составе:

Председатель комиссии – директор НТЦ Анализа ЭМС В.Э. Веерпалу, д.т.н.;

Члены комиссии – зам. директора НТЦ Анализа ЭМС С.Ю. Пастух, к.т.н.;
начальник отдела НТЦ Анализа ЭМС Н.В. Варламов,

установила, что в диссертационной работе Волкова Артёма Николаевича на тему «Исследование и разработка методов построения инфраструктуры и предоставления услуг сетей связи на основе технологий Искусственного Интеллекта» получены новые научные результаты, которые внедрены в 2019 – 2021 гг. в рамках выполнения государственных контрактов по научно-техническому и методическому обеспечению выполнения Министерством цифрового развития, связи и массовых коммуникаций функций администрации связи Российской Федерации в части, касающейся международно-правовой защиты интересов Российской Федерации в области электросвязи и радиосвязи в виде предложений (вкладов) проектов стандартов от имени администрации связи Российской Федерации (Министерства цифрового развития, связи

и массовых коммуникаций Российской Федерации) в Исследовательские комиссии (ИК) Сектора стандартизации электросвязи Международного союза электросвязи (МСЭ-Т).

Эти вклады представлены на заседаниях Исследовательской комиссии 13 МСЭ-Т «Будущие сети, включая облачные технологии» (ИК13 МСЭ-Т), Исследовательской комиссии 11 МСЭ-Т «Требования к сигнализации, протоколы, спецификации тестирования и борьба с контрафактной продукцией» (ИК11 МСЭ-Т) и, Исследовательской комиссии 16 МСЭ-Т «Мультимедиа» (ИК16 МСЭ-Т):

Вклады на 13 ИК МСЭ-Т по проекту Рекомендации МСЭ-Т Y.IMT-2020_mAI: “Traffic typing IMT-2020 management based on an artificial intelligent approach” (март 2019 г., июль 2020 г., март 2021 г.).

Вклад на 11 ИК МСЭ-Т по: проекту Рекомендации МСЭ-Т Q.3963 "Compatibility testing of software-defined networking (SDN)-based equipment using the OpenFlow protocol" (апрель 2020 г.).

Вклад на 16 ИК МСЭ-Т по проекту Рекомендации МСЭ-Т F.DMVSF-Edge "Distributed Multimedia Vehicular Services Framework for V2X based Edge computing" (апрель 2021 г.).

Рекомендация МСЭ-Т Q.3963 прошла процедуру утверждения и в настоящее время является действующей Рекомендацией МСЭ-Т, принятие двух остальных рекомендаций планируется в 2021 году – 2022 годах.

Председатель комиссии В.Э. Веерпалу

Члены комиссии С.Ю. Пастух

Н.В. Варламов

Подписи В.Э. Веерпалу, С.Ю. Пастуха, Н.В. Варламова заверяю.

Начальник отдела кадров ФГУП НИИР Е.П. Буянова