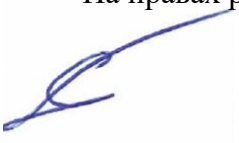
Федеральное государственное бюджетное учреждение науки Санкт-Петербургский институт информатики и автоматизации Российской академии наук

На правах рукописи



САВЕЛЬЕВ
Антон Игоревич

АРХИТЕКТУРЫ, АЛГОРИТМЫ И ПРОГРАММНЫЕ СРЕДСТВА ОБРАБОТКИ ПОТОКОВ МНОГОМОДАЛЬНЫХ ДАННЫХ В ПИРИНГОВЫХ ВЕБ-ПРИЛОЖЕНИЯХ ВИДЕОКОНФЕРЕНЦСВЯЗИ

Специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:

д.т.н., профессор Ронжин А.Л.

Санкт-Петербург – 2016

Содержание

Введение

Актуальность темы диссертации. При разработке систем видеоконференцсвязи основное внимание уделяется методам мультимедийной обработки данных, способам передачи данных, оптимизации архитектуры соединения клиентских приложений. Реализация систем видеоконференцсвязи в мобильных устройствах накладывает свои ограничения на качество и объем обрабатываемой информации вследствие недостаточных вычислительных и сетевых встроенных ресурсов мобильных гетерогенных устройств.

Недостаточная пропускная способность каналов связи для передачи стремительно растущих объемов передаваемых мультимедийных данных требует разработки новых методов их передачи. В многопользовательских системах видеоконференцсвязи между всеми участниками сеанса связи необходимо одновременно передавать несколько идентичных мультимедийных данных, что существенно увеличивает объем потоков и нагрузку на сервер.

Помимо перечисленных недостатков существует проблема встраиваемости приложений для использования их в более крупных системах. Данная проблема обусловлена тем, что все существующие приложения являются законченным программным продуктом и не располагают функционалом для интеграции в сторонние системы.

Поэтому актуальность разработки архитектур, алгоритмов и программных средств автоматической обработки мультимедийных потоков данных в пиринговых (peer-to-peer) веб-приложениях видеоконференцсвязи, обеспечивающих сокращение объема передаваемых данных и возможность построения речевых и многомодальных интерфейсов для инфокоммуникационных приложений, подтверждается отсутствием кроссплатформенного программно-аппаратного обеспечения гетерогенных клиентских приложений, поддерживающих многоканальную коммуникацию удаленных абонентов.

Степень разработанности темы. Проблеме анализа и синтеза цифровых сетей с интеграцией служб, обеспечивающих перенос в сессии различных типов информации единым образом в общей физической среде, посвящены многие научные и практические исследования отечественных и зарубежных ученых, которые проводятся с конца 60-х гг. прошлого столетия (Амосов А.А., Гольдштейн Б.С., Захаров Г.П., Ершов В.А., Кутузов О.И., Кучерявый А.Е., Голд Б., Джитман И., Франк Х., Клейнрок Л, и др.). Большой вклад в совершенствование инфокоммуникационных технологий внесли Полонников Р.И., Александров В.В., Гонсалес Р., Вуддс Р. И др. Проблемами совершенствования методов и алгоритмов маршрутизации в вычислительных сетях занимались такие ученые, как Д. Бертсекас, Д. Гарсиа-Диас, П. Гупта, А.Б. Гольдштейн, Б.С. Гольдштейн, Д. Кантор, О.Я. Кравец, Д.В. Куракин, И.П. Норенков, А. Филипс, С. Флойд и др.

Цель и задачи исследования. Основной целью диссертационной работы является разработка архитектур, алгоритмов и программных средств автоматической обработки потоков данных в пиринговых веб-приложениях видеоконференцсвязи, обеспечивающих сокращение объема передаваемых данных и снижение потребляемых ресурсов сервера и распределенных гетерогенных клиентских устройств при многоканальной обработке многомодальных потоков данных во время сеансов связи. Для достижения указанной цели в работе сформулированы и решены следующие задачи:

1. Анализ современных архитектур систем видеоконференцсвязи, а также методов и программных средств многоканальной обработки потоков аудиовизуальных и служебных потоков данных.
2. Разработка архитектур клиентской и серверной частей в пиринговых системах видеоконференцсвязи, обеспечивающих сокращение объема передаваемых данных и снижение потребляемых ресурсов сервера и клиентских приложений.
3. Разработка алгоритмов взаимодействия клиентской и серверной частей системы видеоконференцсвязи и установления соединений между

клиентами, обеспечивающих распределение и обработку их потоков данных на клиентах.

4. Разработка программных средств клиентской и серверной части веб-приложения видеоконференцсвязи, обеспечивающих кроссплатформенность и многоканальные сеансы связи между гетерогенными устройствами распределенных абонентов.
5. Разработка методики тестирования пиринговых систем видеоконференцсвязи, включающей алгоритмы функционального тестирования и набор тестов, оценивающих потребляемые ресурсы.
6. Разработка приложения видеоконференцсвязи, его верификация и тестирование по предложенной методике и сравнение с существующими аналогами.

Научная новизна:

1. Разработаны архитектуры клиентской и серверной частей в пиринговых многопользовательских системах видеоконференцсвязи, отличающиеся применением «безсерверного» принципа коммуникации клиентов, обеспечивающие сокращение объема передаваемых потоков многомодальных данных с учетом ограниченных вычислительных и сетевых встроенных ресурсов гетерогенных устройств абонентов.

2. Разработаны алгоритмы взаимодействия клиентской и серверной частей системы видеоконференцсвязи, отличающиеся поддержкой взаимодействия групп клиентов по протоколу WebRTC, распределением передаваемых мультимедийных данных по категориям, обработкой служебных данных на сервере.

3. Разработана архитектура программных средств клиентской и серверной части веб-приложения видеоконференцсвязи, обеспечивающая кроссплатформенность за счет реализации веб-интерфейса с использованием языков HTML, CSS и JavaScript, а также сокращение объема передаваемых мультимедийных данных по протоколу WebRTC, за счет программной

реализации «безсерверных» соединений клиентских частей приложения в процессе сеанса связи.

4. Разработана методика тестирования пиринговых систем видеоконференцсвязи, включающая алгоритмы функционального тестирования и набор тестов, определить потребляемые ресурсы, и позволяющая улучшить эргономику пользовательского интерфейса, снизить объем передаваемых данных, контролировать использование оперативной памяти, а также проводить сравнение с аналогичными системами видеоконференцсвязи.

Теоретическая и практическая значимость работы

Разработанные архитектуры, алгоритмы и программные средства предназначены для обработки мультимедийной информации между удаленными участниками видеоконференцсвязи напрямую, без использования промежуточного сервера. Кроссплатформенные веб-технологии позволяют адаптировать работу приложения под различные программно-аппаратные платформы и обеспечивать контроль над обычными и управляемыми аккаунтами в единообразной адаптируемой среде в режиме реального времени. Возможность встраивания разработанного приложения позволит организовывать сложные программно-архитектурные решения, которые обеспечат его работу в составе облачных систем. Сама система видеоконференцсвязи содержит модуль для создания контролируемых аккаунтов и принудительного соединения с ними. Данная особенность позволяет организовывать взаимодействие с пользователями, когда требуется построить систему временного удаленного аудиовизуального взаимодействия с принудительным соединением с абонентом. Управляемые аккаунты могут быть использованы в системах контроля и управления доступом, для обеспечения контроля за рабочими местами, производством и другими объектами, требующими периодического аудиовизуального контакта. Эффективное отображение и передача аудиовизуальной информации, набор дополнительных модулей и возможность встраивания в другие программные решения - одни из главных достоинств пиринговых веб-приложений видеоконференцсвязи,

поэтому разрабатываемые технологии будут востребованы государственными министерствами и ведомствами, крупными корпорациями, общественно-политическими организациями, активно использующими удаленное аудиовизуальное взаимодействие.

Методология и методы исследования. Для решения поставленных задач в работе использовались методы теории передачи данных, методы цифровой обработки сигналов, теории информации, теории множеств, теории объектно-ориентированного проектирования и программирования.

Степень достоверности и апробация результатов. Достоверность научных положений, основных выводов и результатов диссертации обеспечивается за счет анализа состояния исследований в данной области, согласованности теоретических выводов с результатами экспериментальной проверки алгоритмов, а также апробацией основных теоретических положений диссертации в печатных трудах и докладах на международных научных специализированных конференциях.

Исследования, отраженные в диссертации, проведены в рамках прикладных и поисковых научно-исследовательских работ: соглашение с Министерством образования и науки РФ в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014-2020 годы» (проект «Исследование и разработка системы аудиовизуального распознавания речи на базе микрофона и высокоскоростной видеокамеры», соглашение № 14.616.21.0056) в 2015-2016 гг.; грант РФФИ № 15-07-06774 «Разработка методов обработки и обмена мультимедийными данными в пиринговых веб-приложениях многоточечной видеоконференцсвязи» в 2015-2017 гг.; грант РФФИ № 13-08-00741 «Разработка методов и кроссплатформенных программных средств аудиовизуального сопровождения мобильных мероприятий» в 2013-2015 гг.; Грант Президента РФ № МД-3035.2015.8 «Разработка математического и программного обеспечения многомодальной ассистивной технологии для помощи людям с ограниченными возможностями здоровья» в 2015-2016 гг.

Разработанные архитектуры, алгоритмы, программные средства были также использованы при выполнении поисковой НИР Результаты диссертационного исследования представлялись на международной конференции NEW2AN/ruSMART (Санкт-Петербург, 2012, 2014), международной конференции Pattern Recognition and Image Analysis: New Information Technologies, PRIA-2013 (Самара, 2013); международной конференции SPECOM (Греция, 2015; Венгрия 2016), международной конференции ICR 2016 (Венгрия, 2016), IEEE международной конференции ICARSC 2016 (Португалия, 2016), XXIX международной научной конференции «Математические методы в технике и технологиях (ММТТ-29)» (Санкт-Петербург, 2016).

Публикации. По материалам диссертации опубликовано 30 печатных работ, включая 6 публикаций в научных журналах, рекомендованных ВАК, 9 публикаций в изданиях, индексируемых в WoS/Scopus.

Личный вклад автора. Теоретические выводы и практические решения, результаты тестирования. Основные научные положения сформулированы и изложены автором самостоятельно.

Структура и объем работы. Диссертация объемом 136 машинописных страниц содержит введение, четыре главы и заключение, список литературы (116 наименований), 8 таблиц, 37 рисунков, 1 приложение с копиями актов внедрения.

Краткое содержание работы

Во введении обоснована важность и актуальность темы диссертации, сформулированы цели диссертационной работы и решаемые задачи, определена научная новизна работы и ее практическая значимость.

В первой главе диссертации проведен анализ и описаны основные проблемы, возникающие при разработке систем видеоконференцсвязи. На текущий момент существует широкий спектр приложений видеоконференцсвязи широко используемых в различных сферах жизнедеятельности человека. Приложения видеоконференцсвязи, работающие

на настольных компьютерах, в большинстве случаев располагают необходимыми ресурсами для обработки данных, но при большом количестве входящих и исходящих потоков возможны проблемы из-за перегрузки центрального процессора, оперативной памяти и видеокарты устройства. На мобильных устройствах ситуацию усугубляет отсутствие ресурсов необходимых для обработки больших объемов данных и маленькие дисплеи, не способные корректно отображать более двух участников одновременно. Специализированные системы совместной работы помимо передачи аудиовизуальных потоков данных, захватываемых микрофонами и видеокамерами, обеспечивают обмен дополнительной мультимедийной информацией, например, передачу текущего слайда презентации, а также возможность совместного редактирования документов и рукописных набросков.

При анализе архитектур, возможностей и количества пользователей существующих систем видеоконференцсвязи наиболее перспективными были выбраны «безсерверные» пиринговые системы, обеспечивающие сокращение объема передаваемых данных и возможность построения речевых и многомодальных интерфейсов для данного типа телекоммуникационных приложений.

Вторая глава посвящена разработанным архитектурам и алгоритмам приложения видеоконференцсвязи. В разработанном приложении видеоконференцсвязи пиринговым узлом является клиентская часть. Архитектура клиентской части пирингового приложения видеоконференцсвязи содержит три главные составляющие: модуль логики приложения видеоконференцсвязи, модуль графического интерфейса и хранилище состояний. Для обеспечения хранения промежуточных данных, данных пользователей и соединения с другими облачными модулями в разработанной архитектуре используется серверная часть. Для обеспечения авторизации и обмена сигнальными данными в серверной части приложения используется модуль взаимодействия с клиентами. Большая часть работы модуля описана

алгоритмом передачи данных между сервером и клиентом. Алгоритм работает как в серверной, так и в клиентской частях приложения, обеспечивая одинаковый интерфейс передачи данных для обеих частей. Для объединения серверной и клиентской частей приложения была разработана архитектура взаимодействия модулей веб-приложения видеоконференцсвязи

В третьей главе описаны разработанные программные средства приложения видеоконференцсвязи. Разработка любой части приложения начинается с создания его архитектуры и основных алгоритмов. Следующим этапом разработки выступает более подробное описание системы, например, при помощи UML диаграмм. Для представления системы в данной работе были использованы несколько типов UML диаграмм: диаграммы прецедентов и диаграммы классов.

В четвертой главе представлены результаты экспериментальной проверки разработанного приложения видеоконференцсвязи. Тестирование разработанного приложения — это комплекс процедур, который охватывает различные составляющие приложения. Основными направлениями тестирования приложения являются: нагрузочное тестирование, тестирование логики работы и корректности отклика графических элементов приложения. Для тестирования графического интерфейса и логики работы приложения было использовано функциональное тестирование.

Положения, выносимые на защиту

1. Разработанные архитектуры клиентской и серверной частей в пиринговых многопользовательских системах видеоконференцсвязи обеспечивают сокращение объема передаваемых потоков многомодальных данных с учетом ограниченных вычислительных и сетевых встроенных ресурсов гетерогенных абонентских устройств.

2. Алгоритмы взаимодействия клиентской и серверной частей системы видеоконференцсвязи и установления пирингового соединения между абонентами обеспечивают поддержку взаимодействия групп абонентов, распределение и обработку мультимедийных потоков данных, передачу служебных данных для пирингового соединения.

3. Программные средства клиентской и серверной части веб-приложения видеоконференцсвязи обеспечивают кроссплатформенность за счет реализации веб-интерфейса с использованием языков HTML, CSS и JavaScript, а также сокращение передаваемых потоков многомодальных данных по протоколам WebRTC между клиентами.

4. Методика тестирования пиринговых систем видеоконференцсвязи, включающая алгоритмы функционального тестирования и набор тестов, позволяет оценить потребляемые ресурсы и эргономику пользовательского интерфейса, снизить объем передаваемых данных, контролировать использованием оперативной памяти, а также проводить сравнение систем видеоконференцсвязи друг с другом.

Глава 1. Анализ методов и программных средств систем видеоконференцсвязи

Приложения видеоконференцсвязи, работающие на настольных компьютерах, в большинстве случаев располагают необходимыми ресурсами для обработки данных, но при большом количестве входящих и исходящих потоков возможны проблемы из-за перегрузки центрального процессора, оперативной памяти и видеокарты устройства. На мобильных устройствах ситуацию усугубляет отсутствие ресурсов необходимых для обработки больших объемов данных и маленькие дисплеи, не способные корректно отображать более двух участников одновременно. Специализированные системы совместной работы помимо передачи аудиовизуальных потоков данных, захватываемых микрофонами и видеокамерами, обеспечивают обмен дополнительной мультимедийной информацией, например, передачу текущего слайда презентации, а также возможность совместного редактирования документов и рукописных набросков.

Эти и другие проблемы проектирования систем видеоконференцсвязи рассмотрены в первой главе. При анализе архитектур, возможностей и количества пользователей существующих систем видеоконференцсвязи наиболее перспективными были выбраны пиринговые системы, обеспечивающие сокращение объема передаваемых данных и возможность построения речевых и многомодальных интерфейсов.

1.1. Анализ методов аудиовизуальной обработки в системах видеоконференцсвязи

Разработка мобильных приложений с функциями автоматической обработки речи в режиме онлайн, в том числе распознавания и синтеза речи, диаризации дикторов, определения биометрических характеристик, и их внедрение набирают все большую популярность в наше время [1]. Технологии видеоконференцсвязи находят широкое применение в различных сферах

человеческой деятельности, таких как: обучение [2, 3], системы ухода за пожилыми людьми и людьми с ограниченными возможностями [4, 5], коллаборация [6], анализ активности пользователей [7 - 9] и т.д. С увеличением возможностей вычислительных и сетевых ресурсов предъявляются новые требования к приложениям, обеспечивающим связь между удаленными пользователями [10, 11]. Если изначально люди довольствовались общением в текстовых чатах, то на данный момент широко используются средства, использующие аудио- и видео данные для обмена информацией между пользователями в реальном времени. В работе [12] рассматриваются перспективы использования интернет-телевидения, а также дается краткая характеристика оптимизации нагрузки на каналы связи. В работах [13, 14] изучаются способы передачи видео- и аудиосообщений в системах видеоконференцсвязи. Рассматриваются наиболее широко используемые технологии передачи видео- и аудиоданных, приводятся их преимущества и недостатки, анализируется целесообразность использования тех или иных технологий в зависимости от типа системы видеоконференцсвязи.

Характеристики различных протоколов обмена мгновенными сообщениями в режиме реального времени приводятся в [15]. Проблемы передачи данных в приложениях видеоконференцсвязи, работающих в режиме реального времени, тщательным образом исследованы в работе [16]. Подробные исследования методов и подходов к оцениванию качества передачи сигналов различных модальностей приведены в работах [17 - 19]. Рассматриваемые в статье [20] технологии распознавания образов могут быть применены для обработки видеосигналов в приложениях видеоконференцсвязи. Кроме того, этой цели могут служить разработки, описанные в работах [21, 22].

Анализ возможностей применения технологий распознавания речи в рамках более широкой задачи аутентификации пользователей на основе биометрических данных (как статических, так и динамических методов биометрии), а также их подробное рассмотрение представлены в следующих работах [23 - 29].

В настоящее время существует широкий спектр различных приложений видеоконференцсвязи [10, 30]; некоторые из них малоизвестны, другие являются крупными коммерческими проектами, которые широко используются в различных сферах жизнедеятельности человека. Потребность в приложениях видеоконференцсвязи высока и это связанно с одним из важнейших аспектов жизни человека — обменом информацией. Компании, разрабатывающие приложения видеоконференцсвязи, используют собственные методы при их проектировании и разработке, поэтому готовые приложения обладают собственными достоинствами и недостатками. Особенности каждого приложения определяются возможностью использовать его на различных платформах и устойчивость к различному виду нагрузок, связанных с передачей, обработкой и выводом данных, предназначенных для пользователей.

Увеличение числа участников одна из главных проблем всех приложений видеоконференцсвязи. С ростом количества участников возрастает нагрузка на само приложение: увеличивается количество входящих и исходящих потоков информации и возрастает число обрабатываемых данных, выводимых устройством. Приложения видеоконференцсвязи, работающие на настольных компьютерах, в большинстве случаев располагают необходимыми ресурсами для обработки данных, но при большом количестве входящих и исходящих потоков возможны проблемы из-за перегрузки центрального процессора и оперативной памяти устройства. На мобильных устройствах ситуацию усугубляет отсутствие ресурсов, необходимых для обработки больших объемов информации, и маленькие дисплеи, которые не способны корректно отображать более четырех участников одновременно. Частичное решение данных проблем возможно с помощью разработки и создания хорошо оптимизированной архитектуры приложения. Возможные пути повышения скорости передачи данных в пиринговой сети, а также алгоритмы исключения одновременного доступа пользователей к общим ресурсам в пиринговых системах рассматриваются в работах [31-35]. Применение технологии UPnP для VoIP-телефонии, а также анализ существующих решений рассмотрены в работах [36, 37].

На сегодняшний день существует множество интернет-приложений для видеоконференцсвязи и совместной работы в области обучения, бизнеса и индустрии развлечений [38, 39]. Одним из первых приложений, ориентированных на голосовую связь между пользователями персональных компьютеров, стала программа Skype, выпущенная в 2003 году [40]. Основные разработки в области обработки аудио- и видеопотоков направлены на улучшение качества, скорости и объема передаваемой информации между пользователями [41, 42]. В статьях [43 - 46] рассматривается новый специфический речевой жанр интернет-коммуникации — мессенджер, описываются его дифференциальные признаки, выделены его специфические черты, выявленные на основе его сравнения с такими жанрами интернет- и мобильного общения, как SMS, электронное письмо, чат. В [47] приводится сравнительный анализ мессенджеров. В работах [48, 49] рассматриваются проблемы безопасности популярных сервисов обмена мгновенными сообщениями и предлагаются рекомендации по выбору оптимального решения. В работе [50] представлена разработка мобильного приложения, позволяющего отправлять экстренные сообщения в чрезвычайных ситуациях.

В зависимости от назначения системы интернет связи можно разделить на: средства общения, проведения видеоконференций и совместной работы [40, 51, 52]. Средства персонального общения ориентированы на захват клиентским устройством аудиовидеоданных только одного пользователя, и чаще всего, сеанс связи организован между двумя устройствами. При проведении мероприятий (совещаний) между коллективами людей применяются системы видеоконференций, обеспечивающие многоканальную обработку аудиовизуальных сигналов на стороне каждого клиента. Это позволяет реализовать запись крупным планом каждого участника, находящего в помещении, а также оценить общий вид аудитории. Специализированные системы совместной работы помимо передачи аудиовизуальных потоков данных, захватываемых микрофонами и видеокамерами, обеспечивают обмен дополнительной мультимедийной информацией, например, передачу текущего

слайда презентации, а также возможность совместного редактирования документов и рукописных набросков. В зависимости от назначения приложения, интернет связи могут обладать следующими возможностями и инструментами:

- совместный доступ к экрану или отдельным приложениям (screen sharing);
- интерактивная доска (whiteboard);
- демонстрация презентаций;
- синхронный просмотр веб-страниц (co-browsing);
- аннотация экрана;
- мониторинг присутствия участников;
- текстовый чат;
- интегрированная VoIP-связь;
- видеоконференцсвязь;
- возможность менять ведущего;
- возможность отдавать контроль над мышью и клавиатурой;
- модерация онлайн-встреч;
- обратная связь (например, опросы или оценки);
- планирование встреч и приглашение участников;
- запись хода веб-конференции.

Другим классификационным признаком систем связи является тип клиентского устройства, по которому можно разделить системы на мобильные приложения, приложения для настольных компьютеров/ноутбуков и кроссплатформенные приложения. Среди всех систем веб-конференций можно выделить приложения известных компаний: Adobe Acrobat Connect, Google Hangouts, Microsoft Office Live Meeting.

К сожалению, лишь не многие системы веб-конференций обладают кроссплатформенностью, которая достигается за счет встраивания приложения веб-страницы. Большим минусом кроссплатформенных систем веб-конференций является то, что не все мобильные браузеры корректно

отображают информацию с веб-страниц. Кроме того, графическим интерфейсом не всегда удобно пользоваться в мобильном браузере в связи с небольшим размером экрана. Еще одним недостатком таких приложений является малое количество систем видеоконференций, имеющих клиентские части, которые поддерживают возможность кроссплатформенной работы без предустановки приложения в систему. В таблице 1.1 представлены основные свойства и возможности современных приложений видеоконференцсвязи.

Таблица 1.1. Примеры мобильных приложений интернет связи

Название	Максимальное количество участников видеочата	Количество Пользователей, млн	Возможности	Платформы
Skype	10	300	аудио/видео звонки, групповой чат, передача файлов	Mac OS, GNU/Linux, Windows, iOS, PSP, Symbian, Android , J2ME и Windows Mobile, Google Chrome, Mozilla Firefox
Nimbuzz	2	150	аудио/видео звонки, групповой чат	Android, Symbian, MIDP, Windows Phone, BlackBerry, Mac OS, iOS
Название	Максимальное количество участников видеочата	Количество Пользователей, млн	Возможности	Платформы
ooVoo	12	80	групповые аудио/видео звонки, групповой чат, совместный просмотр данных, запись видеочата	Android, iOS, Windows Phone, Kindle HD, PC and Mac
µTox, qTox, Toxic, Ratox, Blight	2-10	~2-10	групповые аудио/видео звонки, групповой чат, трансляция экрана, передача файлов	iOS, Android, Linux, Mac OS, Windows, Sailfish OS, FreeBSD,

				Solaris
iChat (iMessages)	4	Нет данных	текстовый, аудио- и видеочат, удаленный доступ к рабочему столу, удаленная демонстрация документов (ichat theater), групповые конференции	iOS, Mac OS
Fring	4	50	текстовый, аудиочат, видеочат	Symbian 8.x-9.x, Windows Mobile 5-6, iOS
OpenMCU	30	Нет данных	видео, аудио	FreeBSD, Linux
TrueConf	250	Нет данных	видео, трансляция на сайте	Windows Server 2008 или Windows Server 2012
Marratech	5	Нет данных	видео, аудио, текстовый чат	Windows, Linux
Mind	12 012	Нет данных	видео, управление правами участников, чат, показ презентаций и документов, запись мероприятия, опросы и голосования, трансляция на сайте, интеграция с аппаратными вкс, интеграции с корпоративной атс, интеграция с телефонными книгами ldap	Windows XP SP3/Vista/7 или Mac OS X 10.5 – 10.7
VideoGrace	30	Нет данных	видео, аудио	Microsoft Windows (XP, Vista, 7, 8, 8.1, 10, Server 2003, Server 2008, Server 2012), ОС Linux.

Приложения из таблицы 1.1 с максимальным числом участников (больше 12) имеют клиент-серверную архитектуру. Данная архитектура позволяет одновременно взаимодействовать большому количеству участников и проводить широковещательные видеоконференции. Преимуществом данной архитектуры является низкая нагрузка на клиентскую часть приложения, которая используется для получения и вывода данных. Основную нагрузку по обработке, распределению и передаче данных в данной архитектуре осуществляет серверная часть. Такие системы удобно использовать для

совещаний сотрудников в крупных компаниях, проведения масштабных мероприятий и других событий, где необходимо взаимодействие множества участников. Несмотря на все преимущества данной архитектуры, главным ее недостатком является серверное оборудование, стоимость которого изначально высока и растет пропорционально количеству возможных участников данного вида связи. Программное обеспечение таких систем платное и финансово доступно только крупным организациям.

Приложения из таблицы 1.1 с максимальным количеством участников до 10-ти человек имеют пиринговую архитектуру. Главное преимущество данной архитектуры — прямая передача данных между клиентскими частями приложения без использования промежуточного сервера. Такое преимущество влечет за собой недостатки в архитектуре такого типа: обработка и распределение данных происходит в клиентской части приложения, и как следствие, приложение потребляет большое количество ресурсов устройства; также необходима дополнительная система, обеспечивающая хранение информации о других пользователях. Перечисленные недостатки приводят к тому, что приложения, построенные на базе пиринговой архитектуры, позволяют осуществлять видеоконференцсвязь с небольшим количеством участников. Несмотря на все недостатки, на сегодняшний день такие системы довольно популярны среди обычных пользователей. Исходя из данных, представленных в таблице 1.1, можно выделить приложение Skype — один из лидеров рынка пиринговых приложений видеоконференцсвязи. Skype является частично бесплатным программным обеспечением с закрытым исходным кодом, обеспечивающим шифрованную голосовую и видеосвязь через Интернет между компьютерами (VoIP). Также Skype имеет платные услуги для звонков на мобильные и стационарные телефоны. Из преимуществ данного приложения стоит отметить наличие мобильной, десктопной и браузерной версий клиентских приложений и возможность одновременного видеочата для 10-ти человек. Существенным недостатком данной системы является ее проприетарность.

Проприетарность, с одной стороны, обеспечивает полную закрытость исходного кода, что позволяет повысить безопасность системы, с другой стороны, программное обеспечение не может быть использовано сторонними разработчиками для его интеграции с другими сервисами и системами. Все вышеперечисленные решения из таблицы 1.1 обладают тем же недостатком. Такая изолированность систем приводит к тому, что для обеспечения внедрения модуля, включающего взаимодействие пользователей посредством видеоконференцсвязи, разработчикам необходимо создавать собственное программное обеспечение или использовать устаревшие и не стабильные версии неподдерживаемых решений.

Для внедрения системы видеоконференцсвязи как модуля в уже существующую архитектуру приложения необходимо обеспечить простую интеграцию с другими модулями данного приложения или системы, а также разработать простой интерфейс взаимодействия с конечным пользователем приложения, не требующий дополнительной настройки и независимый от операционной системы. Для достижения данной цели, необходимо решить несколько задач: разработать архитектуру, алгоритмы, графический интерфейс и программное обеспечение приложения. Далее рассмотрим анализ методов передачи данных в пиринговых веб-приложениях видеоконференцсвязи.

1.2. Анализ методов передачи данных в пиринговых веб-приложениях видеоконференцсвязи

Несмотря на стремительные темпы развития интернет технологий, существует множество проблем, связанных с потоковой передачей видео- и аудиоинформации. Во многом эти проблемы возникают по причине недостаточной пропускной способности каналов связи. Поскольку системы видеосвязи требуют больших сетевых ресурсов даже для передачи видео между двумя участниками, поддержка многопользовательских видеоконференций является крайне затруднительной задачей.

Сейчас сотни тысяч пользователей могут одновременно использовать пиринговые сети [53, 54]. Обычной практикой в P2P системах потокового видео

является объединение участников, просматривающих один и тот же контент, в «рой» (swarm) и перераспределение частей видеоконтента исключительно между участниками этого роя. Для такой канально-изолированной структуры P2P видеосистем характерны задержки переключения каналов и отставание воспроизведения контента, связанные с оттоком абонентов канала и дисбалансом числа принимающих и ретранслирующих узлов. В целом глобальные P2P сети с канально-изолированной структурой в нынешнее время имеют серьезные проблемы с производительностью, которые будут становиться все более серьезными с ростом числа пользователей каналов.

Кроме того, в сетях потоковых систем P2P существуют проблемы, связанные с задержками каналов коммутации и низкой производительностью систем для каналов с небольшим числом участников. В работе [55] для решения этих проблем предлагается многоканальная потоковая P2P-платформа View-Upload Decoupling (VUD), разделяющая выполняемые пользователем операции загрузки и просмотра данных, а также совместного использования ресурсов перекрестных каналов, чем обеспечивается стабильность для многоканальных систем и качественное распределение ресурсов сети. Для повышения производительности передачи данных учитывается географическое местоположение клиентов сети, и связь устанавливается между наиболее близко расположенными пользователями, имеющих необходимые данные.

В работе [56] представлена архитектура многопользовательской распределенной пиринговой (peer-to-peer, P2P) системы видеоконференций, в которой предполагается, что каждый участник может создавать, отправлять и получать видеосигнал в любой момент времени, но во время видеоконференции участник передает либо видеосигнал, создаваемый его устройством, либо ретранслирует принимаемый видеосигнал другому участнику. Таким образом, участник, который посылает свой собственный видеосигнал, не может действовать в качестве промежуточного узла для прохождения видеосигнала от другого партнера. Данная распределенная архитектура может быть использована для реализации видеоконференций системы P2P, чтобы позволить

каждому участнику видеть другого участника. Архитектура данной системы видеоконференцсвязи P2P определяется на основе так называемой «цепи». Конфигурация цепи во время сеанса связи формируется на основе конфигурационных (служебных) сообщений, рассылаемых между приложениями видеоконференцсвязи участников.

Для уменьшения объема данных, передаваемых в ходе видеоконференцсвязи, в работе [57] используется автоматический способ определения текущего говорящего пользователя, и его потокам мультимедийных данных выставляется наибольший приоритет при передаче остальным участникам. Идентификация, диаризация дикторов, а также другие методы обработки речи и анализа лица человека широко применяются для автоматизации телекоммуникационных сервисов [58 - 60]. В работе [61] предложена «безсерверная» архитектура распределительного узла для передачи видео высокой четкости в многопользовательской системе видеоконференцсвязи. В основе архитектуры заложен многопользовательский управляющий блок, интегрированный в клиентское приложение, который служит для установления множественных канальных соединений, кодирует и декодирует клиентские видеопотоки и распределяет видео между другими участникам сеанса связи.

Производительность способов передачи потокового видео в P2P-сетях зависит также от конфигурации самой сети, ее топологии, неоднородности сетевых ресурсов абонентов, пропускной способности их каналов связи [62]. В отличие от совместной загрузки файлов, где малая пропускная способность канала приводит к медленной загрузке, при передаче потокового видео низкая скорость подключения становится настоящей проблемой. Также остаются актуальными вопросы компрессии видеосигнала, позволяющей снизить загруженность канала без значительного увеличения нагрузки на устройство конечного пользователя при кодировании/декодировании сигнала.

В работах [63, 64] представлены полученные авторами результаты предварительных исследований по сокращению передаваемого объема данных

между пользователями в приложениях видеоконференцсвязи. Описаны алгоритмы и программные средства, позволившие провести оптимизацию разработанного кроссплатформенного приложения видеоконференцсвязи на этапах создания и удаления аудио- и видеопотоков данных, их передачи от сервера к клиенту и обратно, создания цепочек потоков и их поиска на сервере. В ходе исследований было выполнено упрощение клиентской части приложения и реорганизация структуры серверной части приложения.

На основе предложенных принципов оптимизации способов обмена мультимедийными данными в приложении видеоконференцсвязи была разработана новая пиринговая архитектура прямой передачи аудио- и видеоданных между клиентскими частями, представленная в следующем разделе. Описан процесс формирования клиентских веб-страниц и установления связи с сервером по протоколу WebSocket. В последнем разделе обсуждаются разработанные алгоритмы установления соединений между клиентами с использованием протокола WebRTC. Современные подходы к разработке протоколов обмена сообщениями между различными многомодальными устройствами и обзор передовых сетевых технологий представлены в работах [65 - 69].

1.3. Анализ архитектур существующих систем видеоконференцсвязи

Все приложения видеоконференцсвязи по своей архитектуре делятся на два типа: клиент-серверные [70 - 72] и пиринговые (peer-to-peer) [73]. Данные архитектуры сильно отличаются способами распределения нагрузки на устройства и системой обмена данными.

Сначала рассмотрим клиент-серверную архитектуру. В ее основе лежит сервер, который выполняет основные задачи: авторизация клиентов, обработка потоков данных и их распределение между клиентами. Клиентская часть такой архитектуры способна отображать, передавать серверу и принимать потоки данных. Таким образом, нагрузка на устройство конечного пользователя снижается за счет выполнения основных операций по обработке потоков

данных на сервере. Также следует отметить перспективность применения автоматических средств анализа речи и других естественных модальностей, обеспечивающих сокращение объема передаваемых данных и возможность построения речевых и многомодальных интерфейсов для данного типа телекоммуникационных приложений [74 - 76].

Пиринговая (peer-to-peer) архитектура подразумевает обмен информацией напрямую между конечными пользователями. В случае приложений видеоконференцсвязи даже для данной архитектуры в некоторых случаях необходимо наличие сервера, служащего для авторизации и координации работы клиентов. Основные функции все же выполняются на устройстве конечного пользователя, поэтому вся нагрузка, связанная с обработкой и отображением данных, приходится на устройство конечного пользователя.

Современные приложения видеоконференцсвязи позволяют обмениваться не только аудио- и видеопотоками, но и совместно редактировать документы, рисовать схемы, графики, изображения [77]. Личный кабинет пользователя такого приложения содержит настройки и графические компоненты интерфейса, позволяющие взаимодействовать с другими пользователями.

Развитие инфотелекоммуникационных технологий позволило вывести данный тип приложений на новый этап развития и использовать пиринговые сети для передачи данных напрямую между пользователями, тем самым снижая нагрузку на серверную часть приложения и распределяя ее между клиентами [78]. Такой подход позволяет создавать небольшие видеочаты до десяти человек, где хорошее качество связи обеспечивается максимум при пяти участниках. Такие приложения направлены на обеспечение обмена данными в узком кругу участников, но они также востребованы, как и их клиент-серверные аналоги, а значит, их разработка и развитие являются перспективными и целесообразными [79, 80].

Управление числом активных участников в одном сеансе видеоконференцсвязи осуществляется функционалом клиентских приложений, настраиваемых и взаимодействующих с центральным сервером. Анализ

подходов к управлению пользовательскими аккаунтами и средств многомодальных интерфейсов, применяемых в архитектурах видеоконференцсвязи, рассмотрены ниже.

Одно из востребованных направлений в развитии модулей для приложений видеоконференцсвязи — разработка корпоративных аккаунтов. В статье [81] рассматривается управление аккаунтами пользователей корпоративной сети. Для решения задач управления аккаунтами используются несколько технологий, разработанных в Carnegie Mellon University, предназначенных для облегчения работы с пользователями и их правами в корпоративной среде. Информация обычно обрабатывается в дереве LDAP-серверов, поддерживаемых различными группами на различных административных уровнях — от корпорации до отдела. Для оптимизации информации, проходящей через уровни LDAP-серверов, проводится процедура минимизации необходимых связей между поддерживающими группами и использование механизмов кэша для повторно передаваемой информации с высших административных уровней на низшие. Эти технологии могут быть применены как на различных уровнях LDAP-серверов, так и на компьютере конечного пользователя.

Управление аккаунтами в среде гибридного облака рассматривается в работе [82]. Для решения задач контроля аккаунтов используется система управляющая облаком Monsoon. Данная система предоставляет корпоративным пользователям интерфейс и портал для управления инфраструктурой облака во множественных публичных и приватных облачных провайдерах. Она обеспечивает управление пользовательскими аккаунтами, контроль доступа, инструменты анализа и отчетов, корпоративные аккаунты/менеджер ролей и систему, осуществляющую контроль, поддерживающую пользовательские подписки и управление многоуровневыми аккаунтами в гибридном облаке, которое может содержать множество публичных и приватных облачных сервисов.

Немаловажным аспектом при разработке модулей является сама архитектура приложения. Правильно спроектированная архитектура позволяет легко разрабатывать и внедрять новые модули. В работе [83] описывается системная архитектура видеоконференции на базе рекомендации ITU-T - H.323, рассматривается как серверная, так и клиентская часть. Данная архитектура проводит обработку видео для создания эффекта личного присутствия посредством управления объектами видео и динамического распределения скоростей. В работе описан подход, основанный на цветности ключа, разрешающий извлечения объектов сервером PPMCU, так что видеообъектами можно управлять в соответствии с потребностями пользователей. После извлечения объектов сервер перекодирует их, используя интеллектуальный метод распределения битов, и вставляет обратно фон цветового ключа. Клиент извлекает видео из принятого потока и манипулирует 2D-объектами и виртуальным фоном в 3D-пространстве.

Приведем примеры известных приложений видеоконференцсвязи с различными архитектурами. В таблице 1.2 представлены основные преимущества и недостатки использования архитектур данных приложений. Skype (peer-to-peer архитектура) [84] — проприетарное программное обеспечение с закрытым кодом, обеспечивающее шифрованную голосовую связь и видеосвязь через интернет между компьютерами, а также платные услуги для звонков на мобильные и стационарные телефоны. Google Hangouts (клиент-серверная архитектура) — бесплатный сервис групповой видеосвязи в социальной сети Google+. Позволяет получать общий доступ к экрану, совместно использовать программу рисования, редактировать документы, проводить широковещательные вебинары. Adobe Connect (клиент-серверная архитектура) — система, основанная на технологии Flash, позволяет проводить онлайн встречи, презентации с использованием Power Point, совместно работать с установленными приложениями. Может поставляться как веб-сервис, не требующий установки, или как самостоятельное приложение.

Таблица 1.2. Архитектуры приложений видеоконференцсвязи

	Преимущества	**Недостатки**
Клиент-серверная	1. Отсутствие дублирования кода программы-сервера программами-клиентами. 2. Снижение требования к устройствам конечного пользователя. 3. Данные хранятся на сервере, что позволяет защитить их лучше, чем на клиенте. 4. Упрощенное управление полномочиями пользователей, подключенных к серверу.	1. Неработоспособность сервера приводит к остановке передачи любых данных. 2. Для крупных систем необходимо дорогостоящее серверное оборудование, способное обрабатывать большой объем данных.
Пиринговая (peer-to-peer)	1. Передача данных происходит непосредственно между клиентами. Даже при отключении сервера регистрации, данные продолжат передаваться. 2. Невысокие требования к серверному оборудованию.	1. Высокие требования к устройству конечного пользователя в связи с высокой нагрузкой на клиентскую часть приложения. 2. Проблемы работы на мобильных платформах при воспроизведении более четырех аудио- и видеопотоков.

Вне зависимости от типа архитектуры основными задачами приложения видеоконференцсвязи являются обработка аудио- и видеопотоков и их передача от одного пользователя к другому [85]. В таблице 1.3 представлены достоинства и недостатки приложений видеоконференцсвязи с различными архитектурами. Из вышеперечисленных приложений стоит отметить систему видеоконференцсвязи Skype. Архитектура Skype является пиринговой в отличие от клиент-серверной архитектуры Google Hangouts и Adobe Connect, что позволяет приложению экономить интернет трафик и уменьшить задержки при передаче информации от пользователя к пользователю. На сегодняшний день Skype является самым развитым и востребованным приложением

видеоконференцсвязи, поэтому сравнение результатов оптимизации мы будем проводить именно с этой программой.

Таблица 1.3. Примеры приложений видеоконференцсвязи

	Название приложения		
	Skype	**Google Hangouts**	**Adobe Connect**
Достоинства	Существуют клиентские части как для персональных компьютеров, так и для мобильных платформ. Поддержка видеоконференций до 10 абонентов. Обеспечивается передача текстовых сообщений (чат) и передача файлов. Есть возможность вместо изображения с веб-камеры передавать изображение с экрана монитора.	Бесплатные сеансы видеосвязи одновременно до 10 участников, центральный видеопоток переключается автоматически в зависимости от того, кто в данный момент громче говорит. Используются открытые технологии, поэтому есть возможность интеграции со сторонними приложениями и веб-сервисами видеосвязи.	Кроссплатформенность, можно свободно выбирать, использовать ли простой телефон, IP-телефон, видеотелефон, микрофон с веб-камерой или профессиональную конференц-систему для трансляции аудио/видео. Существуют мобильные версии для платформ Android, iOS, BlackBerry PlayBook.
Недостатки	Использование проприетарного протокола, несовместимого с открытыми стандартами (такими, как SIP или H.323), интенсивное использование антиотладочных приемов и обфусцированного кода, возможны вирусные эпидемии из-за пиринговой архитектуры.	Требуется наличие аккаунта в социальной сети Google+, несмотря на то, что работает в браузере имеет мало поддерживаемых платформ и требует установки дополнительного плагина; адаптирован всего под одну мобильную платформу - Android.	Платное использование необходимого количества лицензий на решение Adobe Connect для компании или приобретение услуги хостинга. Очень высокая стоимость, доступная только крупным организациям.

Сложная архитектура и обработка потоков мультимедийной информации пирингового-клиентской части приложения видеоконференцсвязи приводят к

существенному увеличению нагрузки на устройства конечного пользователя. Данному типу приложений необходим сервер, обеспечивающий уменьшение нагрузки на этапе соединения клиентов и хранение промежуточной информации и информации о других пользователях системы.

Сервер позволяет снизить нагрузку на клиентскую часть и обеспечить обмен данными, необходимыми для соединения. Единая база данных, расположенная на сервере, позволяет избежать множества проблем, связанных с синхронизацией и хранением общих данных пользователей на всех клиентских частях. Также общая база данных позволит повысить безопасность системы за счет централизованного хранения в одном и месте, и доступ к ним будет обеспечен только авторизированным пользователям. Промежуточный сервер также помогает решить проблему потери «сигнальных» данных, буферизируя их до необходимого момента передачи.

В следующей главе будут рассмотрены особенности алгоритмов и программных средств разработанного пирингового приложения видеоконференцсвязи и сигнального сервера, а также результаты их экспериментальной проверки.

1.4. Выводы

1. Выполненный в ходе диссертационного исследования анализ методов и программных средств мобильных систем видеоконференцсвязи выявил, что основными направлениями исследований в данной области являются методы мультимедийной обработки данных, способы передачи данных, выбор архитектуры соединения клиентов видеоконференцсвязи.

2. Среди методов мультимедийной обработки данных наибольшее внимание уделяется видео- и звуковой информации. Использование интеллектуальных методов распознавания звуковых и графических образов, биометрической идентификации позволяет существенно улучшить эргономику пользовательского интерфейса и снизить объем передаваемых данных. Реализация систем видеоконференцсвязи в мобильных устройствах

накладывает свои ограничения на качество и объем обрабатываемой информации вследствие недостаточных вычислительных и сетевых встроенных ресурсов мобильных гетерогенных устройств.

3. Недостаточная пропускная способность каналов связи для передачи стремительно растущих объемов передаваемых мультимедийных данных требует разработки новых оптимальных методов передачи данных. В многопользовательских системах видеоконференцсвязи одновременно необходимо передавать несколько идентичных мультимедийных контентов между всеми участниками сеанса связи, что нелинейно увеличивает объем передаваемых данных.

4. При анализе архитектур существующих систем видеоконференцсвязи наиболее перспективными были выбраны «безсерверные» пиринговые системы, обеспечивающих сокращение объема передаваемых данных и возможность построения речевых и многомодальных интерфейсов для данного типа телекоммуникационных приложений.

Глава 2. Архитектуры и алгоритмы взаимодействия модулей системы видеоконференцсвязи

Рассматривается проблема соединения клиентов и передачи аудио- и видеопотоков данных в пиринговых веб-приложениях видеоконференцсвязи. При взаимодействии нескольких клиентских и серверной частей приложения видеоконференцсвязи на основе протокола WebRTC возможна частичная или полная потеря «сигнальных» данных, препятствующая соединению клиентов. Предложенная архитектура передачи и хранения «сигнальных» данных на клиенте и сервере обеспечивает буферизацию и последующую обработку «сигнальных» данных, исключая их потерю и поддерживая взаимодействие между группами клиентов.

Для формального описания задачи синтеза архитектур пиринговых многопользовательских систем видеоконференцсвязи введено множество возможных типов архитектур $A = \{ A_\chi, \chi \in ST \}$, в качестве которых выделяются архитектура клиентской части $A_{\text{КЛИЕНТ}}$, архитектура серверной части $A_{\text{СЕРВЕР}}$ и архитектура обмена данными $A_{\text{ОБМЕН}}$. Для связи перечисленных множеств друг с другом введем в рассмотрение системный динамический альтернативный мультиграф следующего вида:

$$A_\chi^t = \left\langle X_\chi^t, F_\chi^t, Z_\chi^t \right\rangle, \qquad (1)$$

где χ — индекс, характеризующий тип архитектуры приложения видеоконференцсвязи, $\chi = \{1,2,3\}$ — множество индексов, соответствующих архитектуре клиентской, серверной частей и обмена данными соответственно; $t \in T$ – множество моментов времени; $X_\chi^t = \{x_{\chi l}^t, l \in L_\chi\}$ — множество элементов, входящих в состав архитектуры A_χ^t (множество вершин графа) в момент времени t; $F_\chi^t = \{f_{<\chi,l,l'>}^t, l, l' \in L_\chi\}$ — множество дуг графа типа A_χ^t, отражающих

взаимосвязи между его элементами в момент времени *t*; $Z_{\chi}^{t} = \{f_{<\chi,l,l'>}^{t}, l, l' \in L_{\chi}\}$ — множество значений параметров, количественно характеризующих взаимосвязь соответствующих элементов графа. Зададим множество допустимых операций отображения указанных выше графов друг на друга:

$$M_{<\chi,\chi'>}^{t} : F_{\chi}^{t} \to F_{\chi'}^{t}, \tag{2}$$

а также операции композиции указанных отображений:

$$M_{<\chi,\chi'>}^{t} = M_{<\chi,\chi_1>}^{t} \circ M_{<\chi_1,\chi_2>}^{t} \circ \ldots \circ M_{<\chi'',\chi'>}^{t}.$$

Тогда многоархитектурное состояние можно определить как подмножество декартова произведения множеств элементов, на которых строятся соответствующие архитектуры приложения видеоконференцсвязи:

$$S_{\delta} \subseteq X_1^{t} \times X_2^{t} \times X_3^{t}, \quad \delta = 1,\ldots,K_{\Delta}$$

Множество многоархитектурных состояний приложения видеоконференцсвязи запишется в следующим образом:

$$S = \{S_{\delta}\} = \{S_1,\ldots,S_{K_{\Delta}}\}.$$

Введем множество допустимых операций отображения многоархитектурных состояний приложения видеоконференцсвязи друг на друга:

$$\Pi_{<\delta,\delta'>}^{t} : S_{\delta} \to S_{\delta'}.$$

При этом будем полагать, что каждое многоархитектурное состояние приложения видеоконференцсвязи задаётся в результате операции композиции соответствующих графа, описывающих каждый тип архитектуры (2).

Тогда для синтеза архитектур пиринговых многопользовательских систем видеоконференцсвязи может быть предложено ее следующее *теоретико-множественное описание*: необходимо разработать модельно-алгоритмическое

обеспечение, позволяющее находить такие $\left\langle U_*^t, S_{\delta *}^{t_f} \right\rangle$, при которых выполняются следующие условия:

$$J_\theta\left(X_\chi^t, F_\chi^t, Z_\chi^t, M_{<\chi,\chi'>}^t, \Pi_{<\tilde{\delta},\tilde{\tilde{\delta}}>}^t, t \in (t_0, t_f]\right) \to \underset{<U_*^t, S_{\delta *}^{t_f}> \in \Delta_g}{extr}, \tag{3}$$

$$\Delta_g = \left\{ < U^t, S_\delta^{t_f} > \middle| R_\beta\left(X_\chi^t, F_\chi^t, Z_\chi^t, M_{<\chi,\chi'>}^t, \Pi_{<\tilde{\delta},\tilde{\tilde{\delta}}>}^t\right) \le \tilde{R}_g; U^t = \Pi_{<\delta_1,\delta_2>}^{t_1} \circ \Pi_{<\delta_2,\delta_3>}^{t_2} \circ \Pi_{<\tilde{\delta},\delta>}^{t_2}; \beta \in \mathbf{B} \right\},$$

где U^t — управляющие воздействия, позволяющие синтезировать как архитектуры приложения видеоконференцсвязи, так и алгоритмы и программные средства функционирования ее элементов; J_θ — стоимостные, временные, ресурсные показатели, характеризующие качество функционирования системы видеоконференцсвязи; Δ_g — множество динамических альтернатив (множество A_χ архитектур приложения видеоконференцсвязи, их параметров Z_χ^t); $\mathbf{B}$ — множество номеров пространственно-временных, технических и технологических ограничений, определяющих процессы обработки и передачи информации в пиринговых приложениях; $\tilde{R}_g$ — заданные величины; $T = (t_0,\ t_f]$ — интервал времени, на котором синтезируются как архитектуры, так и программно-алгоритмические решения для приложения видеоконференцсвязи.

В качестве показателей J_θ качества функционирования таких систем на этапах синтеза ее различных архитектур целесообразно использовать объемы передаваемых данных и потребляемых ресурсов сервера и клиентских приложений.

Архитектура клиентской части $A_{\text{КЛИЕНТ}}\left(\chi = 1\right)$ пирингового приложения видеоконференцсвязи, представленная на рисунке 1, содержит три основные составляющие X_1^t : модуль логики приложения $M_{\text{ЛП}}$ видеоконференцсвязи, модуль графического интерфейса $M_{\text{ГИ}}$ и хранилище состояний $M_{\text{ХС}}$. Модули логики $M_{\text{ЛП}}$ и графического интерфейса $M_{\text{ГИ}}$ приложения с помощью модуля

действий $M_{Д}$ и модуля изменения состояния хранилища $M_{ИХ}$ осуществляют передачу новых данных *Data* в хранилище. Модуль действий $M_{Д}$ генерирует сообщение S, что произошло некоторое изменение данных в приложении.

Все возможные изменения данных описываются типами действий $Type_{Д}$, которые заранее предопределяются разработчиком. Чтобы использовать действие, необходимо вызвать функцию $F_{Д}$ данного действия, которая отправит сообщение S модулю изменения хранилища $M_{ИХ}$. Отправленное сообщение S должно содержать тип действия $Type_{Д}$ и новую информацию I, предназначенную для хранилища $M_{ХС}$. Одним из главных алгоритмов данного модуля $M_{ХС}$ является алгоритм изменения данных в хранилище состояний, который реализуется в модуле изменения хранилища $M_{ИХ}$. Модуль изменения хранилища $M_{ИХ}$ обеспечивает обработку информации I от модуля действий $M_{Д}$ и создает новое состояние $State_{Д}$ данных для хранилища. Важно отметить, что модуль изменения состояния хранилища $M_{ИХ}$ должен создавать новый набор данных *Data*, а не модифицировать старые. Данное ограничение обусловлено тем, что приложение видеоконференцсвязи должно хранить именно промежуточные состояния, которые легко отслеживать и отлаживать при разработке и работе приложения видеоконференцсвязи. В итоге модуль изменения хранилища $M_{ИХ}$ указывает, как состояние $State_{П}$ приложения видеоконференцсвязи должно измениться в ответ на определенное действие, произошедшее в данном приложении.

2.1. Архитектура клиентской части пирингового приложения видеоконференцсвязи

В разработанном приложении видеоконференцсвязи пиринговым узлом является клиентская часть. Данная часть должна обрабатывать и хранить большое количество промежуточных данных, поэтому рассмотрим ее более подробно. В клиентскую часть приложения видеоконференцсвязи информация поступает асинхронно или потоками от других клиентов и сервера. Разнородная

информация требует больших затрат для ее логической и графической обработки и вывода. Обеспечение взаимодействия различных логических модулей и модулей вывода графики приложения видеоконференцсвязи ведет к увеличению количества кода клиентской части приложения видеоконференцсвязи и к сложности разработки компонентов данной части. Кроме того, необходимо хранить данные текущего состояния приложения видеоконференцсвязи для их графического представления и взаимодействия его различных частей. Для решения проблемы обработки и хранения разнородного типа информации, поступающей в различные моменты времени, обеспечения модульности и простого взаимодействия логических частей приложения видеоконференцсвязи с графическим интерфейсом была разработана архитектура клиентской части пирингового приложения видеоконференцсвязи. Разработанная архитектура включает в себя различные модули, представленные на рисунке 2.1.

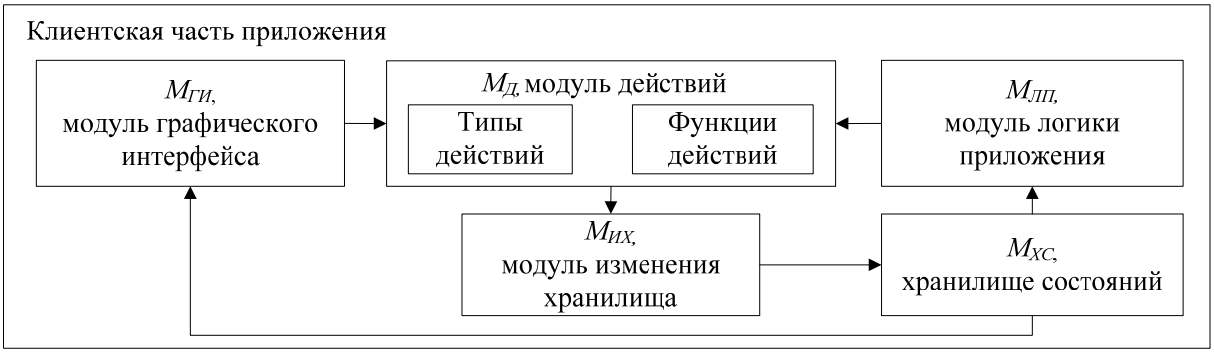


Рис. 2.1. Архитектура клиентской части пирингового приложения видеоконференцсвязи

Данная архитектура содержит три главные составляющие: модуль логики приложения видеоконференцсвязи, модуль графического интерфейса и хранилище состояний. Хранилище состояний $M_{\text{ХС}}$ содержит типы данных $Type_{Data}$, которые соответствуют определенному объекту управления $O_{\text{у}}$. При изменении одного из состояний $State_{\text{П}}$ в приложении видеоконференцсвязи генерируется оповещение для логики $Not_{\text{Л}}$ или графического интерфейса $Not_{\text{ГИ}}$ данной архитектуры. При получении такого оповещения, модуль логики $M_{\text{ЛП}}$

или графического интерфейса $M_{ГИ}$ сможет обработать новые данные *Data* и обеспечить дальнейшее функционирование системы.

Модули логики и графического интерфейса приложения с помощью модуля действий и модуля изменения состояния хранилища осуществляют передачу новых данных в хранилище. Рассмотрим работу этих модулей подробнее.

Модуль действий генерирует сообщение, что произошло некоторое изменение данных в приложении. Все возможные изменения данных описываются типами действий, которые заранее предопределяются разработчиком. Чтобы использовать действие, необходимо вызвать функцию данного действия, которая отправит сообщение модулю изменения хранилища. Отправленное сообщение должно содержать тип действия и новую информацию, предназначенную для хранилища.

Модуль изменения хранилища обеспечивает обработку информации от модуля действий и создает новое состояние данных для хранилища. Важно отметить, что модуль изменения состояния хранилища должен создавать новый набор данных, а не модифицировать старые. Такое ограничение обусловлено тем, что приложение видеоконференцсвязи должно хранить именно промежуточные состояния, которые легко отслеживать и отлаживать при разработке и работе приложения видеоконференцсвязи. В итоге модуль изменения хранилища указывает, как состояние приложения видеоконференцсвязи должно измениться в ответ на определенное действие, произошедшее в данном приложении.

Хранилище состояний содержит различные типы данных, которые соответствуют определенному объекту управления. При изменении одного из состояний в приложении видеоконференцсвязи генерируется оповещение для логики или графического интерфейса данной архитектуры. При получении такого оповещения, модуль логики или графического интерфейса сможет обработать новую информацию и обеспечить дальнейшее функционирование

системы. Алгоритм изменения данных в хранилище состояний представлен на рисунке 2.2.

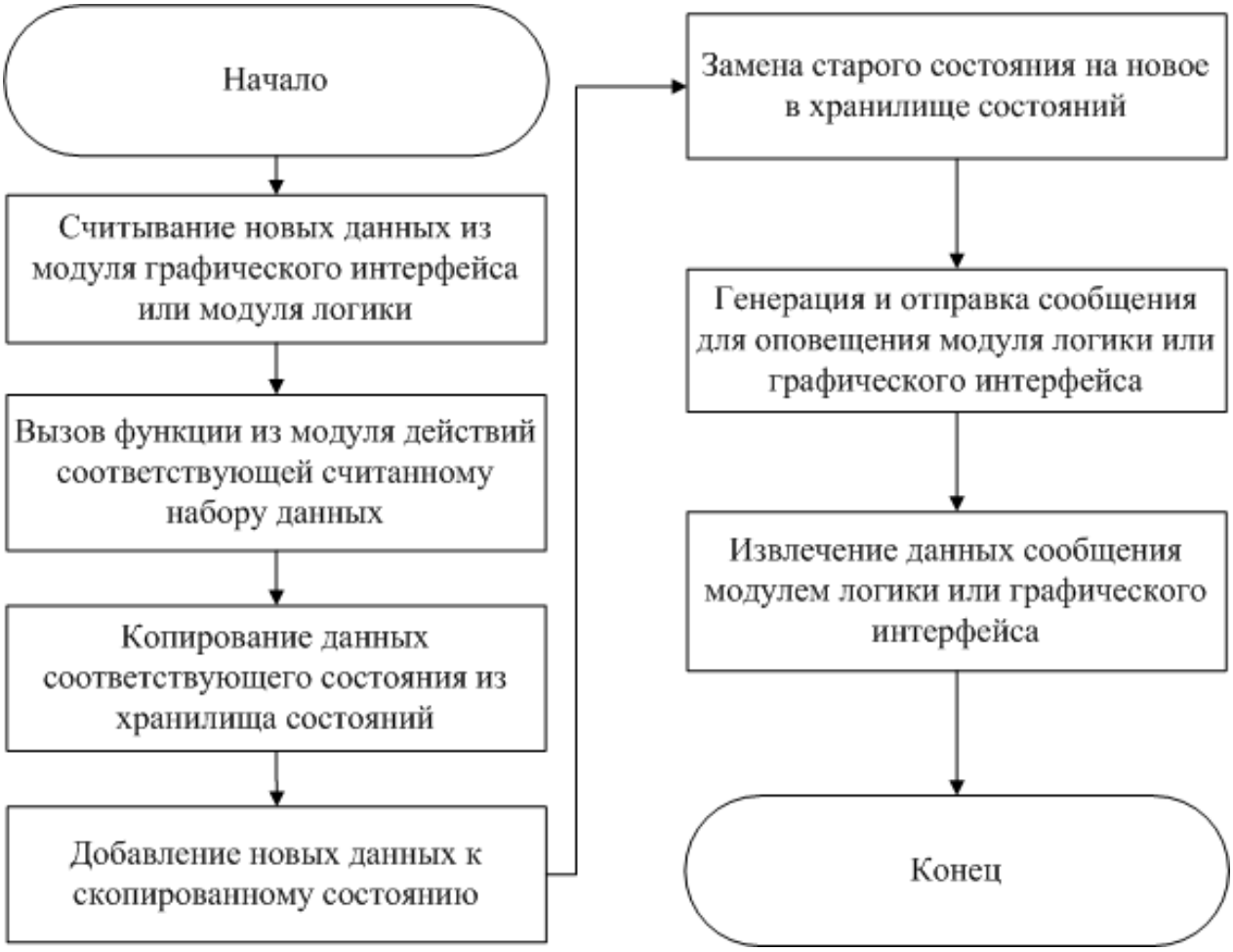


Рис. 2.2. Алгоритм изменения данных в хранилище состояний

Алгоритм изменения данных в хранилище состояний начинает свою работу со считывания данных из модуля графического интерфейса или логики, в худшем случае данная операция является циклом и имеет сложность n. Далее происходит вызов функции модуля действий (рисунок 2.1.), которая обеспечивает передачу определенных данных модулю изменения хранилища. Данный вызов имеет сложность C1 = const. Модуль изменения состояний копирует необходимую информацию из хранилища состояния, добавляет новые данные к скопированному состоянию и заменяет старое состояние преобразованным новым. Несмотря на большое количество действий, необходимых для реализации данного процесса, его сложность является

циклической и равна n. Генерация сообщения с новыми данными о состоянии, и отправка его в модуль логики или графического интерфейса является предпоследней операцией и имеет сложность C2 = const. Последняя операция рассматриваемого алгоритма имеет сложность C3 = const и позволяет извлечь данные из сообщения хранилища состояний. Таким образом, общая сложность алгоритма представляет собой выражение 2n + C1 + C2 + C3 = Θ(n) и является линейной. Данная сложность говорит о малом количестве потребляемых алгоритмом ресурсов, что позволяет использовать его без снижения производительности приложения видеоконференцсвязи.

Подобная архитектура позволяет разграничить работу логики и графического интерфейса, что является важным аспектом разработки крупных и масштабируемых систем. Организация взаимодействия модулей через хранилище состояний обеспечивает отсутствие прямого взаимодействия модулей в системе. Такой подход четко структурирует типы данных для обмена между частями приложения видеоконференцсвязи, исключая повторение, лишние связи между данными и однородность при разработке компонентов системы.

2.2. Архитектура взаимодействия модулей веб-приложения видеоконференцсвязи

Разработанная архитектура $A_{\text{ОБМЕН}}\left(\chi = 3\right)$, представленная на рисунке 2.3, позволяет предотвратить потерю «сигнальных» данных при соединении трех и более участников видеоконференцсвязи. Основными структурными элементами X_3^t архитектуры $A_{\text{ОБМЕН}}$ являются: 1 — клиентская часть приложения; 2 — серверная часть приложения; 3 — блок протоколов передачи данных. Клиентская часть подразделяется на две независимые составляющие — устройство пользователя и веб-страница. Устройство пользователя в приложении необходимо для создания аудио и видеопотоков с камеры и микрофона, подключенных или являющихся частью устройства.

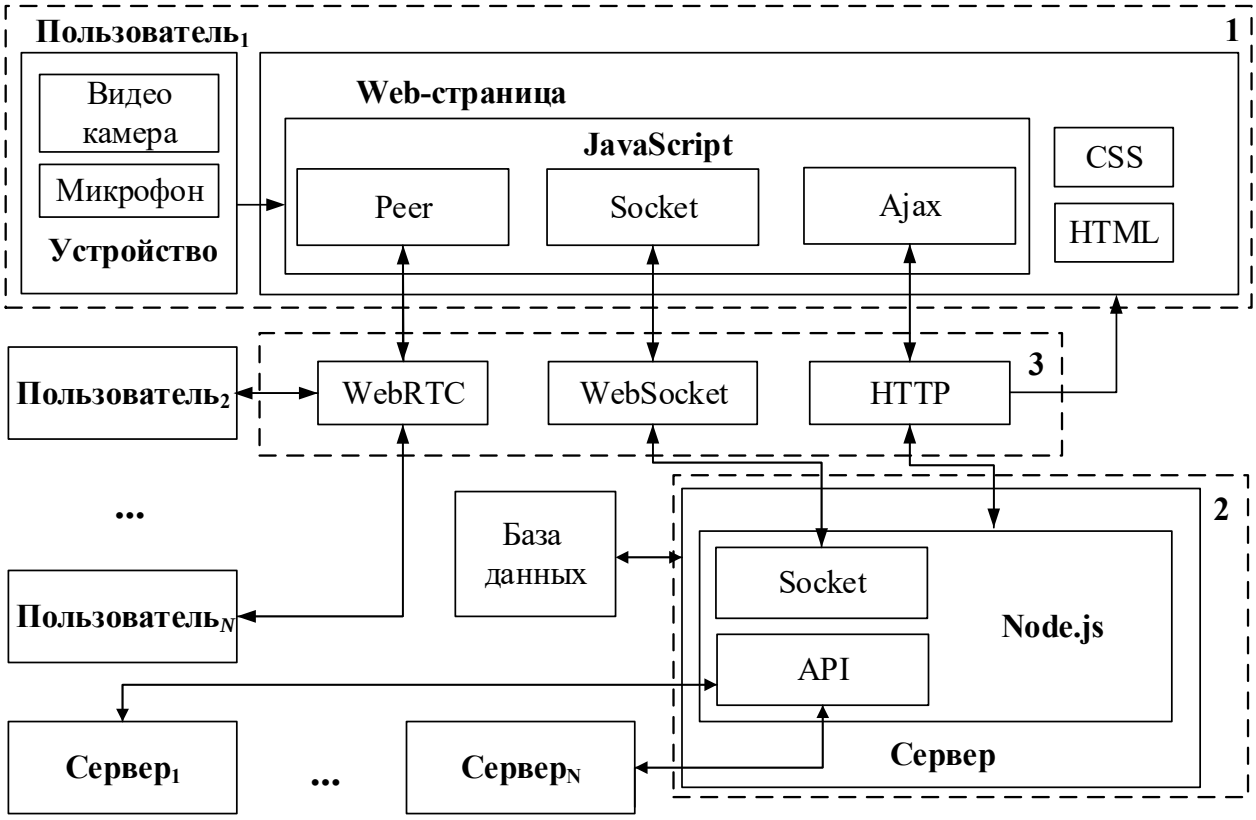


Рис. 2.3. Разработанная архитектура обмена данными в приложении видеоконференцсвязи

Веб-страница клиентской части приложения состоит из классов, написанных на языке программирования JavaScript, необходимых для создания соединений с сервером и другими клиентами с помощью различных протоколов и обработки данных. Средства CSS и HTML служат для построения графического интерфейса, отображения данных и управления клиентской частью приложения. Средства JavaScript, использующиеся на веб-странице видеочата, включают в себя три различных типа инструкций, позволяющих организовать передачу данных по трем протоколам: WebRTC, WebSocket и HTTP. Также средства JavaScript служат для захвата и обработки потоков данных с микрофона и видеокамеры.

Следующим основным элементом архитектуры приложения является серверная часть. Она выполняет несколько различных функций: формирование клиентской части приложения; регистрация клиента; авторизация клиента; обмен «сигнальными» данными между клиентами; создание комнат чата и работу с базой данных. Сам сервер работает на платформе № de.js,

транслирующей JavaScript в машинный код и имеет такую же асинхронную архитектуру, как и клиентская часть, разработанная средствами языка программирования JavaScript. База данных MongoDB, расположенная в серверной части приложения имеет № SQL архитектуру, которая подходит для упрощения реализации серверной части и позволяет быстро адаптировать ее данные к изменениям структуры приложения. Взаимодействие с базой данных MongoDB, происходит с помощью JavaScript и специальной библиотеки драйвера, предназначенной для этой базы данных.

Третий элемент архитектуры, представленной на рисунке 2.3, состоит из протоколов — HTTP, WebSocket и WebRTC. Эти протоколы обеспечивают обмен данными на различных этапах работы приложения, с их помощью осуществляется создание соединения клиентских частей по протоколу WebRTC для передачи потоковых аудио- и видеоданных между ними. Существуют проблемы, возникающие при создании соединения, исходящие из асинхронной архитектуры приложения и протокола WebRTC, не предусматривающего стандартную реализацию соединения множества клиентов. Также необходимо отметить сложность процедуры установления связи между клиентами по протоколу WebRTC, требующую обмена «сигнальными» данными между ними и требующего особого внимания при создании соединения.

В данной работе описано решение вышерассмотренной проблемы потери сигнальных данных при множественном соединении клиентов видеоконференцсвязи с помощью внедрения новых алгоритмов взаимодействия клиентских и серверной частей и использования различных протоколов для организации обмена информацией. Такая архитектура позволяет создать полноценное пиринговое приложение видеоконференцсвязи, которое может работать в режиме группового видео чата. В следующем разделе рассмотрены основные протоколы и программные средства, использующиеся для создания клиентской веб-страницы и ее функционирования при проведении видеоконференции.

2.3. Алгоритмы установления соединений между клиентами по протоколу WebRTC

Чтобы ясно представлять проблему потери «сигнальных» и предложенные в данном исследовании программно-алгоритмические решения данных, вначале рассмотрим основные этапы функционирования клиентской и серверной частей разработанного приложения видеоконференцсвязи.

Клиентская часть приложения начинает работу с формирования веб-страницы регистрации или авторизации, позволяющих взаимодействовать клиенту с сервером посредством отправления или получения данных по HTTP протоколу.

HTTP — это протокол прикладного уровня для передачи произвольных данных. Протокол используется в приложении для передачи клиенту графического интерфейса в виде HTML и CSS данных, логики клиентской части приложения, написанной на языке программирования JavaScript, а также для обмена данными клиента с сервером при регистрации и авторизации клиента с помощью технологии Ajax, позволяющей обмениваться данными с сервером по протоколу HTTP без перезагрузки веб-страницы.

Авторизация клиента позволяет пользователю получить доступ к персональным данным и странице видеоконференцсвязи. Для авторизации пользователь вводит данные в формы «логин» и «пароль». Затем происходит сбор данных из форм и отправка этих данных на сервер с помощью технологии Ajax. Далее сервер обрабатывает полученные данные: проверяет на соответствие определенному набору символов и на превышение максимального размера данных в запросе, выполняет поиск пары логин-пароль по базе данных. При успешном завершении всех операций сервер сформирует страницу видеоконференцсвязи с пользовательскими данными и отправит ее по протоколу HTTP-клиенту. В случае несоответствия данных определенным требованиям или при возникновении ошибки, сервер отправит клиенту информационное сообщение, способствующее устранению возникшей ситуации, используя протокол HTTP.

Регистрация клиента, также как и авторизация, предполагает заполнение форм данными и их отправку на сервер по протоколу HTTP. Далее сервер обрабатывает данные присланные клиентом. В случае положительного результата обработки, сервер сохранит все полученные данные в базе данных, проведет в автоматическом режиме регистрацию клиента, сформирует и отправит ответ на запрос клиента в виде страницы видеоконференцсвязи с пользовательскими данными. В случае ошибочных данных сервер вернет уведомление об ошибке клиенту по протоколу HTTP.

Таким образом, приложение использует HTTP протокол для надежной передачи HTML, CSS и JavaScript данных между сервером и клиентом. Преимущество использования этого протокола в том, что он специально предназначен для передачи веб-страниц и их логики, а также хорошо поддерживается всеми существующими браузерами. Протокол HTTP имеет набор стандартных команд, среди которых есть две основные команды: «GET» и «POST» соответственно, позволяющие осуществлять запросы на выдачу страниц к серверу и запрос на обмен различным типом данных между сервером и клиентом при авторизации или регистрации клиента.

После получения клиентом веб-страницы видеоконференцсвязи средствами JavaScript, создается сокет на клиенте, который устанавливает соединение с сервером с помощью протокола WebSocket. WebSocket — это протокол полнодуплексной связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени. Протокол WebSocket открывает сокеты на клиенте и сервере, позволяющие обмениваться любыми типами данных. В случае успешного соединения по протоколу WebSocket сервер создаст у себя сокет с данными клиента и начнет его авторизацию: попытается получить http cookie данные клиента, которые хранят необходимую информацию для авторизации, произведет распаковку cookie данных, попытается загрузить из базы данных сессию, соответствующую данным из http cookie, по загруженной сессии определит пользователя, принадлежащего сессии, привяжет пользовательские данные к сокету, создаст

уникальный номер для сокета, сгенерирует и отправит событие «connection» внутри сервера. Если одно из действий при авторизации сокета сгенерирует ошибку, то сокет на сервере будет автоматически отключен и удален, а клиентский сокет получит сообщение о разрыве соединения. Событие «connection», возникающее на сервере привязывает к сокету, созданному на сервере — «слушателей» событий, посылаемых сокетом клиента. Сокет клиента еще при его создании формирует набор «слушателей» событий, посылаемых сокетом сервера. Таким образом, устанавливается связь между клиентом и сервером через протокол WebSocket, которая позволяет им быстро обмениваться сообщениями различного типа, не требующими их идентификации, так как для каждого вида сообщений существует отдельный «слушатель».

Данный протокол при построении архитектуры обмена данными позволяет достичь высокой скорости обмена информации и уменьшение нагрузки на клиента и сервер за счет отсутствия затрат на идентификацию потоков данных. В разработанном приложении видеоконференцсвязи протокол WebSocket играет важную роль — он занимается передачей «сигнальных» данных браузеров клиентов, которые позволяют создать соединение по протоколу WebRTC. Таким образом, данный протокол является основой для создания соединения по протоколу WebRTC и упрощает процесс передачи данных необходимых для пирингового соединения.

После установления связи с сервером по протоколу WebSocket пользователю для совершения видеозвонков необходимо включить видеокамеру и микрофон и дать к ним доступ браузеру. Браузер, получивший доступ к камере и микрофону пользователя, используя средства JavaScript, сформирует медиа потоки данных с подключенных устройств. Полученные аудио- и видеопотоки можно будет передать по протоколу WebRTC между браузерами клиентов напрямую. WebRTC — интернет-протокол, предназначенный для организации передачи потоковых данных между браузерами или другими поддерживающими его приложениями по технологии точка-точка. Для соединения двух клиентов по протоколу WebRTC необходимо следующий набор JavaScript инструкций: создание пира для каждого

из клиентов; назначение одного из клиентов как «вызывающего»; назначение другого клиента как «отвечающего»; формирование «сигнальных» данных; обмен «сигнальными» данными; завершение установки соединения.

Для передачи «сигнальных» данных между клиентами используется сервер и протокол WebSocket. Ранее созданные сокеты в клиентской части приложения позволяют передавать «сигнальные» данные по определенным каналам серверу, сервер в свою очередь транслирует эти данные другим клиентам, для которых они предназначены. Для соединения клиентов по протоколу WebRTC необходимо три типа данных: «call offer», «call answer» и «candidate». «Call offer» служит для инициализации сессии WebRTC, он формируется на одном из клиентов и с помощью WebSocket протокола пересылается серверу, сервер в свою очередь передает данное сообщение «отвечающему» клиенту. «Call offer» имеет формат SDP (Session Description Protocol). Сообщение SDP, передаваемое от одного узла другому, которое может указывать: адреса места назначения, служащие для медиа-потоков мультикастинг-адресами, номера UDP портов для отправителя и получателя, медиа-форматы (например, кодеки), применяющиеся во время сессии, время старта и остановки. Сообщение SDP используется для широковещательных сессий, например, телевизионных, радиопрограмм или видеоконференций. Клиент, получивший «call offer», сформирует и отправит ответ по протоколу WebSocket в виде данных «call answer», которые так же имеют формат SDP. Как только клиент, отправлявший «call offer», получит SDP сообщение типа «call answer», между клиентами начнется обмен данными типа «candidate» по протоколу WebSocket. Данные типа «candidate» имеют формат ICE (Interactive Connectivity Establishment) Candidate. Создание интерактивного подключения (ICE) — это метод, используемый в компьютерных сетях, включающий в себя передачу сетевых адресов (NATs) в таких интернет приложениях как ip-телефония(VoIP), приложениях пиринговой передачи данных (peer-to-peer communications), видеоприложениях, системах мгновенного обмена сообщениями (instant messaging) и других интерактивных медиа приложениях. Данные типа «candidate» используются для соединения

клиентов, устанавливая путь между ними, по которому будут передаваться медиа потоки. При успешном обмене данными типа «candidate», каждый из клиентов откроет канал для передачи различного типа данных по протоколу WebRTC, в том числе аудио- и видеопотоков.

Протокол WebRTC имеет особенности, которые создают сложности при соединении пользователей: для создания соединения между клиентами требуется выполнить операцию «handshake», которая состоит из обмена различным типом «сигнальных» данных между браузерами, но одновременно клиент может устанавливать только одно соединение по протоколу WebRTC.

Данная специфика протокола влечет за собой ряд проблем при создании полноценной видеоконференцсвязи. Проблемы возникают из-за асинхронной архитектуры приложения в ситуациях соединения: одного клиента со множеством, множества с одним, множества со множеством. Такие ситуации приводят к нарушению алгоритма соединения — полной или частичной потере данных, требующихся для установления связи между клиентами. Чтобы решить сложившуюся проблему, было предложено несколько дополнительных подходов к построению архитектуры обмена данными в приложении: буферизация «сигнальных» данных протокола WebRTC на клиенте и сервере; объединение сокетов, соединяемых клиентов в «комнату» на сервере. «Комната» — это массив, находящийся на сервере, состоящий из нескольких сокетов и позволяющий обмениваться данными только с сокетами, находящимися в данном массиве. Таким образом «комнаты» изолируют группы сокетов друг от друга, способствуя распространению данных только внутри определенных групп. Такие подходы помогают исключить потерю данных, позволяют создать все необходимые соединения между клиентами и управлять процессами соединения клиентов на различных этапах работы приложения. Далее рассмотрим алгоритмы, основанные на разработанных подходах, позволяющие создать соединение по протоколу WebRTC, контролировать обработку «сигнальных» данных, осуществлять буферизацию «сигнальных» данных и группировать сокеты клиентов в отдельные «комнаты».

Алгоритм, представленный на рисунке 2.4, описывает этап соединения клиентов до формирования «сигнальных» данных. Сначала «вызывающий» клиент подает запрос на сервер для соединения с «отвечающим» клиентом по протоколу WebSocket. Далее сервер производит поиск «отвечающего» клиента среди подключенных. Если клиента нет, то сервер завершит звонок «вызывающего» клиента, если «отвечающий» клиент найден, то ему отправляется запрос на соединение по протоколу WebSocket. «Отвечающий» клиент формирует и отправляет ответ на запрос. Если ответ отрицательный, то сервер завершает звонок «вызывающего» клиента. В случае положительного ответа сервер получит id сокетов «вызывающего» и «отвечающего» клиентов, по id сокетов найдет «комнату», в которых сокеты находятся в данный момент. После завершения данного алгоритма, происходит создание и обработка буферов сокетов на сервере посредством алгоритма, изображенного на рисунке 2.5.

После того как «комната», где находятся сокеты каждого из клиентов, сформирована, начинает работать алгоритм, представленный на рисунке 2.5. Сначала извлекается сокет из «комнаты» «отвечающего» клиента, для него создается буфер для хранения сокетов, ожидающих соединения по протоколу WebRTC. Затем сокет из «комнаты» «отвечающего» клиента добавляет в уже существующий буфер — сокет, взятый из «комнаты» «вызывающего» клиента. Если взятый сокет из «комнаты» «вызывающего» клиента не был последним, то операция извлечения сокета из «комнаты» «вызывающего» клиента и его добавление в буфер повторяется со следующим сокетом из этой комнаты. После того как будет взят последний сокет из «комнаты» «вызывающего» клиента, сокет из «комнаты» «отвечающего» клиента отсоединяется от своей «комнаты» и добавляется в «комнату» «вызывающего» клиента.

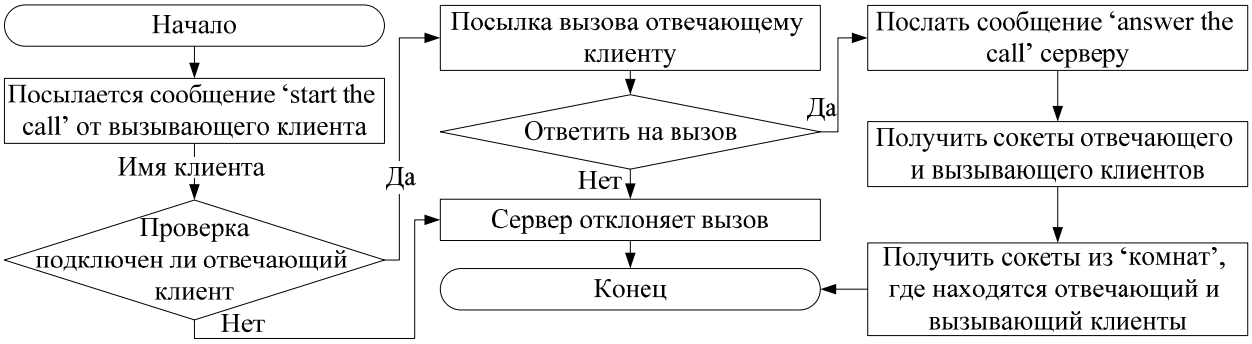


Рис. 2.4. Алгоритм подготовки клиентских частей перед формированием сигнальных данных

Далее происходит вызов функции обработки буфера сокета, которая выполнит запрос на формирование сигнальных данных для всех клиентов, находящихся в очереди данного буфера. В конце алгоритма выполняется проверка на пустоту «комнаты» «отвечающего» клиента. Если «комната» не пуста, то алгоритм повторит все действия с самого начала, в ином случае алгоритм считается завершенным. Как можно заметить, данный алгоритм добавляет в «комнату» «вызывающего» клиента клиентов из «комнаты» «отвечающего» клиента, и при каждом обращении к «комнате» «вызывающего» клиента в данном алгоритме «комната» должна увеличиваться. Но это не так, поскольку перед началом алгоритма происходит дублирование всех пользователей из «комнаты» «вызывающего» клиента в отдельный массив. Таким образом, алгоритм работает правильно и каждый раз он использует не основную «комнату», а заранее подготовленный массив.

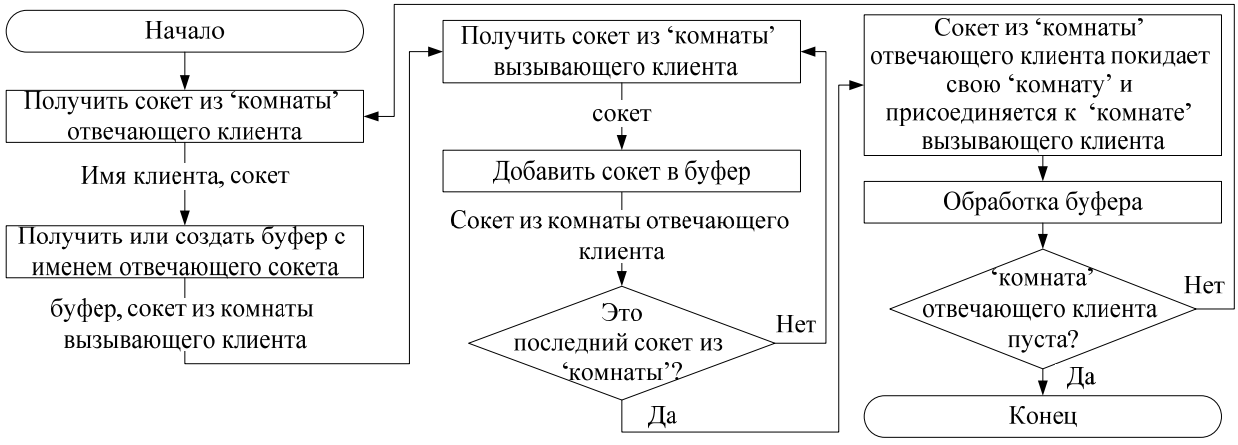


Рис. 2.5. Алгоритм распределения сокетов и обработки их буферов на сервере

Алгоритм, представленный на рисунке 2.6, является общим для обработки запросов от сервера на формирование сигнальных данных «call offer» или на обработку пришедших от другого клиента сигнальных данных «call answer». В клиентской части приложении видеоконференцсвязи существует два отдельных буфера для каждого типа данных.

Приходящий запрос или «сигнальные» данные сначала добавляются в соответствующий им буфер. Затем происходит проверка обоих буферов на предмет обработки одного из них в данный момент. Если один из буферов занят обрабатывающей функции, то алгоритм завершается, а новые данные в буфере будут обработаны позже. Если ни один из буферов не занят, они проверяются на пустоту. В случае если оба буфера пусты, клиент по протоколу WebSocket отправит серверу уведомление, что он готов получать новые данные для установления соединения с другими клиентами. Если какой-то из буферов содержит данные, то будут извлечены первые в очереди данные и обработаны соответствующим образом для установления подключения. Как только подключение будет установлено, алгоритм продолжит свою работу с места опроса занятости буфера.

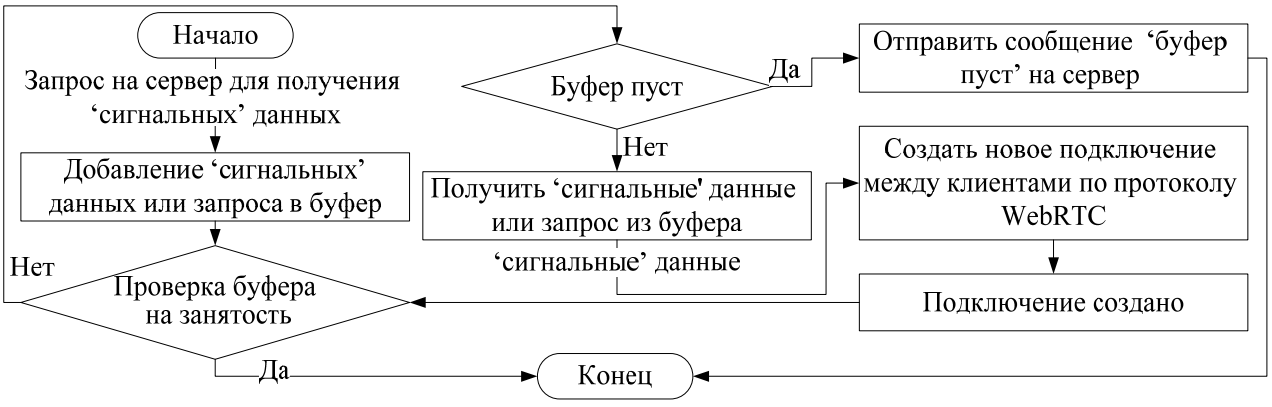


Рис. 2.6. Алгоритм работы с буфером данных на клиенте

Стоит отметить, что «сигнальные» данные типа «candidate» не имеют специальных буферов для их хранения ни на клиенте, ни на сервере, так как обработка происходит сразу же, как только клиент их получит. Во время обработки «сигнальных» данных типа «candidate» буферы, принадлежащие данным типа «call answer» или «call offer», в зависимости от ситуации

обработки, продолжают быть заняты обрабатывающей функцией. Таким образом, остальные данные типа «call answer» или «call offer» продолжают добавляться в буферы, не нарушая алгоритма соединения клиентов по протоколу WebRTC.

Три выше приведенных алгоритма составляют одну из главных частей приложения, которая осуществляет создание видеоконференцсвязи с другими пользователями посредством peer-to-peer протокола WebRTC. Алгоритмы позволяют контролировать асинхронную архитектуру клиентской и серверной частей приложения, осуществляя обработку данных по необходимости, препятствуя возникновению ситуаций перемешивания данных и прерывания выполняющихся соединений по протоколу WebRTC. Следовательно, в приложении достигается возможность создавать групповые видеоконференции и контролировать их состояние с помощью объединенных в «комнаты» клиентов на сервере.

2.4. Алгоритмы взаимодействия клиентской и серверной частей системы видеоконференцсвязи

Для обеспечения хранения промежуточных данных $Data_{\text{ПРОМ}}$, данных пользователей $Data_{\text{ПОЛ}}$ и соединения с другими распределенными модулями в данной архитектуре используется серверная часть, архитектура $A_{\text{СЕРВЕР}}\left(\chi = 2\right)$ которой показана на рисунке 2.7.

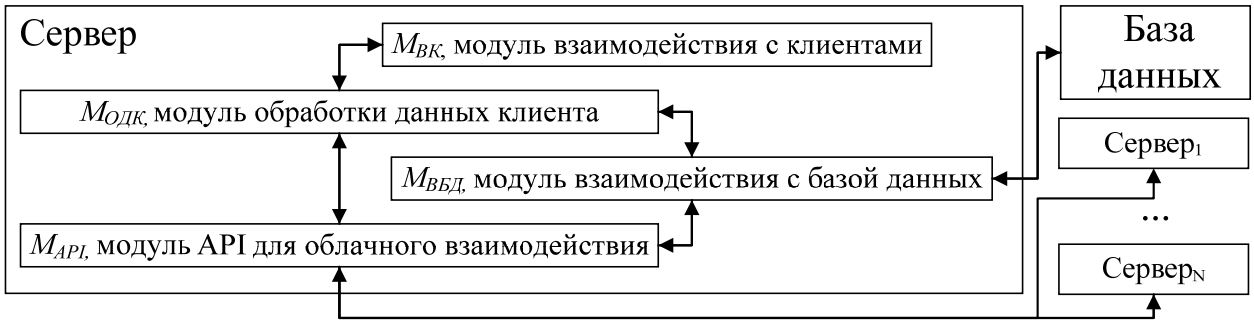


Рис. 2.7. Архитектура серверной части приложения видеоконференцсвязи

Для обеспечения авторизации и обмена сигнальными данными $Data_{\text{СИГН}}$ в серверной части приложения используется модуль взаимодействия с клиентами

$M_{\text{ВК}}$. Алгоритмы, определяющие взаимосвязи между элементами графа $A_{\text{СЕРВЕР}}$, описывают обработку «сигнальных» данных на клиентской и серверной части. Алгоритмы передачи сигнальных данных были подробно описаны в предыдущей части работы. Алгоритм взаимодействия между клиентом и сервером достаточно простой и представлен на рисунке 2.8.

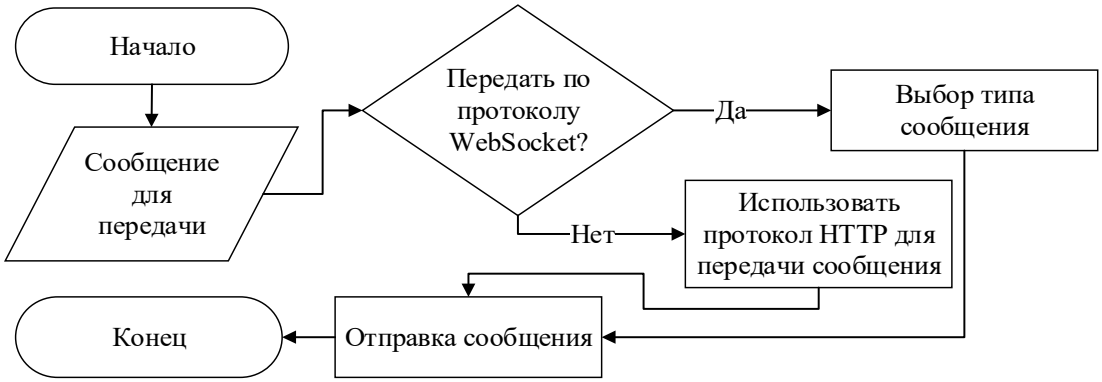


Рис. 2.8. Алгоритм передачи данных между сервером и клиентом

На вход данному алгоритму приходит сообщение, которое содержит в себе данные и тип сообщения. Следующим шагом следует определение, по какому протоколу необходимо отправить данное сообщение. Если выбирается протокол HTTP, то сообщение просто отправляется в том виде, в каком оно есть. Если выбран протокол WebSocket, то информация будет отправлена по определенному каналу, соответствующему типу сообщения. Алгоритм работает одинаково как в серверной, так и в клиентской частях приложения, обеспечивая одинаковый интерфейс передачи данных, для обеих частей. Алгоритм модуля взаимодействия с клиентами тесно связан с алгоритмами модуля обработки данных клиента, поэтому далее рассмотрим их.

Модуль обработки данных клиента $M_{\text{ОДК}}$ обеспечивает распределение, проверку и временное хранение данных, подключенных к серверу пользователей. Для работы с клиентом в серверной части создается копия его некоторых данных $Data_{\text{МОДК}}$. Алгоритмы, обеспечивающие вышеприведенную работу, представлены на рисунке 2.9.

Рис. 2.9. Алгоритмы модуля обработки данных

Для работы с клиентом в серверной части создается копия некоторых его данных. Для создания копии используется алгоритм, представленный на рисунке 2.9а. Работа разработанного алгоритма начинается с получения данных клиента для его авторизации. Следующим шагом идет проверка логина и пароля клиента на соответствие такой же пары в базе данных. Стоит отметить, что проверка соответствия производится модулем взаимодействия с базой данных; модуль обработки данных клиента является посредником, предоставляющим логин и пароль и получающий ответ об их истинности. При

получении положительного ответа сформируется новая копия клиента на сервере и ему присваивается специальная комната для обеспечения взаимодействия нескольких клиентов в едином пространстве. Дальнейшее управление передастся модулю взаимодействия с клиентами. В случае неверного логина и пароля алгоритм закончит свое выполнение, не создав копию клиента.

После создания копии клиента в серверной части для обеспечения передачи данных между клиентами одного пространства (находящимися в одной комнате), используется алгоритм, представленный на рисунке 2.9б. Алгоритм получает на вход данные от одного из клиентов. Далее он перебирает всех клиентов, находящихся в комнате, и формирует для них сообщения, содержащие необходимую информацию и отправляет их с помощью модуля взаимодействия с клиентами. Если данные не соответствуют условиям проверки, то формируется сообщение об ошибке, которое отправит модуль взаимодействия с клиентами.

Последний алгоритм модуля обработки данных клиента, представленный на рисунке 2.9в, получает на вход некоторые данные от клиента, которые необходимо сохранить. В основном такими данными выступают настройки клиента, личная информация о пользователе и информация об администрируемых аккаунтах. В зависимости от типа данных, производятся определенные проверки и преобразование полученной информации к необходимому типу для их дальнейшей записи в базу данных. В случае несоответствия данных условиям проверки, модуль сформирует сообщение об ошибке, которое будет передано клиенту модулем взаимодействия с клиентом.

Для сохранения, изменения и поиска информации в базе данных на сервере используется модуль взаимодействия с базой данных $M_{\text{ВБД}}$. Работу данного модуля обеспечивает алгоритм, представленный на рисунке 2.10.

Алгоритм работы с базой данных начинается с формирования данных и настроек для запроса. Данные необходимы для уточнения поиска или сохранения информации базы данных. Настройки позволяют задать тип реализуемой операции: сохранение, изменение или извлечение. В случае

успешной обработки операции, заданной запросом, база данных сформирует ответ на запрос. Ответ может содержать как необходимую информацию, в случае операции извлечения, так и положительное подтверждение в случае сохранения или изменения информации в базе данных. Если запрос сформирован некорректно, система управления базой данных определит ошибку и отправит ее на сервер. Сервер, в случае получения ошибки, передаст ее своим модулям для дальнейшей обработки. Необходимо отметить, что разработанная система максимально защищена от ошибок при работе с базой данных благодаря фильтрации и жестко заданным схемам работы с базой данных, которые будут рассмотрены в следующем разделе.

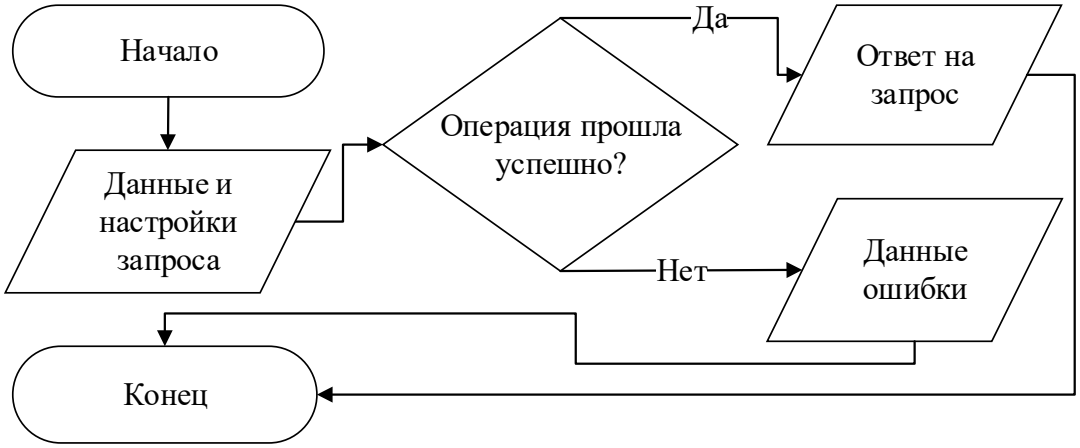


Рис. 2.10. Алгоритм работы с базой данных

Последний модуль сервера M_{API} обеспечивает внешнее взаимодействие сервера с другими серверами, находящимися в единой облачной системе. Данный модуль позволяет получить интересующую информацию другим устройствам, обеспечив формирование данных в заранее заданном виде. На данный момент модуль API облачного взаимодействия позволяет получать актуальную информацию о клиентах и некоторые их данные. Алгоритм реализующей данные возможности представлен на рисунке 2.11.

Работа данного алгоритма начинается с получения данных запроса и его параметров. Параметры запроса будут определять тип поиска данных клиентов. Обработчик запросов получает параметры и данные из тела запроса и определяет, нужно ли провести поиск среди подключенных клиентов. Поиск

среди подключенных клиентов позволяет получить текущую актуальную информацию о данном клиенте, а также удостовериться в его соединении с сервером. Если искать среди подключенных клиентов нет необходимости, то модуль сформирует параметры и данные для запроса в базу данных. В случае необходимости данных о подключенных пользователях, модуль сначала попытается найти требуемую информацию среди подключенных пользователей. Затем, если будет необходимо, сформирует запрос в базу данных на получении информации о тех пользователей, которые не подключены. Последним шагом работы данного алгоритма будет формирование ответа на запрос с его последующей отправкой по сети.

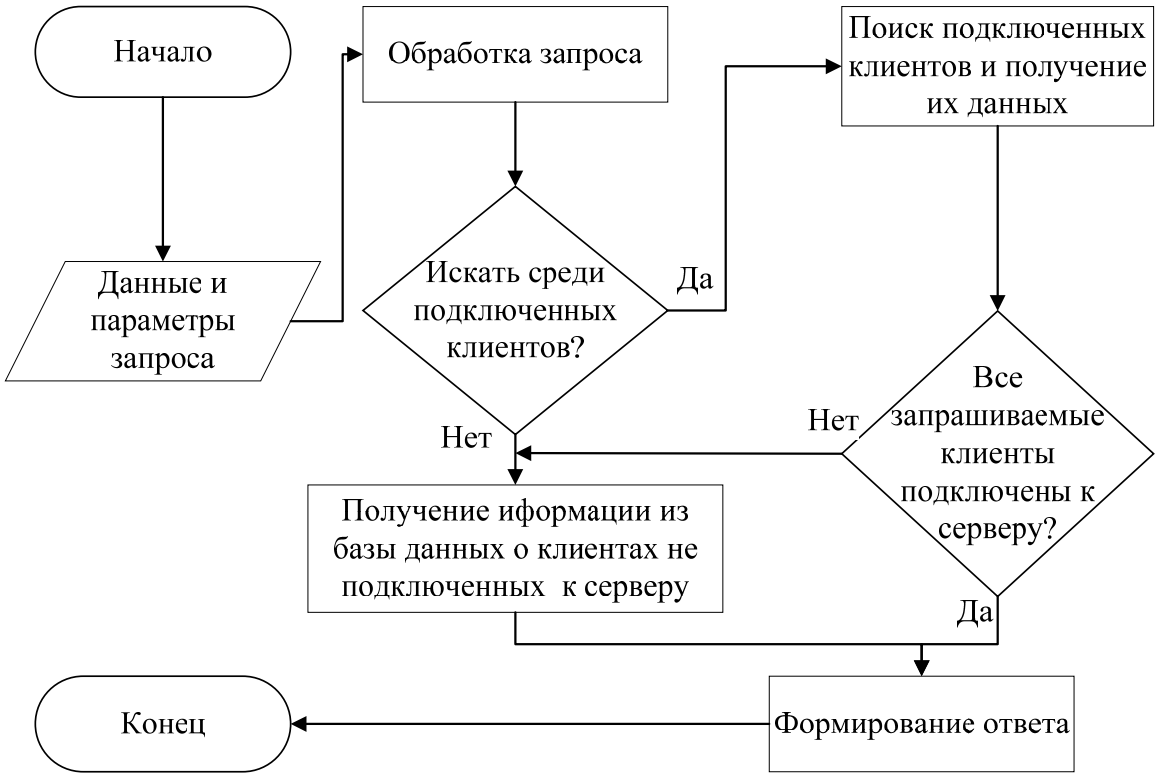


Рис. 2.11. Алгоритм работы API облачного взаимодействия

Вышеперечисленные алгоритмы позволяют организовывать целостную серверную структуру для передачи и обработки внутренних данных и данных из различных источников. Так же сервер позволяет хранить промежуточные данные из различных источников, обеспечивая их контроль и распределение для отдельных групп клиентов. Таким образом, данная серверная архитектура предоставляет полноценную поддержку и обеспечение необходимыми данными всех клиентов.

2.5.Функции и алгоритмы модуля контролируемого аккаунта

В данном разделе рассматриваются алгоритмы обработки данных модуля, внедряемого в уже существующее приложение видеоконференцсвязи на примере контролируемых личных кабинетов (далее контролируемых аккаунтов). Работа контролируемых аккаунтов взаимосвязана с алгоритмами работы приложения видеоконференцсвязи и является частью приложения, архитектура которого представлена в работе [86].

Графический интерфейс для регистрации контролируемого аккаунта доступен только из зарегистрированного обычного аккаунта приложения. Схема графического интерфейса, представленная на рисунке 2.12, позволяет пользователю осуществить регистрацию нового имени пользователя (логина), пароля, а также задать права доступа для управления контролируемым аккаунтом. Для выявления необходимых данных внедрения приведем сравнение обычного и контролируемого аккаунтов относительно возможностей контролируемого аккаунта.

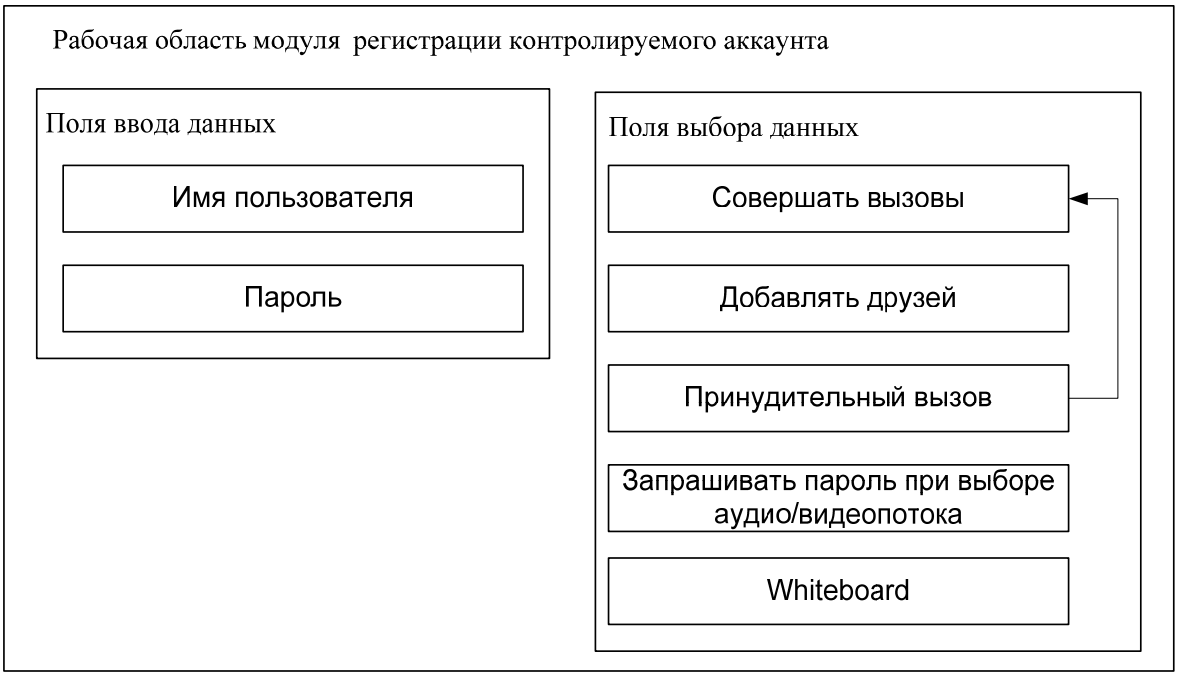


Рис. 2.12. Схема графического интерфейса регистрации контролируемого аккаунта

Из рисунка 2.12 видно, что в области полей выбора существует несколько различных параметров для управления контролируемым аккаунтом. Блокировка возможности совершать вызовы запрещает контролируемому аккаунту самому устанавливать соединение с другими пользователями приложения. При выборе данного пункта приложение автоматически будет разрешать ряду пользователей, выбранных администратором аккаунта, принудительно соединяться с данным аккаунтом, что соответствует третьему пункту полей выбора данных на схеме графического интерфейса. На рисунке 2.12 первый и третий пункты полей выбора соединены стрелкой, что показывает их зависимость друг от друга; при выборе третьего пункта, первый пункт будет автоматически заблокирован. Взаимодействие между клиентской и серверной частями всех рассматриваемых далее алгоритмов осуществляется по протоколу WebSocket. Алгоритмы обычного соединения аккаунтов и принудительного соединения с контролируемым аккаунтом представлены на рисунке 2.13 а, б соответственно.

Данные алгоритмы очень схожи, за исключением нескольких дополнительных проверок, присутствующих в алгоритме принудительного соединения, и последнего этапа работы алгоритмов при удачном выполнении всех проверок. Сначала более подробно рассмотрим алгоритм принудительного соединения и его сложность, так как он содержит в себе большую часть алгоритма обычного соединения аккаунтов. Данный алгоритм направляет текстовый идентификатор запроса и имени аккаунта из клиентской части приложения в серверную. Сложность этой операции равна C1 и является постоянной. Серверная часть приложения опознает идентификатор запроса и запускает соответствующий ему процесс обработки данных, который попытается найти интересующего клиента по идентификатору имени среди списка подключенных к серверу аккаунтов, в худшем случае — содержащему все n-пользователей базы данных. Если клиент будет найден, производится проверка наличия пользователя в друзьях. В худшем случае сложность этой проверки равна n. Если пользователя в друзьях нет, он добавляется в список ожидающих подтверждения, после чего производится проверка, подключен ли пользователь к серверу. При положительном результате трех выше рассмотренных проверок между клиентами будет произведен обмен данными для осуществления пирингового соединения. В случае отрицательного исхода

одной из проверок сервер отправит клиенту сообщение об отклонении запроса на соединение. Эта операция имеет постоянную сложность и равна C2. В итоге выражение, описывающее сложность алгоритма, выглядит как 2n+C1+C2 = Θ(n), поэтому можно сказать, что сложность алгоритма линейна.

Рис. 2.13. а) Алгоритм обычного соединения аккаунтов; б) алгоритм принудительного соединения с аккаунтом

В алгоритме обычного соединения аккаунтов выполняется всего одна проверка, при положительном результате которой будет выполнен не обмен данными для соединения аккаунтов, как в случае с контролируемым аккаунтом, а отправление запроса вызываемому клиенту. Это отличие является очень важным, так как в случае обычного соединения аккаунтов оба аккаунта являются равноправными, и вызываемый аккаунт при получении запроса сможет как принять, так и отклонить его. В случае с контролируемым аккаунтом,

соединение происходит принудительно и никакого подтверждения о соединении с вызываемым аккаунтом не требуется, если для этого выполнены все рассмотренные выше условия. Если рассматривать вопрос сложности, то можно сказать, что изменения в алгоритме не сильно влияют на его сложность. Она описывается линейным выражением $2n+C1+C2= Θ(n)$, также как и в алгоритме принудительного соединения.

При выборе пункта графического меню — добавлять друзей (рисунок 2.12), контролируемый аккаунт сможет осуществлять поиск пользователей в базе данных и отправлять заявки на добавление их в список друзей. Алгоритмы поиска пользователей и добавления пользователей в друзья представлены на рисунке 2.14 а, б соответственно. Два выше приведенных алгоритма работают одинаково для всех типов аккаунтов, ограничения возникают только при отключении работы данных алгоритмов в контролируемых аккаунтах. Далее рассмотрим подробнее этапы их выполнения и вычислительную сложность.

Алгоритм поиска пользователей начинает свою работу с отправления запроса из клиентской части приложения в серверную. Сложность этой операции равна C1 и является постоянной. Запрос представляет собой строку, содержащую целое имя пользователя или его часть. Далее сервер создаст запрос в базу данных, содержащую n-пользователей с данными из переданной строки и получит результат. В случае возврата пустого результата сервер отправит клиенту информационное сообщение, в противном случае, сервер отправит найденные данные в виде массива найденных пользователей. Независимо от варианта развития событий, сложность этой конструкции можно оценить как постоянную, равную C2 . Клиентская часть приложения обработает полученные данные; в случае если новые данные отличаются от тех, что уже есть в хранилище клиентской части приложения, выполнится поиск новых пользователей, со сложностью в худшем случае равной n. Далее произойдет перерисовка графического интерфейса на основе полученных данных, сложность которой постоянна и равна C3. В результате сложность всего

алгоритма описывается выражением 2n+C1+C2+C3=Ɵ(n). Таким образом, сложность алгоритма линейна.

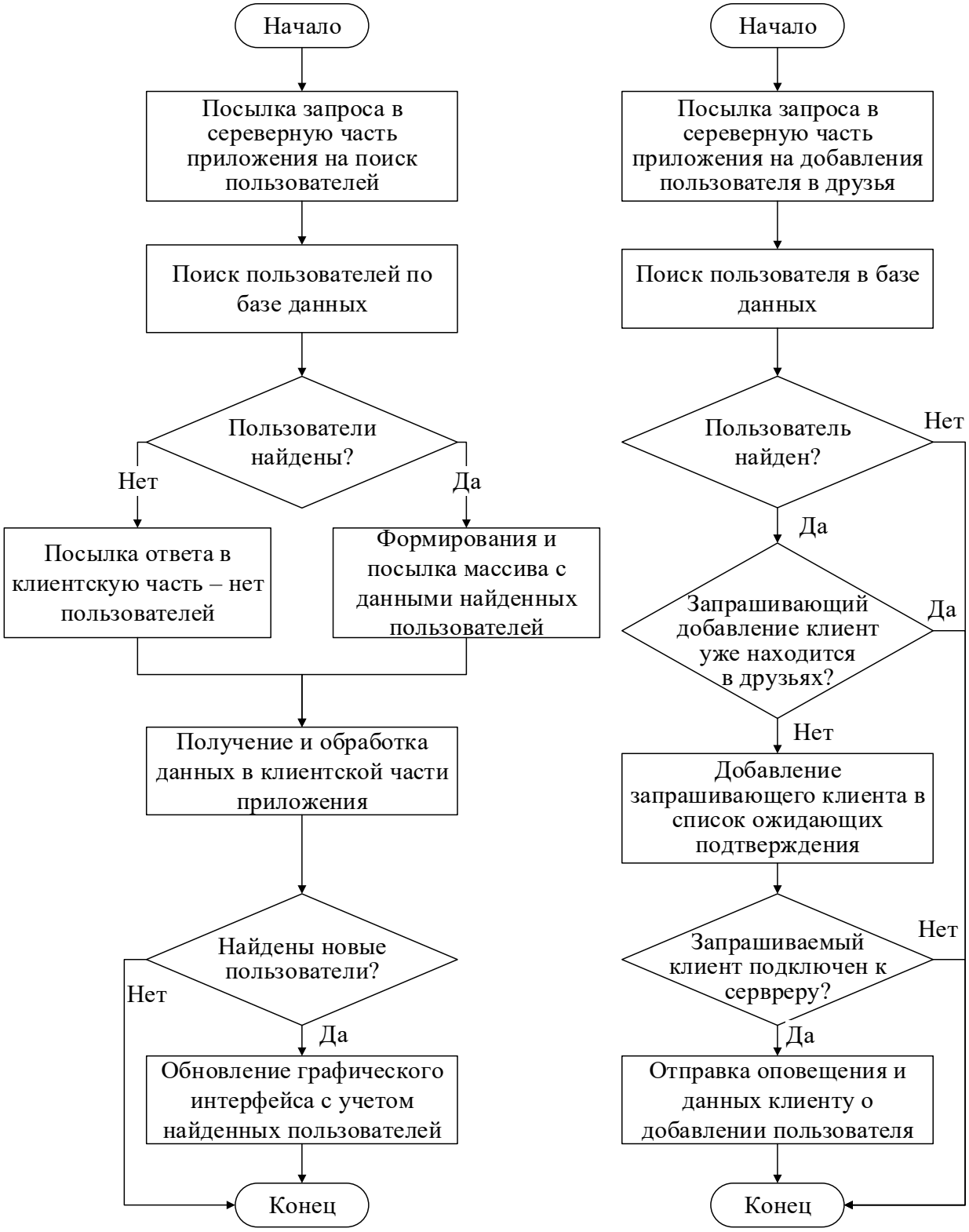


Рис. 2.14. а) Алгоритм поиска пользователей; б) Алгоритм добавления пользователя в друзья

Алгоритм добавления пользователя в друзья начинает свою работу с посылки строки в серверную часть приложения с именем пользователя, которого необходимо добавить в друзья. Сложность этой операции, как и операции из предыдущего алгоритма, постоянна и равна C1. Далее серверная часть приложения выполнит поиск пользователя по базе данных, содержащей n-пользователей, и в случае если пользователь найден (сложность проверки постоянна и равна C2), проверит наличие его в друзьях найденного аккаунта. В худшем случае сложность этой проверки равна n. Если запрашиваемый аккаунт уже находится в друзьях, то алгоритм завершит свою работу, если нет — добавит найденного пользователя в список ожидающих подтверждения запроса на добавление в друзья. Последним шагом алгоритма будет проверка, подключен ли аккаунт найденного пользователя к серверу или нет. В случае если аккаунт подключен, сервер отправит ему данные о новых заявках в друзья. Данные операции имеют сложность n. В итоге выражение, описывающее сложность алгоритма, выглядит как $3n+C1+C2=\Theta(n)$, поэтому можно сказать, что сложность алгоритма линейна.

Примерно 30% операций во всех вышеприведенных алгоритмах занимают различные проверки, которые заменяются обычным блокированием возможностей графического интерфейса в клиентской части приложения. Но ограничение графического интерфейса клиентской части приложения не гарантирует, что пользователь умышленно, с помощью вызова определенных команд, не попытается воспользоваться возможностями, которые заблокированы в графическом интерфейсе и на программном уровне. Также невозможно быть полностью уверенным, что при обфускации клиентского кода его не смогут расшифровать и использовать в собственных целях для совершения XSS-атак на сервер. Поэтому для безопасности серверной части приложения, обеспечения конфиденциальности информации пользователей и разграничения доступа к функционалу приложения используются различные проверки в серверной части приложения.

В любых типах аккаунтов возможность включения и выключения захвата аудио- и видеопотоков с соответствующих устройств доступна по умолчанию. В контролируемых аккаунтах пункт настроек «Запрашивать пароль при выборе аудио/видеопотока» (рисунок 2.12) позволяет ограничить доступ к управлению устройствами посредством логина и пароля. Логин и пароль хранятся в серверной части приложения, и при каждой попытке доступа к устройствам клиентская часть приложения будет запрашивать ввод логина и пароля для отправки его в серверную часть приложения и дальнейшей проверки. Последний пункт графического интерфейса, так же как и предыдущий, оперирует логином и паролем, в данном случае они используются для ограничения доступа к выходу из профиля пользователя. Логин и пароль в приложении впервые используются при регистрации пользователя, затем следует авторизация. Алгоритмы данных процессов представлены на рисунке 2.15. Все последующие операции для подтверждения логина и пароля аккаунта используют тот же алгоритм авторизации, что и в начале работы приложения, но данные в серверную часть пересылаются через протокол WebSocket.

Алгоритм регистрации пользователя начинает свою работу с посылки запроса на регистрацию в серверную часть приложения. Запрос содержит в себе данные имени пользователя, пароля и почты клиента. Сервер проверит поступившие данные на соответствие допустимому набору символов. Сложность этих операций является постоянной и равна C1. Затем следует проверка на уникальность почты и имени пользователя со сложностью n, где n — число пользователей. В случае если данные не прошли одну из проверок, сервер сформирует сообщение с описанием некорректных данных и отправит его в клиентскую часть. В случае если все введенные пользователем данные корректны, сервер сохранит их в базу данных и сформирует ответ на запрос клиента с перенаправлением на страницу авторизации. Сложность этих операций оценивается как постоянная и равна C2. В результате сложность алгоритма описывается выражением $n+C1+C2=Θ(n)$, из чего следует, что алгоритм имеет линейную сложность.

Рис. 2.15. а) Алгоритм регистрации пользователя; б) Алгоритм авторизации пользователя

Алгоритм авторизации посылает запрос из клиентской части приложения с именем пользователя и паролем в серверную. Сервер проверит поступившие данные на соответствие допустимому набору символов. Сложность этих операций является постоянной и равна C1. Затем сервер попытается найти пользователя в базе данных с n-пользователями. Если данные, отправленные

клиентом, не корректны, сервер сформирует сообщение с описанием некорректных данных и отправит его в клиентскую часть. В случае успешного завершения проверок сервер перенаправит клиента на страницу-чат.

Сложность этих операций оценивается как постоянная, равная C2. После завершения авторизации сервер отправит клиенту cookie-данные, которые будут идентифицировать клиента всякий раз, когда он попытается отправить новые запросы на сервер, не требуя от него повторного ввода логина и пароля, кроме ситуаций с контролируемыми аккаунтами, описанными выше. Таким образом, сложность алгоритма описывается выражением $n+C1+C2=\Theta(n)$, из чего следует, что алгоритм имеет линейную сложность.

Регистрация контролируемых аккаунтов проходит по тому же алгоритму, что и регистрация обычных аккаунтов, но существуют некоторые различия. Регистрация контролируемого аккаунта может быть осуществлена только со страницы чата зарегистрированным и авторизированным пользователем. Второе различие заключается в протоколе передачи данных: при обычной регистрации аккаунта используется протокол HTTP, при регистрации контролируемого аккаунта — протокол WebSocket, который способствует ускорению идентификации пользователя на сервере, упрощению процесса приема и передачи данных между сервером и клиентом. Последнее различие заключается в количестве данных, отправляемых при регистрации аккаунта; при регистрации контролируемого аккаунта используются только логин и пароль, в то время как при регистрации обычного аккаунта требуется еще и адрес электронной почты. Данное различие возникает по причине того, что контролируемый аккаунт позволяет управлять им с помощью обычного аккаунта, на который был зарегистрирован управляемый аккаунт, поэтому почта у таких аккаунтов является общей.

Контролируемые аккаунты в архитектурном плане отличаются от обычных только взаимодействием с сервером. В случае с пиринговым соединением и передачей мультимедийных данных они подобны обычным аккаунтам.

2.6. Выводы

1. Разработанные пиринговая «безсерверная» архитектура и алгоритмы взаимодействия модулей системы видеоконференцсвязи обеспечивают соединение клиентской и серверной частей системы по протоколу WebRTC, а также коммуникацию клиентских веб-приложений.

2. Предложенная архитектура обмена данными в приложении видеоконференцсвязи позволяет распределить каналы передачи данных по соответствующим протоколам и разделить задачи между ними. Так как изначально архитектура приложения видеоконференцсвязи является асинхронной, то была выполнена разработка и внедрение новых алгоритмов по управлению данными, необходимых для соединения клиентских приложений по протоколу WebRTC. В итоге было получено полноценное приложение видеоконференцсвязи, способное производить групповые звонки и контролировать состояние клиентов, связанных в общие группы — «комнаты» —на сервере во время проведения группового чата.

3. Стоит отметить, что на данный момент существуют еще некоторые проблемы, ограничивающие использование протокола WebRTC на устройствах: 1) всего три браузера (Opera, Mozilla Firefox, Google Chrome) поддерживают данный протокол; 2) требуется наличие мощного процессора и достаточного количества памяти для обработки аудио- и видеопотоков данных. Кроме того, указанные браузеры не работают с графическими сопроцессорами, в результате чего нагружается основной процессор.

4. Дальнейшие исследования будут направлены на упрощение способа передачи и улучшение обработки аудио- и видеоданных с использованием пиринговых связей для распределения нагрузки по обработке данных между клиентами.

Глава 3. Разработанные программные средства приложения видеоконференцсвязи

3.1. Программные средства разработки кроссплатформенных приложений

Сейчас существует множество средств для разработки кроссплатформенных приложений. Данные средства обеспечивают переносимость и однозначную работу приложения на различных платформах. Для обеспечения кроссплатформенности в данной работе были выбраны средства языков разметки и стиля — HTML, CSS и язык программирования JavaScript. Данные технологии позволяют создавать приложения, которые работают в различных браузерах. Браузеры являются одним из основных приложений любой современной операционной системы. Также преимущество разработки браузерных приложений состоит в том, что пользователю не требуется устанавливать программное обеспечение в свою операционную систему, достаточно чтобы сервер передал браузеру все необходимые файлы для запуска и работы клиентской части приложения.

HTML — это язык разметки, позволяющий создавать различные элементы страницы, с которыми будет взаимодействовать пользователь. В данной работе используется язык версии HTML 5. Данная версия имеет следующие преимущества: улучшен синтаксис для осуществления SEO-продвижения (комплекс мер для повышения позиций сайта в результатах выдачи поисковых систем по заранее отобранным запросам); добавлены новые элементы для работы с медиаконтентом и веб-формами; новые возможности для перемещения элементов при помощи JavaScript APIs. Данные возможности позволяют легко интегрировать вывод аудио- и видеопотоков в веб-страницу.

Для стилизации и точного позиционирования элементов графического интерфейса используется язык стиля CSS. Данный язык позволяет взаимодействует с технологией HTML, являясь стандартом для стилизации HTML-тегов. CSS позволяет увеличить скорость разработки новых страниц. Стили, определенные один раз, могут быть использованы неограниченно в любом месте документа. Важной деталью является возможность задать стили в

отдельном файле в виде присвоения различных свойств тегам. Переопределение стилей в таком файле вызовет автоматическое изменение стиля отображения всех объектов, для которых применялся измененный стиль. Переопределение стиля в элементе разметки — это применение некоторого доступного в CSS стиля к данному элементу разметки. При таком способе переопределения изменения коснутся только того элемента, за который отвечает данный тег, и не будут отражаться на других элементах, также выводимых этим тегом в другом месте страницы. Позволяет применить нужный стиль к конкретному участку документа.

Последней кроссплатформенной технологией является язык программирования JavaScript. Данный язык позволяет легко манипулировать объектами HTML и CSS. Обычно JavaScript подключается к HTML странице отдельным файлом, но также может быть интегрирован как часть HTML разметки, отделенная тегом script. Благодаря гибкости языка и новому стандарту — ES6—, можно разрабатывать сложные архитектурные решения, которые могут включать в себя классы, наследуемых потомков, итераторы, генераторы, модули и другие новые полезные возможности языка. В целом JavaScript позволяет просто организовывать управление как с графической составляющей веб-страницы, так и с ее логикой. Большое сообщество разработчиков JavaScript предоставляет огромный набор библиотек для быстрой разработки необходимых компонентов. Одной из таких библиотек, уже встроенных в сам язык, является WebRTC. Данная библиотека обеспечивает реализацию протокола передачи мультимедийных данных в пиринговых сетях, а также позволяет реализовать захват аудио- и видеопотоков стандартными средствами браузера. Таким образом JavaScript прекрасно подходит для реализации пиринговых кроссплатформенных приложений видеоконференцсвязи.

3.2. Программные средства клиентской части приложения

Разработка любой части приложения начинается с создания его архитектуры и основных алгоритмов. Следующим этапом разработки выступает более подробное описание системы, например, при помощи UML-

диаграмм. Для представления системы в данной работе были использованы несколько типов UML-диаграмм: диаграммы прецедентов и диаграммы классов. Диаграммы прецедентов необходимы для описания системы на концептуальном уровне. В работе данный тип диаграмм отображает взаимодействие пользователей с модулями, данные которых представлены в графическом интерфейсе. На рисунке 3.1 представлена диаграмма прецедента, которая отображает возможные варианты выбора модулей в клиентской части приложения.

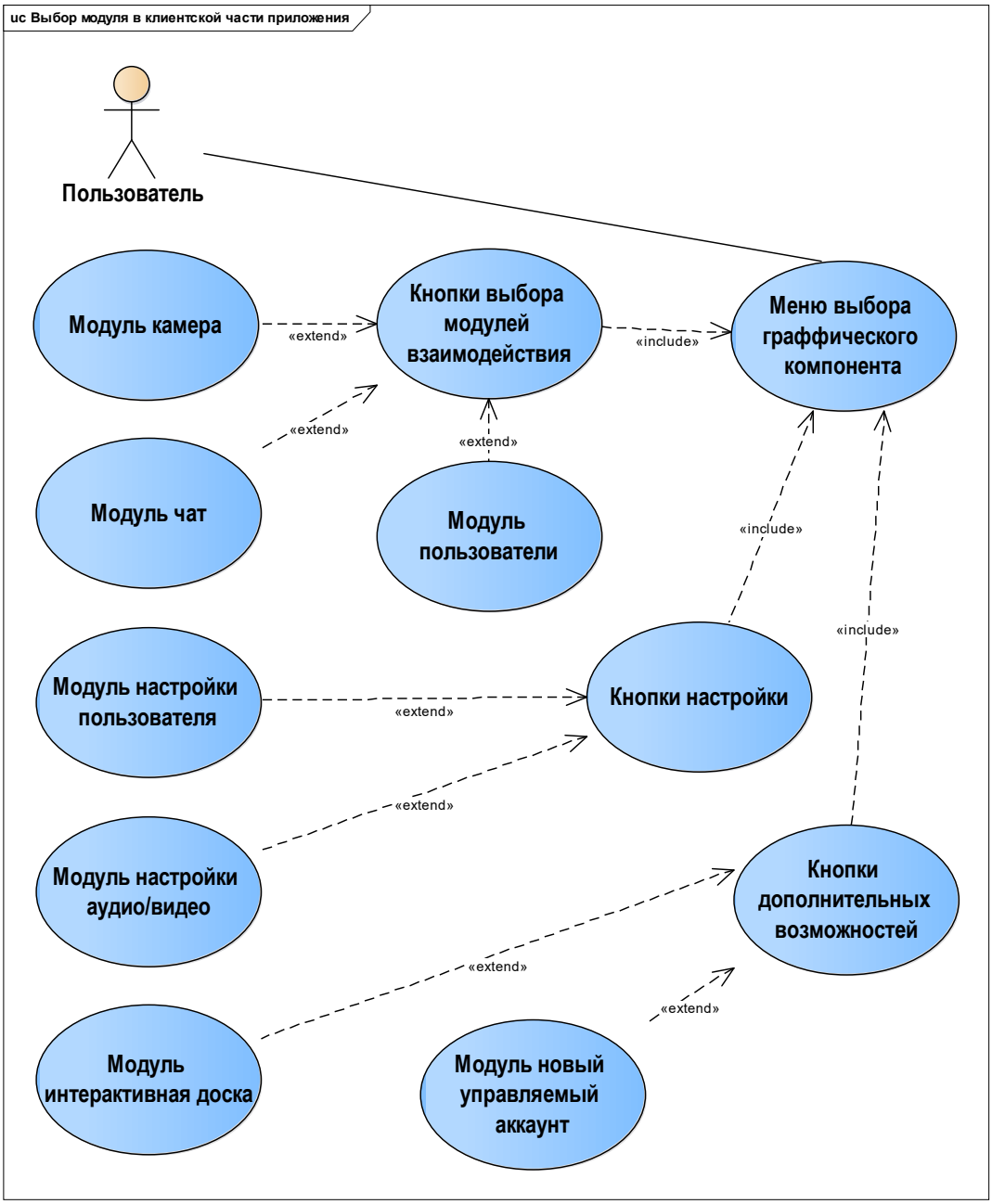


Рис. 3.1. Диаграмма выбора модуля в клиентской части приложения

Пользователь на диаграмме имеет двустороннюю связь с компонентом для выбора определенного модуля. Данный компонент содержит в себе некоторый набор кнопок, который позволяет управлять состоянием видимости различных модулей. Пользователь осуществляет взаимодействие с меню графического компонента для переключения между графическими интерфейсами интересующих его модулей.

Рисунок 3.1 иллюстрирует, что меню выбора графического компонента включает в себя различные кнопки выбора модулей. Логика работы кнопок выбора модулей имеет сильную межмодульную связь, которая обеспечивает отображение только одного модуля в текущий момент времени. Чтобы понять логику работы переключения видимости модуля, рассмотрим рисунок 3.2, на котором представлена диаграмма взаимодействия модулей.

Класс Modules — это синглтон (класс существует в единственном экземпляре), который содержит в себе информацию об остальных модулях и умеет управлять данными, необходимыми для их работы. Внутри класса Modules происходит связь между другими модулями и данными, которыми они обмениваются. Все компоненты, представленные на рисунке 3.2, имеют слабую связь, благодаря ранее разработанной архитектуре, которая полностью позволяет разделить приложение на независимые части. Связь между модулями осуществляется посредством различных хранилищ состояний приложения. Одни модули инициируют изменения в данном хранилище, другие могут подписываться на данные изменения и получают оповещении при их изменении. Таким образом осуществляется передача информации из одного модуля в другой, где прямые связи между модулями полностью отсутствуют и создаются абсолютно новые промежуточные состояния. Так как любой из модулей может быть видимым или скрытым в графическом интерфейсе, то все модули имеют связь с хранилищем состояний отображения. Данное хранилище содержит в себе массив состояний, которые могут принимать два значения «true» или «false» для отображения или скрытия модуля соответственно. Также существует отдельное состояние, которое помеченное как «active»,

соответствующее текущему отображаемому модулю. Состояние «active» помогает быстро скрыть текущий видимый модуль, не итерируя при этом весь массив состояний в поисках текущего видимого модуля. Несмотря на то что модуль меню имеет всего лишь одну функцию, он позволяет полностью контролировать отображение всех модулей приложения. Далее рассмотрим структуру и работу каждого модуля отдельно.

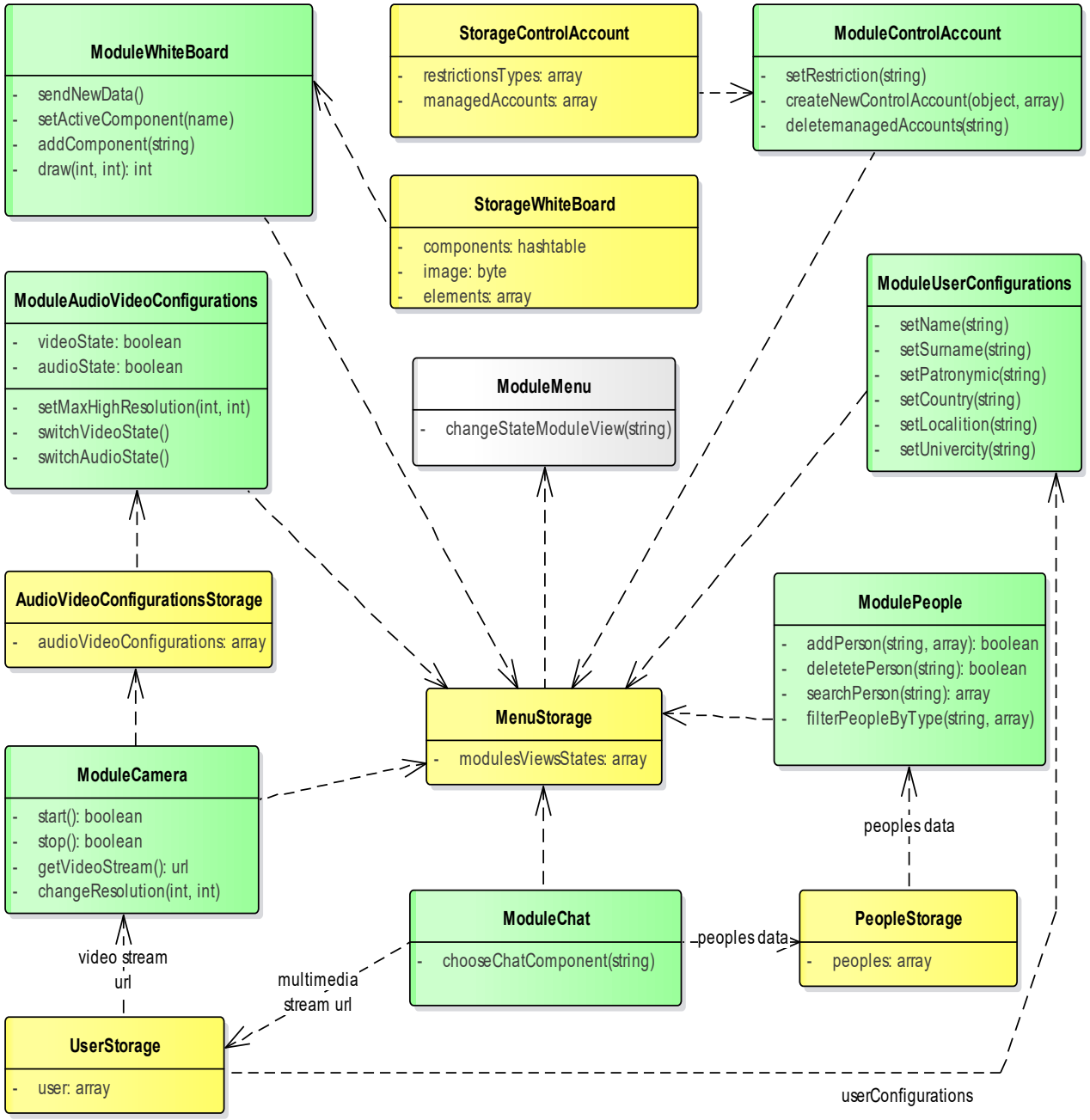


Рис. 3.2 Диаграмма взаимодействия модулей

После запуска приложения в графическом интерфейсе программы по умолчанию отображается модуль камеры, диаграмма взаимодействия с которым представлена на рисунке 3.3.

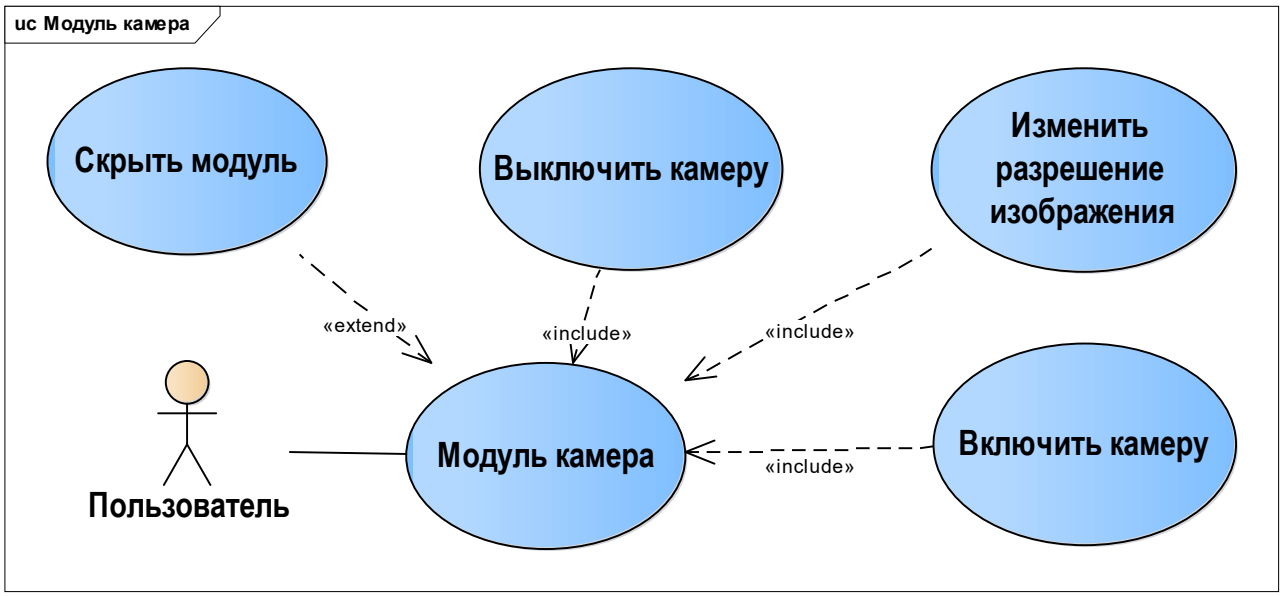


Рис. 3.3. Диаграмма модуля камеры

Для включения и выключения камеры используются методы start() и stop() соответственно. При вызове метода start() произойдет поиск необходимы устройства для захвата аудио-и видеопотоков, если устройства найдены, полученные данные преобразуются в единый мультимедийный поток, который будет доступен по некоторому url-адресу. Метод stop() позволяет прервать передачу данных с камеры и микрофона устройства, при этом происходит корректное завершение работы всех функций, осуществляющих формирование потоков данных и удаление объектов, хранящих данные о потоках. Для изменения разрешения изображения, которое необходимо захватывать с камеры, используется функция changeResolution(), параметрами которой служат необходимые значения разрешения по ширине и высоте. Последняя функция в модуле камеры представленная на рисунке 3.2 — getVideoStream(), обеспечивает получения url мультимедийного потока, который записывается в специальное хранилище состояний пользователей, для дальнейшего использования другими модулями. Для настройки аудио и видеоданных используется модуль, представленный на рисунке 3.4.

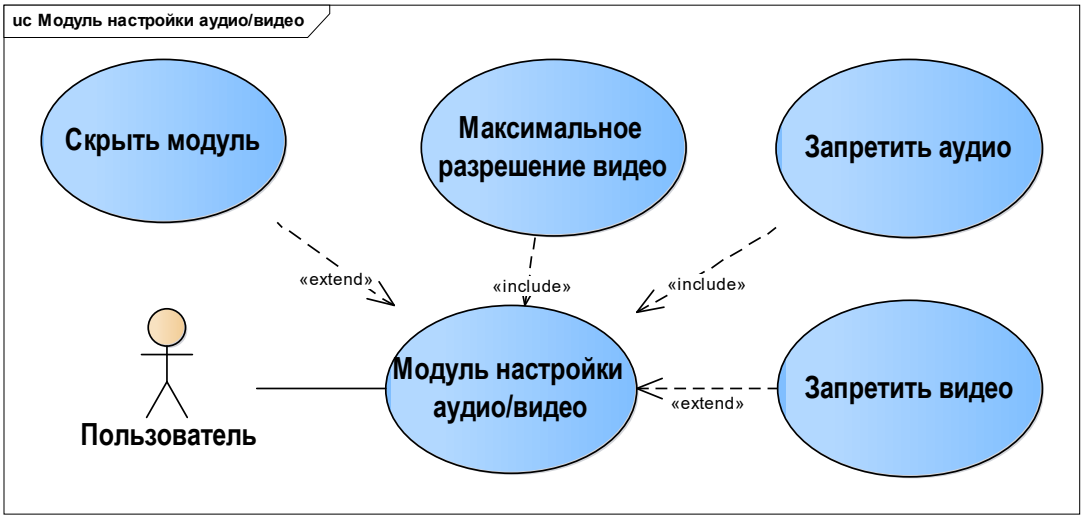


Рис. 3.4. Диаграмма модуля настройки аудио/видео

Данный модуль позволяет отключить захват видео- или аудиоданных с устройства при помощи функций switchVideoState() и switchAudioState() соответственно. Данная возможность позволяет ограничить передачу аудио- или видеоданных другому пользователю. По умолчанию камера всегда имеет определенное максимально захватываемое разрешение, которое равно 640 на 480 пикселей. Для изменения данного значения служит функция setMaxHighResolution(), получающая параметры максимальной ширины высоты, которое необходимо обеспечить на выходе видеопотока. Перед назначением максимального разрешения, функция setMaxHighResolution() проверит возможность получать данное разрешение с камеры и при положительном результате назначит его в качестве значения по умолчанию.

После активации камеры и микрофона устройства и формирования мультимедийного потока пользователь может совершать видеозвонки другим абонентам. Для совершения видеозвонков, пользователю необходимо иметь хотя бы одного друга у себя в профиле. Для поиска и добавления друзей используется модуль People, диаграмма которого представлена на рисунке 3.5.

Модуль пользователей обеспечивает поиск, добавление пользователей, а также удаление друзей и возможность их вызова. Для добавления и удаления пользователя из личного профиля служат функции addPerson() и deletetePerson() соответственно. На вход функции addPerson() могут быть переданы параметры

в виде строки для добавления одного пользователя либо массива для добавления группы пользователей.

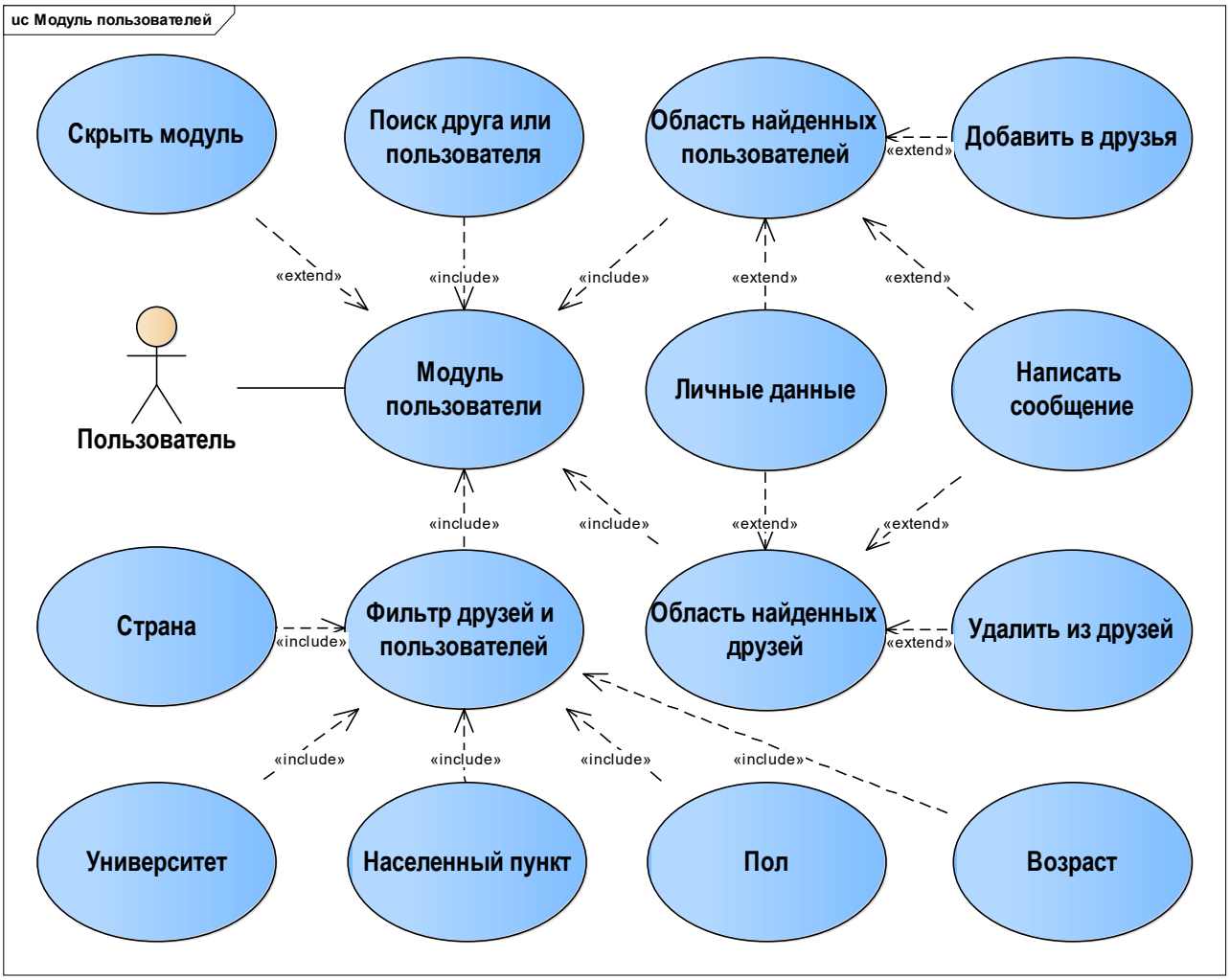


Рис. 3.5 Диаграмма модуля пользователей

Цикл добавления пользователя состоит из следующих шагов: считывание данных добавляемого пользователя, отправление данных на сервер для их обработки и записи в базу данных, получение ответа на клиенте об успешном добавлении пользователя в друзья, вывод новых данных в графическом интерфейсе. Функция deletetePerson() принимает на вход только строку и позволяет удалять пользователей по одному. Для фильтрации найденных пользователей и друзей служит функция filterPeopleByType(), представленная на рисунке 3.6.

Функция filterPeopleByType() на вход принимает значение типа, по которому будет происходить фильтрация, и массив пользователей, который

будет обработан. Данная функция возвращает массив отфильтрованных пользователей, который в дальнейшем будет отображен в графическом интерфейсе.

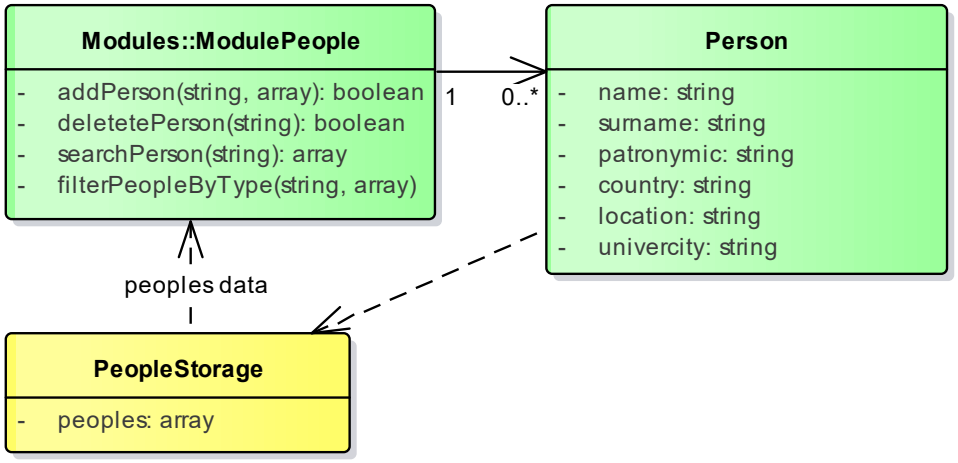


Рис. 3.6. Взаимодействие модуля пользователей с компонентами приложения

Из рисунка 3.6 видно, что каждый пользователь содержит некоторые данные о себе, которые позволяют его отфильтровать или идентифицировать. Последний компонент, представленный на рисунке 3.6, — это хранилище пользователей, которое содержит массив всех пользователей и друзей. Любые изменения в данном хранилище может производить модуль пользователей, обеспечивая хранение актуальной информации об остальных абонентах.

Добавленным в друзья пользователям можно произвести вызов для организации видеоконференцсвязи. За вызовы пользователей отвечает модуль «чат», диаграмма которого представлена на рисунке 3.7. На диаграмме модуля «чат» находится несколько основных компонентов: компонент выбора друга, компонент видео-чат, компонент текстовый чат. Основной компонент в модуле «чат» — выбор друга, который позволяет инициировать начало видеозвонка или текстового чата, а также прерывание видеовызовов. Компонент выбора друга активирует возможности компонентов TextChat и VideoChat посредством функций activateTextChat() и activateVideoChat(), представленных на рисунке 3.8.

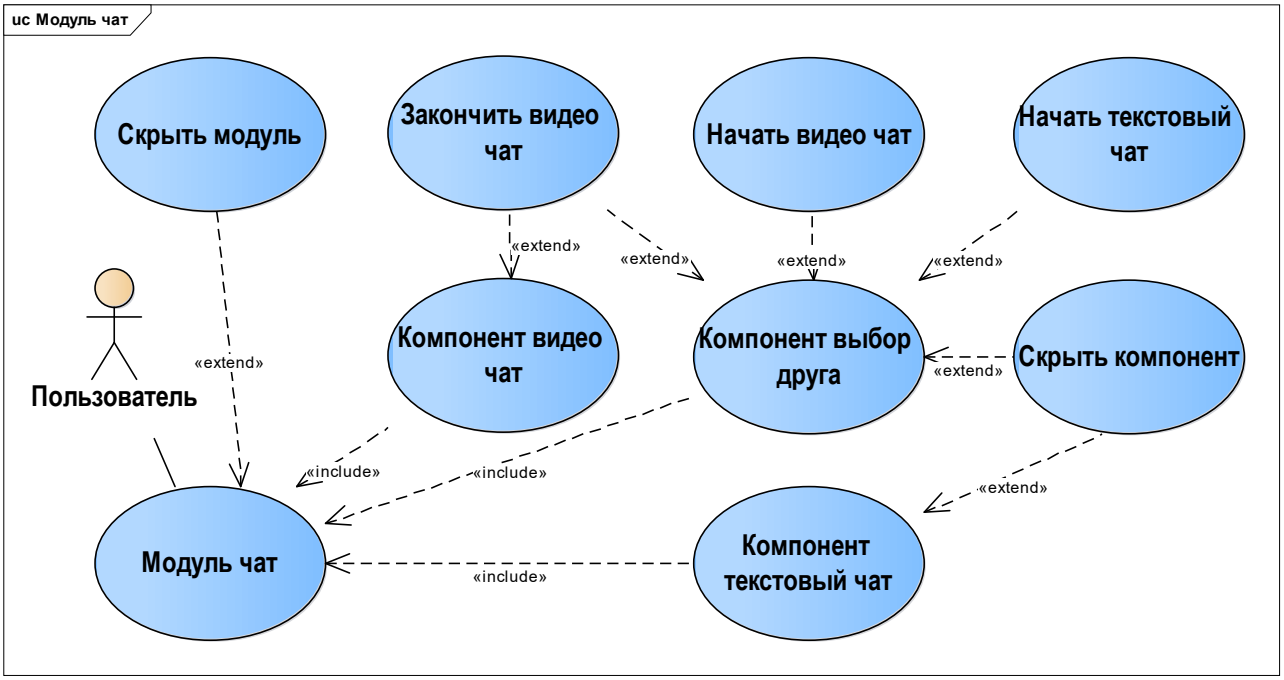


Рис. 3.7. Диаграмма модуля «чат»

Для разрыва соединения с абонентами вовремя видеочата компонент ChooseFriendArea использует функцию deactivateVideoChat(), которая обеспечит разрыв соединения со всеми абонентами, подключенными пиринговым соединением к клиентской части приложения в данный момент.

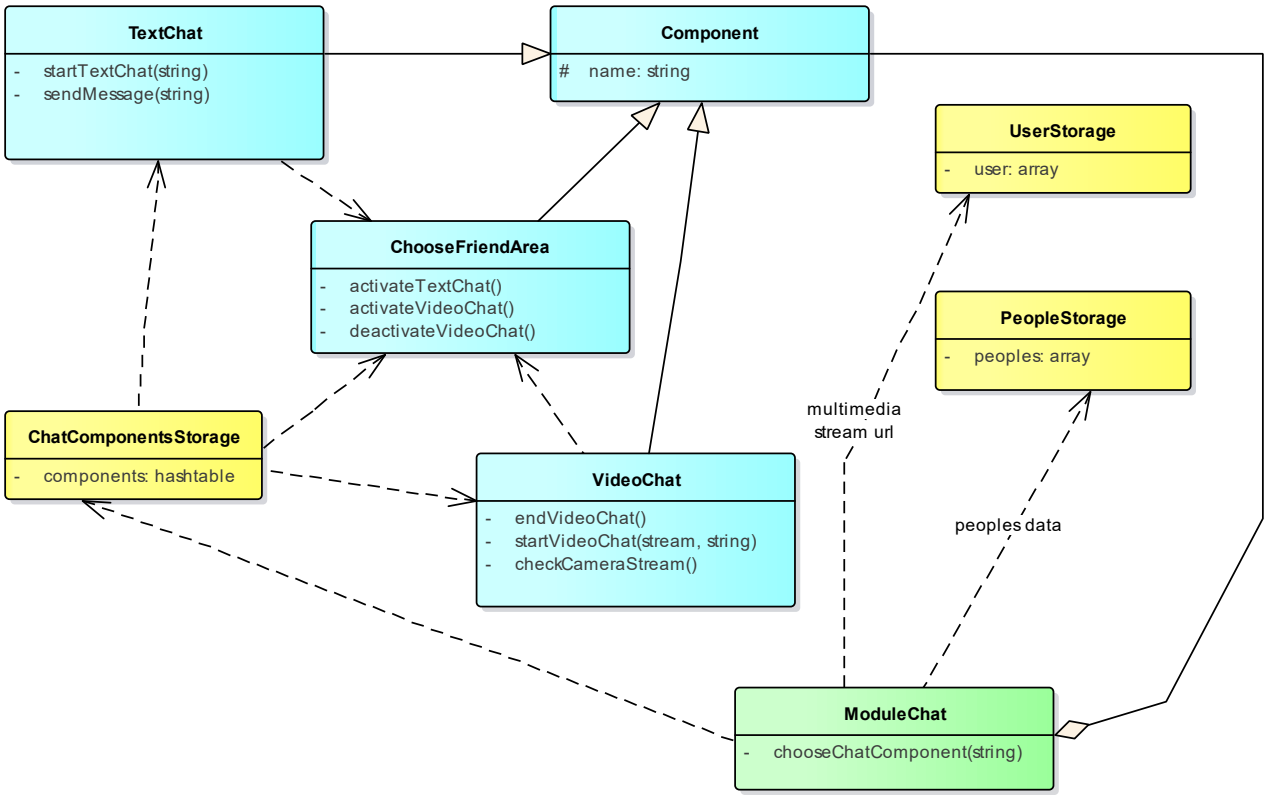


Рис. 3.8. Диаграмма взаимодействия модуля «чат»

Компоненты TextChat и VideoChat, изображенные на рисунке 3.8, имеют функции, обеспечивающие начало текстового чата и видеочата startTextChat() и startVideoChat() соответственно. Функция startVideoChat() на вход принимает поток мультимедийных данных клиента и идентификатор пользователя для соединения с ним в виде строки. Полный цикл соединения с другим абонентом для обеспечения видеоконференцсвязи состоит из следующих шагов: отправляется предложение об установлении пирингового соединения другому пользователю, приходит ответ о подтверждении и сигнальное сообщение, производится отправка ответного сигнального сообщения. На последнем этапе будет произведен обмен сообщениями с настройками для обеспечения пирингового соединения. После установки пирингового соединения клиенты начнут передавать мультимедийный потоки данных. Функция startTextChat() на вход принимает идентификатор другого клиента, который обеспечит открытие окна для диалога и загрузку истории сообщений для данного клиента. Для отправки сообщений другому пользователю используется функция sendMessage(), расположенная в компоненте TextChat. Данная функция на вход принимает значение имени пользователя, которому необходимо отправить сообщение. Функция позволяет отправлять данные как по пиринговому соединению, так и через сервер посредством протокола WebSocket. Перед тем как отправить сообщение другому пользователю, функция произведет проверку наличия соединения с ним. В случае наличия пирингового соединения, функция отправит сообщение по пиринговому каналу и на сервер для сохранения его в истории сообщений. Если пиринговое соединение отсутствует, функция sendMessage() отправит данные только по протоколу WebSocket на сервер для сохранения его в истории сообщений, и при наличии подключения у другого клиента к серверу сервер перешлет ему новое сообщение. Такой подход обеспечивает уменьшение нагрузки на сервер при наличии пирингового соединения между клиентами. Все компоненты в модуле «чат» наследуются от одного класса, что в конечном итоге обеспечивает простоту взаимодействия с ними в самом модуле. Например, при

необходимости визуализации одного из них при помощи функции chooseChatComponent(), которая на вход принимает имя необходимого компонента. Для обеспечения данных для различных компонентов модуль «чат» использует хранилища приложения. Данные по друзьям для компонента chooseFriendsArea предоставляет хранилище PeopleStorage. Для получения URL медиа потока используется хранилище UserStorage. Последнее хранилище ChatComponentsStorage необходимо для управления состоянием компонентов во время работы всего приложения. Таким образом, модуль чат оптимизирован для минимального взаимодействия с сервером и передачи данных по пиринговому протоколу.

Для настройки данных об аккаунте в приложении используется модуль настройки пользователя, диаграмма которого представлена на рисунке 3.9.

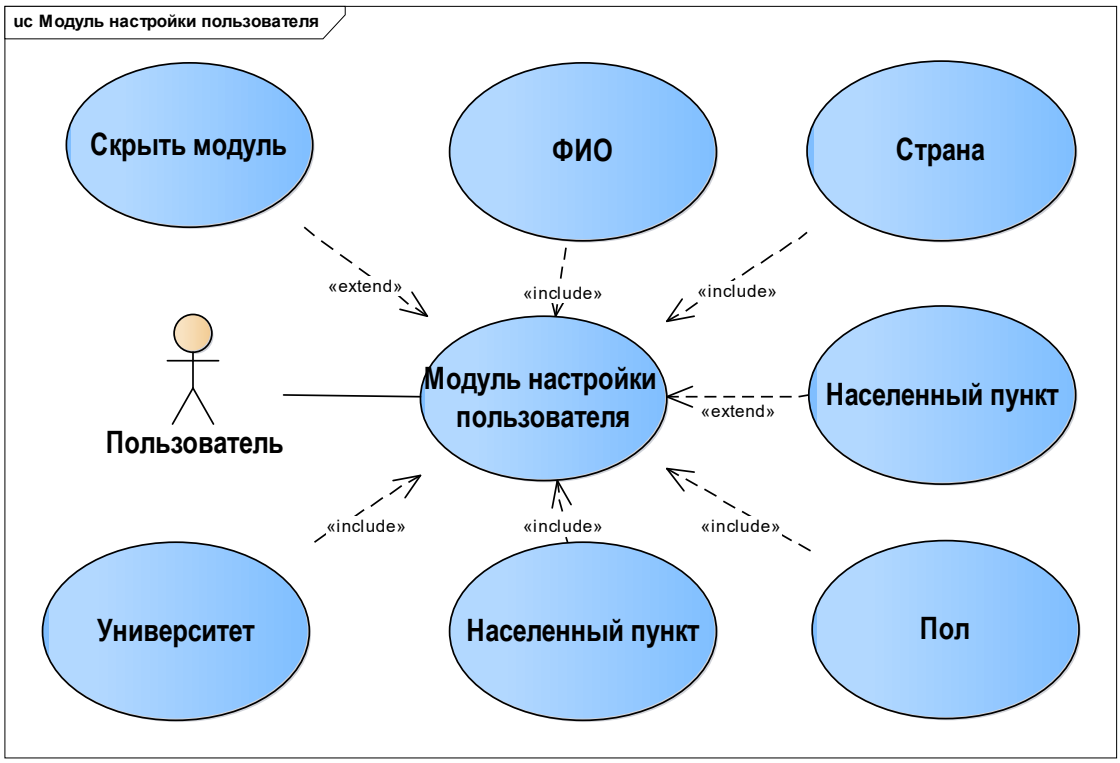


Рис. 3.9. Диаграмма модуля настройки пользователя

Модуль, представленный на рисунке 3.9, обеспечивает изменение текущей информации о пользователе, которая необходима для его поиска и фильтрации другими клиентами. Набор функций, присутствующих в данном модуле и начинающихся со словосочетания «set», имеют схожие задачи: считывают

данные с введенных полей настроек пользователя и сохраняют их в UserStorage. При изменении определенных полей в UserStorage новые данные пользователя отправятся на сервер, где будет произведено их сохранение в базе данных.

Для создания управляемого аккаунта в приложении используется модуль, диаграмма которого представлена на рисунке 3.10.

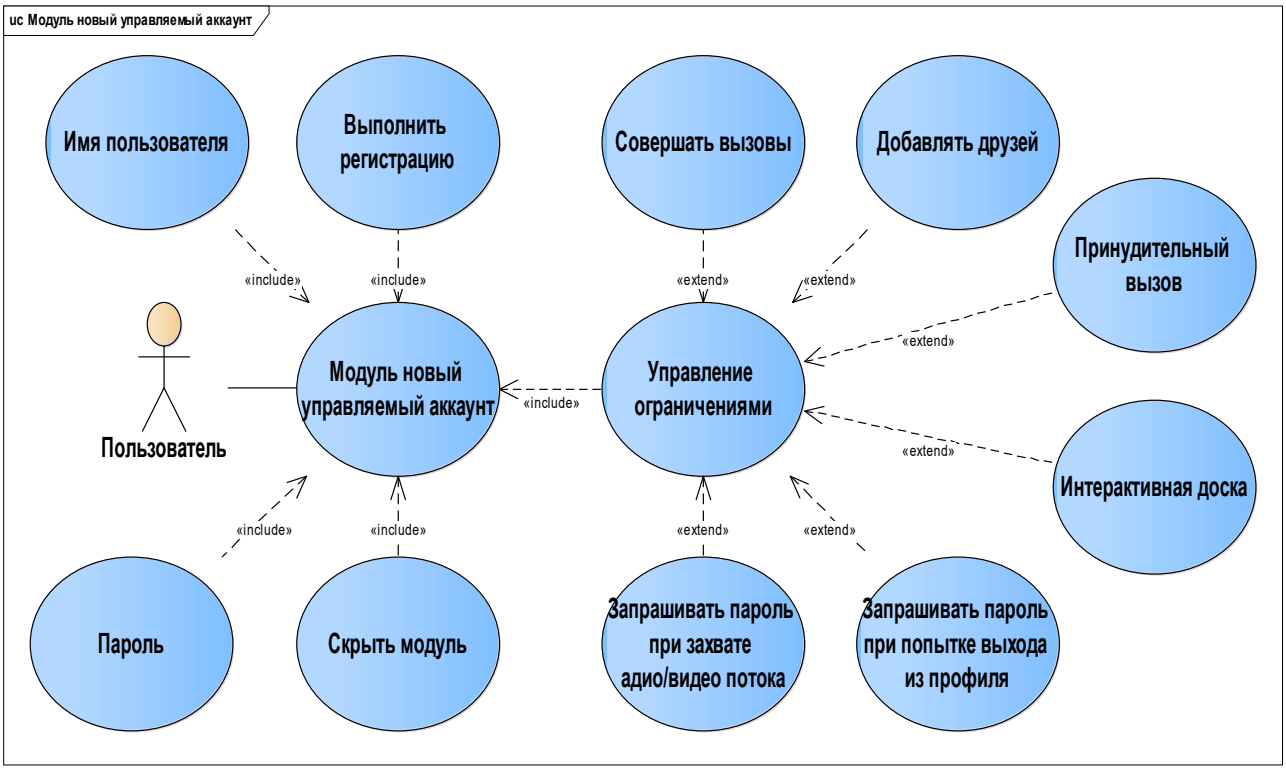


Рис. 3.10. Диаграмма модуля нового управляемого аккаунта

Данный модуль представляет собой систему регистрации обычного пользователя, но с определенными ограничениями. Для регистрации управляемого аккаунта используется функция createNewControlAccount(), которая принимает на вход два аргумента: параметры пользователя — логин и пароль, — ограничения. Ограничения устанавливаются функцией setRestriction(), которая на вход принимает имя ограничения и изменяет его текущее состояние на противоположенное. Все ограничения, представленные на рисунке 3.10, кроме ограничения на принудительное соединение, работают независимо от остальных. При условии, что ограничение совершения вызовов имеет положительное значение, ограничение принудительного соединения будет также установлено в положительное значение автоматически. Данное

свойство необходимо для обеспечения хотя бы одного варианта пирингового соединения. Пользователь, создавший аккаунт, может произвести удаление с данного аккаунта с помощью функции deletemanagedAccounts(). Функция deletemanagedAccounts() принимает на вход имя удаляемого аккаунта, затем данные отправляются на сервер, где происходит поиск и удаление записей об аккаунте.

Последний модуль, связанный с графическим интерфейсом, используемым в клиентской части приложения, является модуль интерактивной доски. Диаграмма модуля представлена на рисунке 3.11.

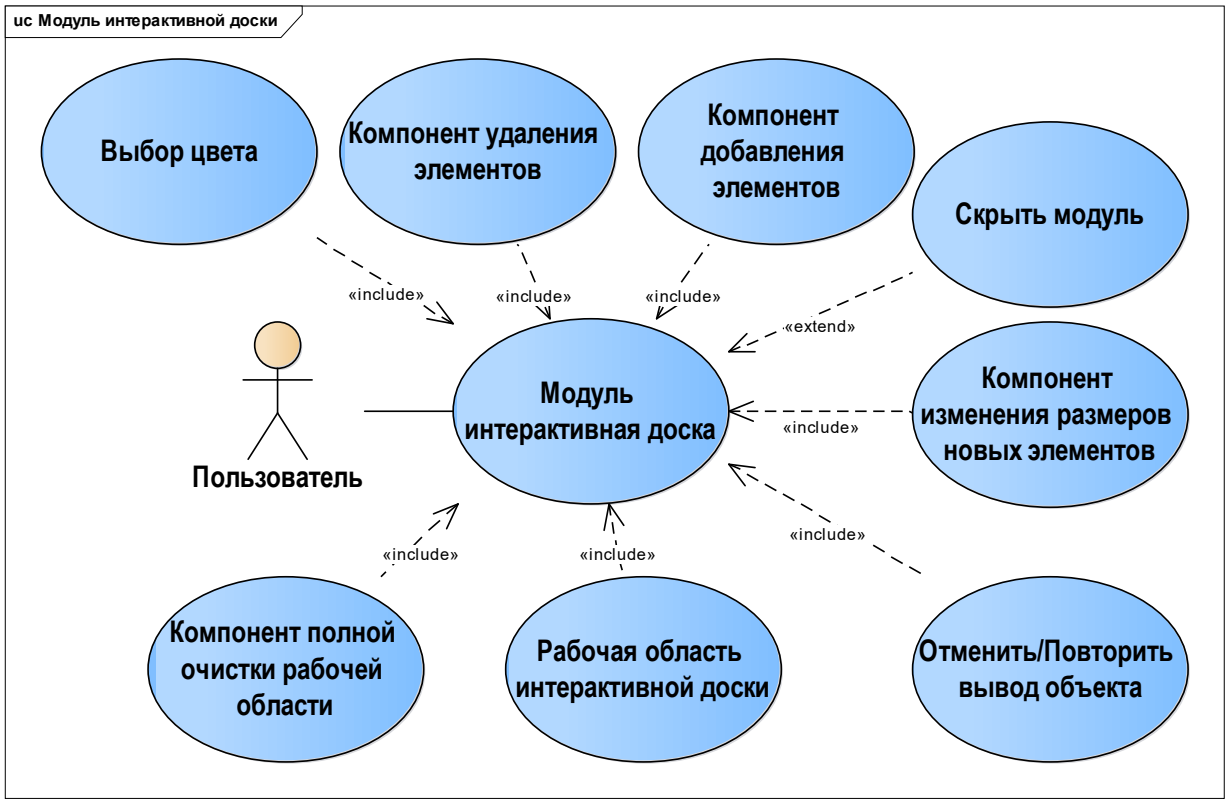


Рис. 3.11. Диаграмма модуля интерактивной доски

Данный модуль позволяет рисовать изображения удаленным пользователем в едином пространстве в режиме реального времени. Для рисования в данном модуле используются различные компоненты, которые обеспечивают формирование, изменение или удаление элементов. Из рисунка 3.12 видно, что все компоненты наследуются от класса Component. Класс Component включает в себя имя компонента и порядковый номер позиции в меню выбора компонентов. Имя позволяет сохранять данный

компонент в хранилище интерактивной доски и является ключом доступа для него, тем самым обеспечивая фильтрацию компонентов для получения и изменения их данных.

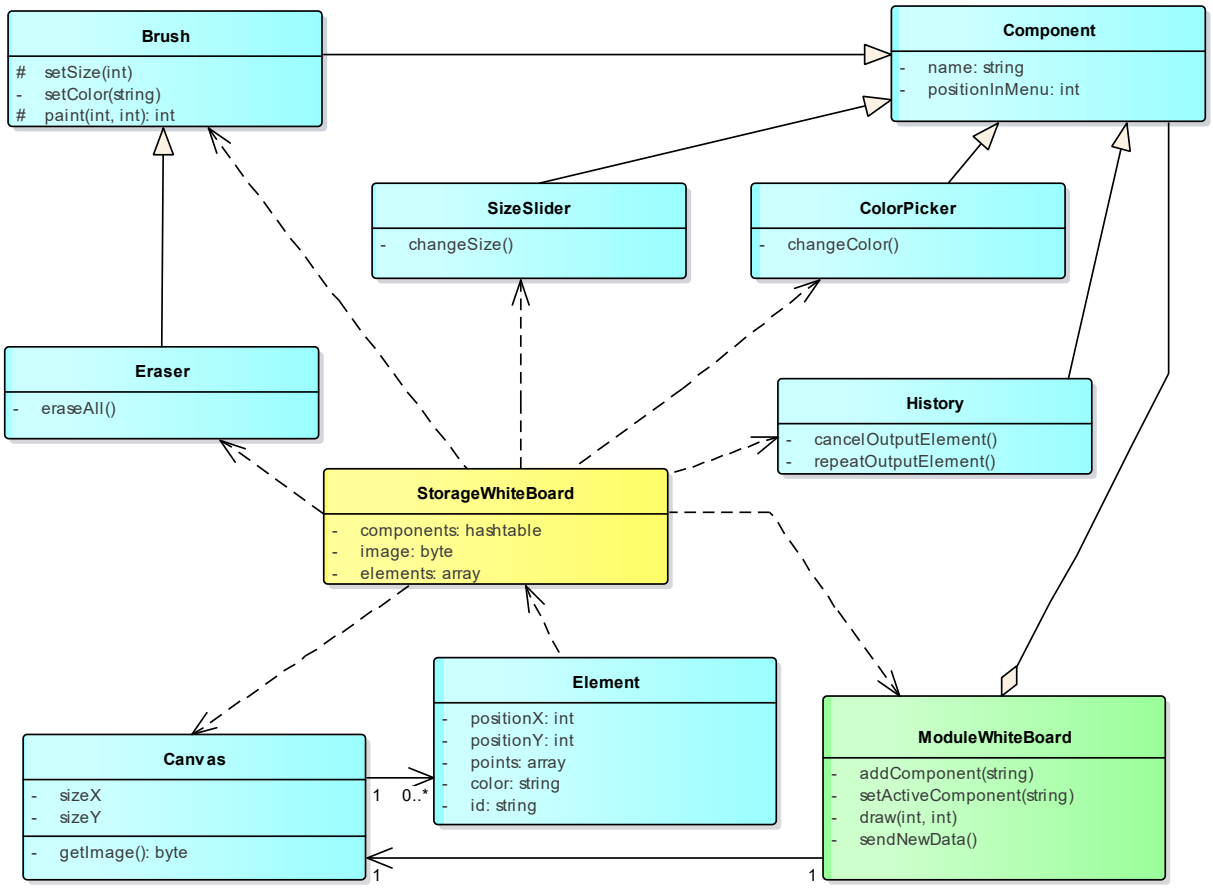


Рис. 3.12. Взаимодействие частей модуля интерактивной доски

Для того чтобы взаимодействовать с рабочей областью интерактивной доски, используются координаты курсора или касания объекта в случае сенсорных экранов. Рабочая область — класс Canvas — представляет собой пространство белого цвета с заранее заданными размерами и функцией getImage(), которая возвращает набор байтов, соответствующий закодированному текущему изображению на ней. Цель работы каждого компонента — проведение манипуляций с элементами в зависимости от переданных координат. Элемент интерактивной доски представляет собой объект со следующими параметрами: позиция расположения по осям x и y на рабочей области, массив точек и цвет. Для того, чтобы назначить активный компонент, используется функция setActiveComponent(), которая на вход

принимает имя компонента и назначает его активным в хранилище интерактивной доски. Активный компонент используется для манипуляции элементами при вызове функции draw(), которая на вход принимает значения координат курсора или касания объекта в рабочей области. После изменения рабочей области модуль интерактивной доски получит оповещение об изменении и новые данные от хранилища и вызовет функцию sendNewData(), которая отправит эти данные по пиринговому соединению всем подключенным в данный момент пользователям. Для отмены или повторения действия компонента используются функции cancelOutputElement() и repeatOutputElement() соответственно, входящие в состав модуля History. Компонент Brush с помощью своей функции paint() позволяет создавать различные элементы на рабочей области, задавая им размер и цвет функциями setSize() и setColor() соответственно. Компонент Eraser наследуется от компонента Brush и имеет функцию paint(), которая в данном случае не создает элементы, а удаляет. Дополнительная функция eraseAll(), входящая в состав данного компонент, позволяет удалить все элементы с рабочей области. Компоненты SizeSlider и ColorPicker позволяют задать цвет и размер элементов, которые будут сохранены в хранилище состояний интерактивной доски для дальнейшего их использования в компонентах Brush и Eraser. Следовательно, используя все вышеперечисленные компоненты, можно обеспечить полноценную рабочую область для изображения различных элементов в режиме реального времени.

Последнее, что необходимо отметить — модуль передачи данных серверу, который не был отображен в общей структуре для ее упрощенного вида. Данный модуль имеет очень простую структуру, состоящую из функции соединения с сервером и функций передачи данных по определенным каналам. Функции передачи данных по различным каналам имеют одинаковый принцип работы, который заключается в отправке информационных сообщений с определенным строковым идентификатором. Модуль имеет некоторые простые настройки, которые позволяют ему произвести повторное подключение к

серверу при его разрыве, а также настроить количество и время данных подключений, которые необходимо повторять в случае отсутствия соединения.

3.3. Программные средства серверной части приложения

Архитектура серверной части приложения видеоконференцсвязи, рассмотренная в разделе 2.4, содержит в себе четыре модуля: модуль взаимодействия с клиентами, модуль обработки данных клиента, модуль API для облачного взаимодействия и модуль взаимодействия с базой данных. Для взаимодействия с клиентами используются два протокола: HTTP и WebSocket. Клиент отправляет некоторую информацию по одному из протоколов, в зависимости от текущего состояния приложения, затем информация поступает на сервер, где ее получает модуль взаимодействия с клиентами, представленного на рисунке 3.13.

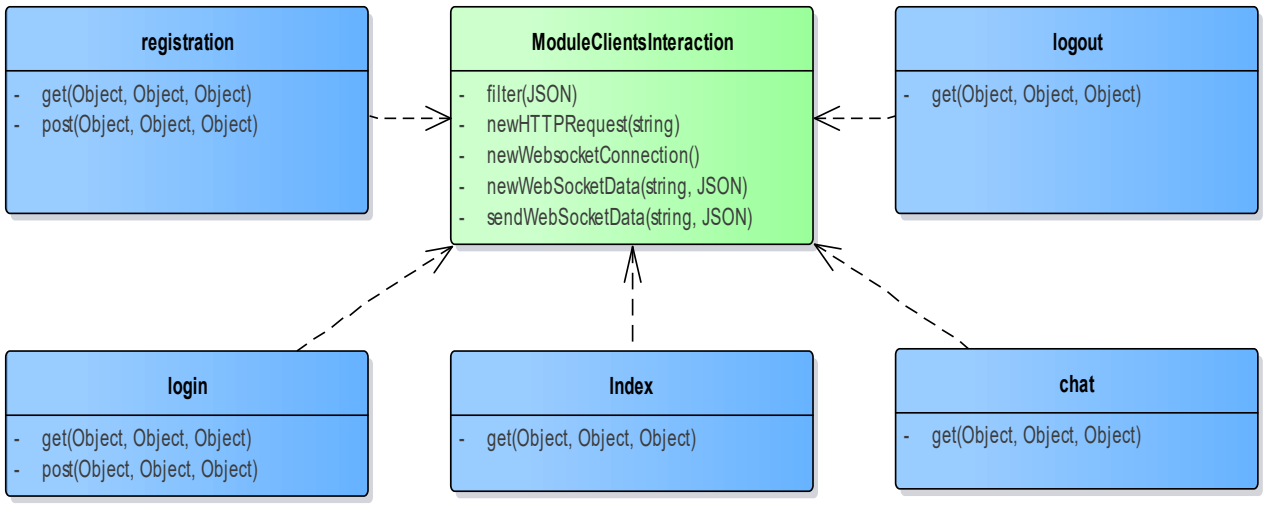


Рис.3.13. Модуль и компоненты для взаимодействия с клиентом

Данный модуль обеспечивает прием, отправку, проверку и передачу информации другим модулям. Прием информации осуществляют функции newHTTPRequest() и newWebSocketData() для протоколов HTTP и WebSocket соответственно. Для проверки информации используется функция filter(), которая на вход получает некоторый набор данных в формате JSON, который проверяется на отсутствие символов, не удовлетворяющих запросу. Для получения различных страниц и обмена данными по протоколу HTTP используются дополнительные компоненты, которые обеспечивают обработку

различных HTTP запросов. Для получения страниц чата, входа, регистрации и главной, используются компоненты chat, login, registration и index соответственно, представленные рисунке 3.13.

Выбор компонента, которому необходимо передать запрос осуществляет функция newHTTPRequest(), на вход которой поступает сам запрос. Компоненты имеют функцию get(), которая принимает значения req, res, next и отвечает на HTTP запрос типа GET. Значение req — это объект обработанного запроса, который включает всю необходимую служебную информацию о запросе в виде ключ-значение данных, включенных в req. Объект res используется для ответа по протоколу HTTP, в данном случае он при помощи функций render() или redirect() позволяет отобразить или перенаправить на необходимую страницу соответственно. Последний объект — next позволяет осуществить передачу информации и рабочего потока сервера другим его частям. Если выполняется функция render() или redirect(), содержащиеся в объекте res, то рабочий поток автоматически передается другому модулю.

Метод post(), содержащийся в компонентах login и registration, отвечает на HTTP запросы типа POST. Данный метод, в отличие от get(), предназначен для обработки некоторой дополнительной информации, отправляемой вместе с телом запроса; в данном случае это информация из полей для ввода данных пользователя. Метод post() также принимает три типа параметров — req, res, next, которые выполняют те же функции, что и в методе get(), за исключением параметра res. Параметр res в данном случае отсутствует, так как вместо функции render() используется функция send(), которая позволяет отправить некоторые данные клиенту в ответ на запрос.

В функциях get() и post() в параметре req, содержится функция isAuthenticated(), которая проверяет наличие аутентификации у пользователя. Если пользователь аутентифицирован, то при GET запросе на страницу логина и регистрации происходит перенаправление на страницу чата, а главная страница отображает в меню ссылку на страницу чата и скрывает ссылки на страницу логина и регистрации. Если пользователь не аутентифицирован, то при GET

запросе на страницу чата, произойдет вызов функции redirect(), которая перенаправит клиента на страницу логина, а на главной странице, страницах логина и регистрации будут отображены ссылки на данные страницы.

Функции post(), реализованные в компонентах login, и registration, также вызывают функцию redirect() при успешно проведенных операциях входа и регистрации, для перенаправления на страницы чата и логина соответственно. В случае возникновения ошибки при осуществлении входа или регистрации данные компоненты в ответ на запрос типа POST сформируют страницу с информацией об ошибке при помощи функции render(). Компонент logout реализует логику работы только функции get(), которая осуществляет удаление сессии пользователя и перенаправление клиента на главную страницу.

После осуществления входа при помощи функции post() компонента login для пользователя создается рабочая сессия, в которой будут доступны некоторые данные пользователя для осуществления аутентификации по протоколу WebSocket. После удачного входа пользователь попадет на страницу чата, которая автоматически инициирует соединение с сервером по протоколу WebSocket. Сервер прослушивает все попытки соединения по данному протоколу при помощи функции newWebsocketConnection(), которая при наличии сессии у данного клиента авторизует его сокет и осуществит соединение.

В случае отсутствия сессии будет осуществлен отказ от соединения с данным клиентом. Присоединенные к серверу клиенты могут отправлять и получать данные от сервера, которые прослушиваются и передаются функциями newWebSocketData() и sendWebSocketData() соответственно. Данные функции имеют следующие входные параметры — строку типа данных и данные в формате JSON. Строка типа данных позволяет легко определить на клиенте и сервере, для чего были предназначены переданные данные без осуществления дополнительных операций парсинга. Пересылаемые данные между клиентом и сервером, упакованные в формат JSON, позволяют сократить передаваемый трафик и осуществлять их быстрое преобразование в

формат JavaScript объекта. После получения некоторых данных и их фильтрации на стороне сервера, они отправляются на обработку модулю обработки данных клиента, представленного на рисунке 3.14.

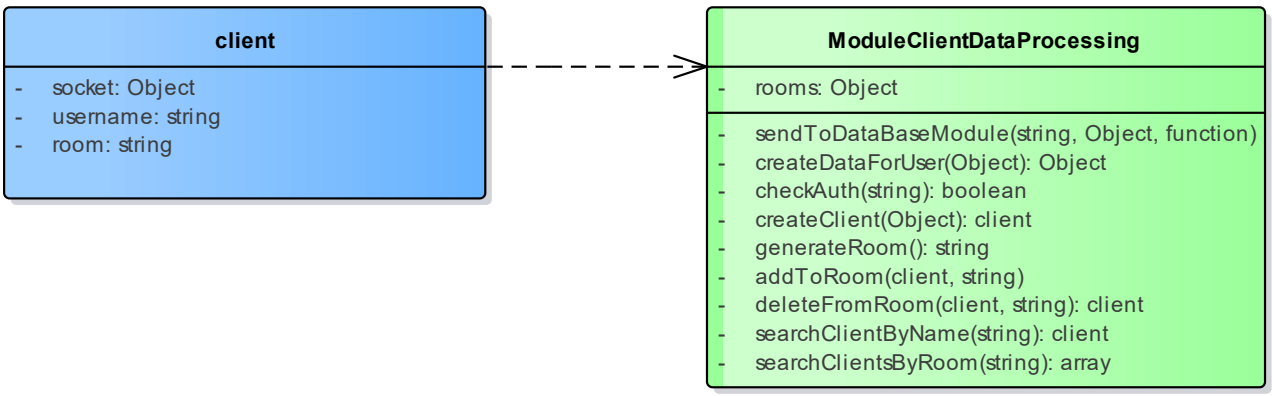


Рис. 3.14. Модуль и компонент обработки данных клиента

Модуль позволяет на основе полученных данных принимать решение об их дальнейшем использовании и хранении. После того как сервер получил данные для регистрации клиента и произвел их проверку, они отправляются в модуль обработки данных клиентов, и при помощи функции sendToDataBaseModule() будут переданы в базу данных для дальнейшей обработки. База данных обработает информацию и вернет ответ в функцию, которая является третьим параметром функции sendToDataBaseModule(). Данный ответ будет переформатирован при помощи функции createDataForUser() для дальнейшей отправки клиенту. Функция createDataForUser() на вход принимает некоторый объект, который вернула база данных, а возвращает другой объект, в котором содержится информация, необходимая для отправления клиенту или для формирования страницы приложения. Для осуществления обработки данных авторизации используются те же функции, что и при регистрации. После того как данные для авторизации были проверены на соответствие в базе данных и получен положительный ответ, callback() функция сформирует cookie данных для клиента и создаст новую сессию, что будет использовано для идентификации клиента на сервере без использования повторной авторизации.

После авторизации клиента происходит соединение с сервером по протоколу WebSocket, где модуль обработки данных клиента проверяет наличие cookie данных авторизации и их соответствие существующей сессии при помощи функции checkAuth(), которая возвращает логическое значение типа boolean. Если функция checkAuth() вернула false, соединение с клиентом не будет установлено. В случае удачной проверки авторизации модуль обработки данных клиента сформирует нового клиента при помощи функции createClient() с его личными данными, переданными в виде первого аргумента функции. Каждому клиенту при соединении с сервером по протоколу WebSocket создается собственная комната с помощью функции generateRoom(). Данная функция генерирует имя комнаты, которое затем будет использовано как ключ для добавления нового клиента в объект rooms. После авторизации соединения по протоколу WebSocket, генерации комнаты и добавления в нее клиента клиент может производить звонки другим пользователям. Когда один из клиентов инициирует соединение, модуль обработки данных клиента попытается найти вызываемого клиента в хэш объекте rooms при помощи функции searchClientByName(). В случае успешного поиска вызываемому клиенту будет отправлен запрос на соединение.

После соединения двух клиентов модуль обработки данных клиента переместит их в общую комнату и проверит наличие в старой и новой комнате других клиентов с помощью функции searchClientsByRoom(). На вход данная функция получает название комнаты, а на выходе отдает массив клиентов, содержащихся в данной комнате. В случае если функция нашла других клиентов в одной из комнат, то модуль обработки данных клиента начнет принудительное соединение всех найденных клиентов друг с другом. Для обработки других данных, предназначенных для сохранения пользовательских настроек, создания контролируемых аккаунтов, удаления и поиска друзей и другой информации по базе данных, используется функция sendToDataBaseModule(). Данная функция первым аргументом принимает тип производимой операции, которая позволяет идентифицировать, каким образом обрабатывать запрос. Вторым параметром

функции является информация, которую необходимо обработать базой данных. Последним параметром является функция, которая получит ответ от базы данных по итогу проведения операции.

Для совершения операций с базой данных в серверной части приложения используется модуль взаимодействия с базой данных. Данный модуль реализует запросы в базу данных при помощи ORM Mongoose. ORM Mongoose является технологией программирования, которая связывает базы данных с концепциями объектно-ориентированного языка программирования JavaScript, создавая «виртуальную объектную базу данных». На основе Mongoose создана схема пользователя, которая повторяет поля пользователя в базе данных. Данная схема удобна тем, что легко преобразует структуру базы данных в объект, с которым удобно осуществлять взаимодействие на сервере.

Рассматриваемая схема пользователя, представленная на рисунке 3.15, включает следующие поля: адрес электронной почты; имя, фамилия и отчество пользователя; зашифрованный пароль; hash и salt для шифрования и расшифровки пароля; время создания записи; массив ссылок на друзей. Почти все перечисленные поля имеют тип строка, кроме массива ссылок на друзей, который указывает на информацию о других пользователях в базе данных и времени создания записи с типом дата, которая создается автоматически базой данных после добавления нового пользователя.

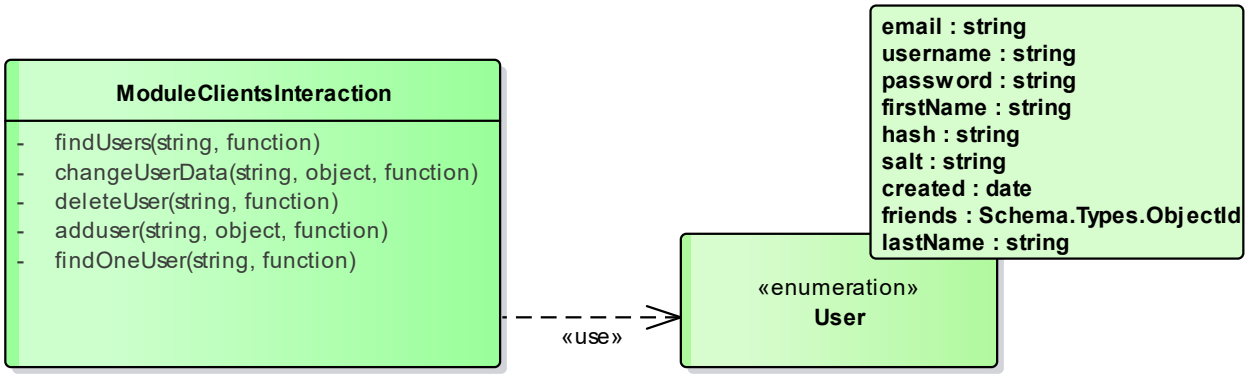


Рис. 3.15. Модуль и схема для взаимодействия с базой данных

Для поиска и удаления одного пользователя используются функции findOneUser() и deleteUser() соответственно. Данные функции принимают на

вход два параметра — имя пользователя и функция для возврата значения после запроса. Функция findUsers() первым параметром принимает некоторую строку, которая должна содержать часть или полное имя пользователя, второй параметр — функция для возврата значения, которая в данном случае будет принимать на вход массив найденных клиентов. Функции addUser() и changeUserData позволяют добавить нового пользователя и изменить его данные соответственно. Первым параметром данных функций является имя пользователя, затем идет объект данных, который необходимо изменить или добавить, последним параметром является функция возврата значения, которая позволяет убедиться в корректности проведенной операции.

Внутри модуля взаимодействия с базой данных создан механизм выбора типа запроса для идентификации и корректной обработки данных пришедших от другого модуля. Благодаря не сложному функционалу данный модуль позволяет организовать полный цикл взаимодействия с базой данных, обеспечивая к ней безопасный доступ другим модулям и отслеживая корректность выполнения запросов.

Для обеспечения доступа к данным другими серверами используется модуль API для облачного взаимодействия. Этот модуль позволяет получить функционал страницы чата по протоколу WebSocket, чтобы отобразить ее в составе другого клиентского приложения. В таком случае передача страницы произойдет не напрямую от сервера видеоконференцсвязи, а от промежуточного сервера, который обеспечивает взаимодействие с клиентом. За формирование и передачу страницы отвечает функция getChatPage(), представленная рисунке 3.16, которая на вход получает cookie данные в виде строки и при ее успешном выполнении вернет данные в формате HTML.

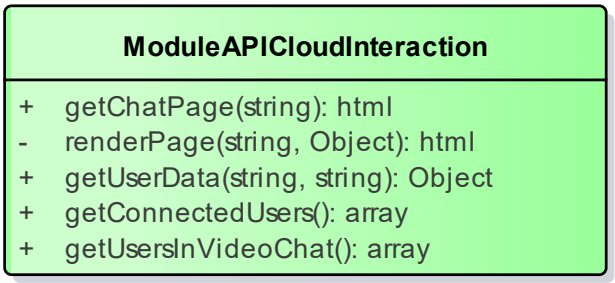


Рис. 3.16. Модуль API для облачного взаимодействия

Функция getChatPage() тесно взаимодействует с модулем обработки данных клиента, в который она отправляет cookie данные для проверки авторизации, затем, когда авторизация подтверждена, данная функция сформирует страницу на основе информации, полученной из модуля взаимодействия с базой данных. В конечном итоге эта функция сформирует HTML страницу с помощью вспомогательной функции renderPage(), которая на вход первым аргументом принимает строку тип шаблона, необходимого для вывода в HTML, а вторым объект, содержащий информацию для наполнения шаблона.

Модуль API для облачного взаимодействия позволяет также получать данные о пользователях из базы данных и о подключенных в данный момент к серверу. Для получения данных из базы данных используется функция getUserData(), которая на вход получает тип необходимых данных и имя пользователя, а на выходе, при удачном выполнении запроса в базу данных с помощью модуля взаимодействия с базой данных, объект содержащий необходимые данные. Функции getConnectedUsers() и getUsersInVideoChat() возвращают массивы всех присоединенных пользователей и пользователей, которые соединены в видеочате на данный момент. Эти данные могут быть полезны для аналитики и построения алгоритмов взаимодействия с пользователем в режиме реального времени. В конечном итоге данный модуль предоставляет функционал для взаимодействия с сервером, который обеспечивает различную актуальную информацию о пользователях и других данных, связанных с ними.

3.4. Выводы

1. Рассмотренные программные средства приложения видеоконференцсвязи обеспечивают коммуникацию клиентской и серверной частей приложения и поддерживают кроссплатформенность за счет реализации веб-интерфейса с использованием языков HTML, CSS и JavaScript.

2. С помощью UML диаграммы показаны возможные варианты выбора модулей графического интерфейса в клиентской части приложения; порядок взаимодействия модулей; а также функционирование каждого из модулей доступных пользователю в графическом интерфейсе.

3. Разработанные программные средства серверной части приложения видеоконференцсвязи содержат четыре основных модуля: модуль взаимодействия с клиентами, модуль обработки данных клиента, модуль API для облачного взаимодействия и модуль взаимодействия с базой данных. Для обеспечения взаимодействия с клиентами используются два протокола: HTTP и WebSocket. Пересылаемые данные между клиентом и сервером упакованы в формат JSON, позволяющий сократить передаваемый трафик и осуществить быстрое преобразование в формат JavaScript объекта.

Глава 4. Экспериментальная проверка разработанного приложения видеоконференцсвязи

Тестирование разработанного приложения — это комплекс процедур, который охватывает различные составляющие приложения [87]. Основными направлениями тестирования приложения является: нагрузочное тестирование, тестирование логики работы и корректности отклика графических элементов приложения [88]. Для тестирования графического интерфейса и логики работы приложения было использовано функциональное тестирование. Данный вид тестирования применяется на любых стадиях разработки приложения, тем самым помогая выявить проблемные места уже на начальных этапах разработки. Функциональное тестирование заключается в имитации фактического использования системы, где результат имитации должен быть сопоставлен с заранее известными данными результата [89]. В разработанном приложении для проведения функционального тестирования была использована библиотека Mocha [90], которая предоставляет полный функционал для тестирования и отображения результатов проведенных тестов.

Чтобы собрать показатели и определить производительность разработанной системы в данной работе используется нагрузочное тестирование. Данный вид тестирования используется, чтобы установить работоспособность и отклик системы при нагрузках в режиме реального времени. Далее рассмотрим подробнее алгоритмы и результаты функционального и нагрузочного тестирований.

4.1. Алгоритмы и результаты функционального тестирования частей приложения

Функциональное тестирование разработанного приложения разделено на три части: 1) тестирование графического интерфейса в клиентской части приложения [91]; 2) тестирование логики работы клиентской части [92]; 3) тестирование логики работы серверной части [92]. Такое разделение

позволило более качественно протестировать различные составляющие системы и выявить некорректную работу различных компонентов.

Для функционального тестирования графического интерфейса в клиентской части приложения используется алгоритм, представленный на рисунке 4.1.

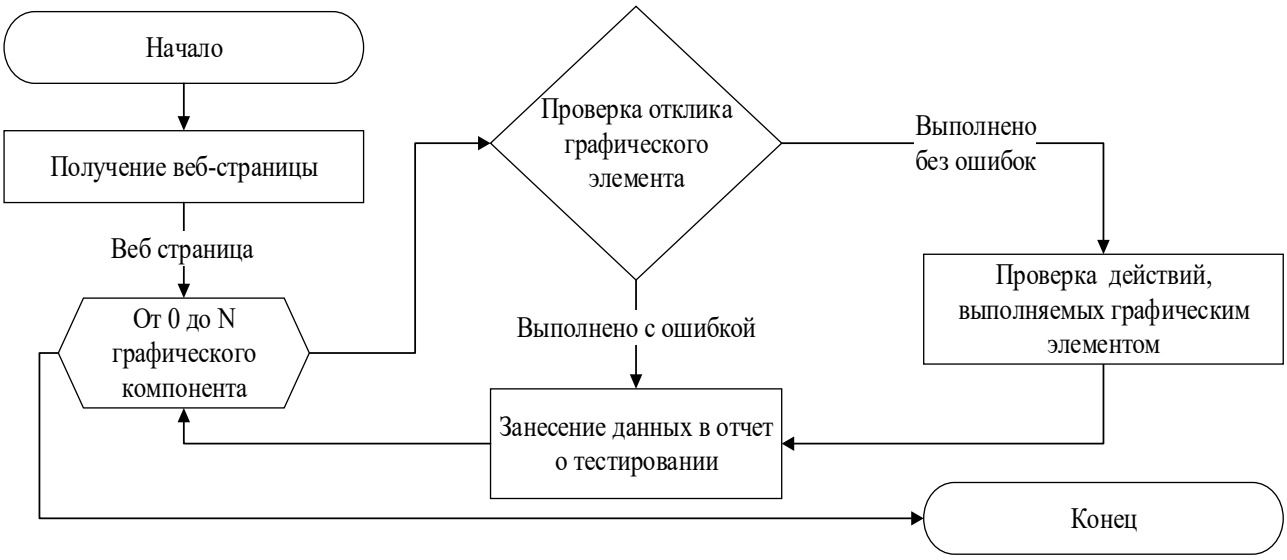


Рис. 4.1. Алгоритм функционального тестирования графического интерфейса клиентской части приложения

В ходе работы данного алгоритма тестирования происходит формирование одной из страниц приложения. Следующим этапом идет выбор графического компонента для проверки отклика при взаимодействии с пользователем. Данный этап подразумевает формирование некоторого программного события при взаимодействии пользователя с элементом графического интерфейса [93]. Сформированное программное событие проверяется функцией тестирования, которая сначала произведет эмуляцию взаимодействия с компонентом, а затем попытается перехватить данное событие. В случае неудачной попытки перехвата события, алгоритм перейдет к тестированию следующего элемента, при этом информация о неправильной работе элемента будет занесена в отчет о тестировании. В случае перехвата правильного типа события произойдет переход на следующий этап тестирования, где будет произведена проверка действий, выполняемых графическом элементом. На данном этапе будет запущена функция, которая получит событие с данными от графического

компонента и передаст их на обработку соответствующей функции приложения, производящей необходимые манипуляции с данными. При проверке действий, выполняемых графическим элементом, функция тестирования будет ожидать данные от приложения, которые должны совпасть с заранее сформированной корректной информацией. Из-за асинхронной архитектуры приложения в функцию тестирования введен еще один параметр, который ограничивает время ожидания ответа от приложения. Данный параметр гарантирует корректное завершения тестирования в случае, если приложение не генерирует ответ функции на тестовые данные во время определенного промежутка времени. После проверки действий, выполняемых графическим элементом, следует операция занесения данных в отчет о тестировании. После того, как все элементы страницы были проверены, тестировщик может посмотреть отчет в браузере в виде веб-страницы. Графический интерфейс отчета показывает тестировщику места, где тестирование было провалено, красным цветом и зеленым цветом помечает те этапы для каждого графического элемента, где тестирование было проведено успешно. На основе этих данных тестировщик может сделать вывод о проблемах работы приложения и отправить эти данные разработчикам для их исправления.

Для функционального тестирования логики работы серверной и клиентской частей используется следующий алгоритм, приведенный на рисунке 4.2.

Рис. 4.2. Общий алгоритм функционального тестирования логики работы клиентской и серверной частей приложения

Работа алгоритма, представленного на рисунке 4.2, начинается с выбора одной из тестируемых функций. Затем происходит ее вызов с аргументами в виде тестовых данных, которые помогают заранее предопределить итог работы данной функции. Следующим этапом работы алгоритма происходит сравнение итога работы функции с заранее предопределенными данными. Результат сравнения покажет корректность выполнения теста, который будет занесен в отчет о тестировании. Данный алгоритм абсолютно идентичен для тестирования как серверной, так и клиентской частей логики работы приложения, что способствует упрощению тестирования и сохранению однородного стиля написания тестов. Результаты функционального тестирования приведены в таблице 4.1.

Таблица 4.1 Результаты функционального тестирования

Тестируемая часть	Версия приложения					
	1	2	3	4	5	6
	Количество тестируемых функций и результаты					
Графический интерфейс клиентской части приложения	Всего - 25, 23 рабочих	Всего - 28, 27 рабочих	Всего - 32, 32 рабочих	Всего - 37, 37 рабочих	Всего - 41, 38 рабочих	Всего - 45, 44 - рабочих
Логика работы клиентской части приложения	Всего - 105, 96 рабочих	Всего - 150, 130 рабочих	Всего - 243, 128 рабочих	Всего - 274, 269 рабочих	Всего - 278, 275 рабочих	Всего - 283, 280 - рабочих
Логика работы серверной части приложения	Всего – 58, 56 рабочих	Всего – 87, 83 рабочих	Всего - 99, 95 рабочих	Всего - 118, 117 рабочих	Всего - 120, 117 рабочих	Всего - 123, 120 рабочих

Как видно из таблицы 4.1, в ходе разработки приложения было проведено шесть итераций тестирования каждой из частей приложения. Каждая итерация тестирования была направлена на выявление проблем на различных этапах разработки приложения. В ячейках таблицы приведено количество всех тестируемых функций и функций, которые удачно прошли проверку. Все неработоспособные функции были отправлены на доработку. Ошибки, возникавшие в ходе работы данных функций, были устранены к следующему этапу тестирования приложения. Функциональными тестами невозможно

покрыть работу всех функций приложения, поэтому для проведения тестирования были выбраны функции и графические компоненты в наиболее сложных местах работы приложения. Каждая следующая версия приложения содержит новые разработанные функции, которые необходимо протестировать. Из таблицы 4.1 видно, что в некоторых версиях программы количество неработоспособных функций увеличивалось. Данная проблема связана с тем, что некоторые тестируемые функции включают в себя вызов других тестируемых функций. Таким образом, при некорректной работе включенной тестируемой функции функция, которая является оборачивающей, также выполняется некорректно. Следовательно, если в предыдущей версии приложения была работоспособная функция, то при включении в нее новой тестируемой версии приложения другой некорректно выполняющейся функции, количество работоспособных функций в новой может уменьшиться относительно предыдущей версии приложения. После тестирования шестой версии приложения были устранены все выявленные проблемы. Дальнейшее тестирование направленно на выявления поведения системы и ее отклика в момент нагрузки.

4.2. Оценка новых графических интерфейсов для модуля контролируемого аккаунта и модуля интерактивной доски

Несмотря на линейную сложность алгоритмов и схожесть либо идентичность некоторых из них, необходимо учитывать сложность проектирования графических интерфейсов приложения, работу которых можно оценить при помощи нагрузочного тестирования. Графические интерфейсы позволяют осуществлять взаимодействие между пользователями и клиентской частью приложения, запуская на выполнение различные алгоритмы.

В разработанной системе существуют два дополнительных модуля, которые не являются частью основного функционала приложения, но позволяют осуществлять дополнительные операции. Данные модули — модуль контролируемого аккаунта и модуль интерактивной доски — позволяют

создавать аккаунты, управлять которыми может один пользователь, администратор, и обмениваться графическими данными в режиме реального времени. Каждый из данных модулей обладает собственным графическим интерфейсом, который необходимо внедрить в общую систему приложения.

Для внедрения нового графического интерфейса в приложение требуется разработать ряд функций, которые будут связывать его с хранилищем данных, и ряд функций, которые связывают хранилище и конечные модули приложения. Количество функций, связывающих хранилище данных и модули, не всегда линейно зависит от количества данных, полученных из графического интерфейса, поэтому рассмотрим конкретные случаи внедряемых модулей.

Схема графического интерфейса модулей обычного и контролируемого аккаунтов представлена на рисунке 4.3.

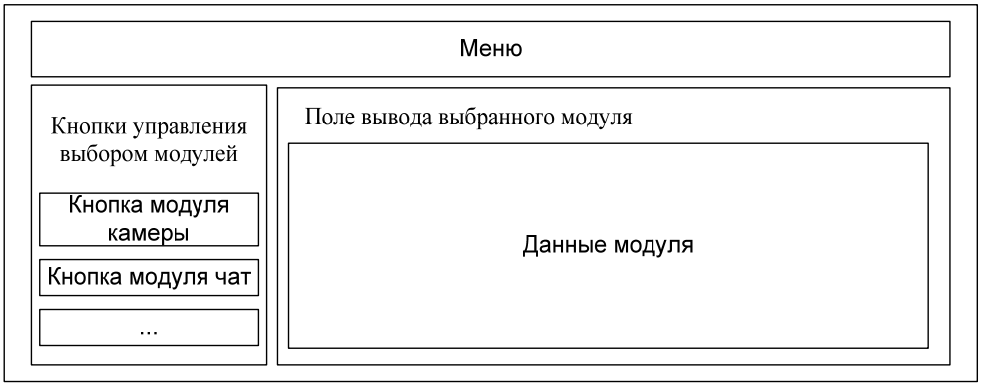


Рис. 4.3. Схема графического интерфейса аккаунта

Данный графический интерфейс полностью идентичен для обоих аккаунтов, за исключением ситуаций, когда контролируемый аккаунт имеет ограничения в настройках и не содержит некоторых элементов интерфейса. При добавлении модуля контролируемого аккаунта в приложение, в обычном аккаунте появляется новый графический элемент, позволяющий управлять настройками контролируемого аккаунта. Новый элемент содержит в себе восемь различных настроек, представляющих поля выбора и два поля ввода данных. Все перечисленные поля тесно взаимодействуют с логикой приложения, формируя данные для сервера. Взаимодействие с каждым графическим элементом настроек и полями ввода данных описывается

функцией, позволяющей передавать данные в хранилища, находящиеся в клиентской части приложения. В клиентской части приложения существует несколько типов хранилищ, которые отвечают за различные модули и типы данных. При изменении данных в хранилищах все модули, которые взаимодействуют с ними, получают оповещения, что данные изменились. Модуль, получив оповещение об изменении данных, запустит алгоритмы, необходимые для их обработки. В общей сумме имеется 16 новых функций, 8 из которых взаимодействуют с хранилищем, и 8 функций, которые служат для оповещения модулей о дальнейшей обработке. Такой подход оправдан, так как перед отправлением на сервер данные проходят предварительную обработку в клиентской части приложения. Обработка подразумевает различные проверки на соответствие определенному набору и количеству символов и взаимодействие связанных друг с другом пунктов. Контролируемый аккаунт в клиентской части приложения полностью заимствует элементы графического интерфейса обычных аккаунтов, поэтому сложность приложения не возрастает.

Модуль интерактивной доски в отличие от контролируемого аккаунта содержит абсолютно новые элементы графического интерфейса, которые представлены на рисунке 4.4.

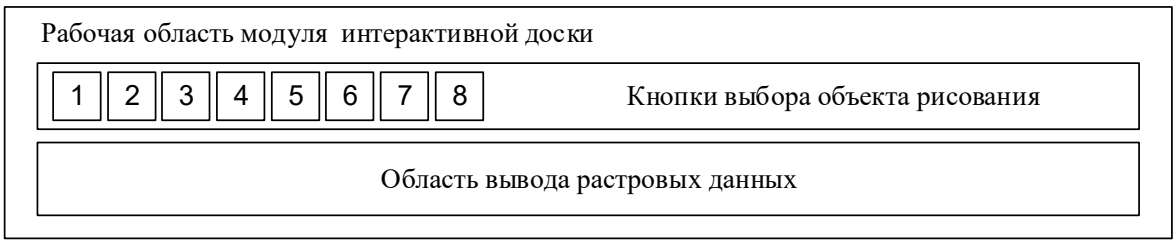


Рис. 4.4. Графический интерфейс модуля интерактивной доски

Значение кнопок в верхней части схемы: 1 — выбор цвета; 2 — элемент рисования кисть; 3 — стирающий элемент; 4 — элемент заливка; 5 — выбор размера кисти или стирающего элемента; 6 — отменить действие; 7 — повторить действие; 8 — очистить область рисования. Интерактивная доска включает в себя 9 графических элементов, одним из которых является область

для рисования изображенная на схеме как область вывода растровых данных. Остальные 8 элементов предназначены для управления областью рисования.

Каждый элемент связан с хранилищем данных через специальные функции. При изменении состояния одного из элементов хранилище сохраняет его текущее состояние и оповещает об изменении элемента другие модули приложения. В данном случае каждый графический элемент имеет всего одну функцию взаимодействия с хранилищем — итого 9 функций. В свою очередь, хранилище для Whiteboard имеет 10 функций, реагирующих на изменение графического интерфейса данного модуля. Девять функций влияют на процесс изменения области рисования, а последняя функция осуществляет подготовку и отправление новых данных с модуля интерактивной доски по протоколу WebRTC другим пользователям. Функции изменения области рисования являются простыми и лишь изменяют состояние объекта рисования.

Несмотря на различные задачи, которые должны выполнять модули и каналы связи, передающие необходимые данные [94 - 96], сложность внедрения модулей в данное пиринговое приложение видеоконференцсвязи примерно одинакова. Это достигается за счет проектирования такой архитектуры приложения, которая позволяет добиться однозначности при интеграции модулей, использовать имеющиеся алгоритмы, незначительно модернизировать их или разрабатывать новые с линейной сложностью. Благодаря подобной архитектуре упрощается поддержка и развитие приложения. Все алгоритмы, необходимые для внедрения модулей в приложение, имеют линейную сложность, что обеспечивает высокую скорость обмена данными между модулями и самим приложением. Наконец разделение самого приложения на отдельные модули предоставляет систему модернизации, в которой каждый элемент является целостным, независимым напрямую от остальных и легко поддается изменению, не затрагивая остальных частей приложения.

Для внедрения модуля контролируемого аккаунта необходимо разработать шестнадцать новых функций, модернизировать два и привязать к четырем уже существующим алгоритмам, также необходимо разработать новый графический

интерфейс, состоящий из восьми элементов. Для внедрения модуля whiteboard в общую структуру приложения потребуется разработать девятнадцать новых функций, два новых алгоритма и один новый графический интерфейс, включающий в себя девять графических элементов. Внедрение модуля контролируемого аккаунта сопровождается разработкой шестнадцати новых функций, модернизацией двух и привязкой к четырем уже существующим алгоритмам; также необходимо разработать новый графический интерфейс, состоящий из восьми элементов.

Все вышеприведенные данные обеспечивают полную интеграцию каждого из модулей в приложение видеоконференцсвязи [97]. Конечная сложность внедрения модулей является достаточно низкой и обеспечивает простоту и надежность интеграции. При такой невысокой сложности можно утверждать, что алгоритмы и функции, связывающие ядро приложения и модулей, потребляют достаточно мало ресурсов оперативной памяти и процессорного времени [98]. Все алгоритмы являются линейными и частично или полностью заимствованными, таким образом, их работа не существенно влияет на производительность приложения. При проведении нагрузочного тестирования в клиентской части приложения были полученные следующие результаты по работе модулей в составе приложения, представленные в таблице 4.2.

Таблица 4.2. Ресурсы, потребляемые дополнительными модулями приложения видеоконференцсвязи

Режим работы	Потребляемые ресурсы			
	Модуль интерактивной доски		Модуль контролируемого аккаунта	
	ОП	ЦП	ОП	ЦП
Запуск приложения	15-20 кб	1-2%	2-10 кб	1%
Одно пиринговое соединение	3000-5000 кб	2-3%	2-10 кб	1%
Три пиринговых соединения	10000-50000 кб	5-10%	2-10 кб	1%

Из таблицы 4.2 видно, что при различных режимах работы приложения ресурсы центрального процессора (ЦП) и оперативной памяти (ОП), потребляемые модулями, незначительны для современных стационарных и мобильных устройств. Таким образом, можно судить о высокой эффективности и быстродействии данных модулей в составе работающего приложения. Модуль контролируемого аккаунта не изменяет свои средние показатели потребления ресурсов на всех этапах работы. Это связанно с тем, что данный модуль формирует заранее определенное количество объектов на этапе запуска приложения и в ходе его работы занимается только изменением данных объектов или передачей их в другие части приложения. Модуль интерактивной доски при запуске приложения потребляет небольшое количество ресурсов, но при присоединении новых участников видеоконференцсвязи количество потребляемых ресурсов возрастает. Возрастание потребления ресурсов связанно с тем, что каждый из пользователей может добавлять новые данные в модуль интерактивной доски. На создание каждого объекта и перерисовку области вывода растровых данных выделяется процессорное время, для хранения данных об объекте и области вывода растровых данных выделяется оперативная память. Чем больше объектов формируется и хранится в модуле интерактивной доски, тем больше нагрузка на приложение. Несмотря на данную особенность работы модуля, он все равно потребляет небольшое количество ресурсов, что в целом не сказывается на производительности приложения. Далее будет рассмотрен анализ производительности самого приложения на различных этапах его работы, чтобы оценить влияние нагрузок на его отклик и работоспособность.

4.3. Анализ потребляемых ресурсов устройствами в ходе видеоконференцсвязи

Для анализа потребляемых устройствами ресурсов в ходе видеоконференцсвязи необходимо провести нагрузочное тестирование приложения на различных этапах работы. Критериями для анализа будет объем

потребляемой оперативной памяти и загруженность центрального процессора [99 - 101]. Так как потоки данных для обеспечения видеоконференцсвязи создаются только в клиентской части приложения, то для тестирования были выбраны именно клиентские устройства пользователей [100].

Автоматизация данного тестирования достаточно сложная и трудоемкая задача, которая требует разработки определенной системы, которая в конечном итоге не сможет обеспечить полноценную оценку качества видео- и аудиоконтента, так как оценка пользователя системы является субъективной и более достоверной в плане восприятия качества мультимедийного контента [102 - 104]. Для тестирования видеоконференцсвязи были выбраны четыре пользователя и четыре различных по характеристикам устройства. На двух из четырех устройств была установлена операционная система Windows 7-ой и 10-ой версий, остальные устройства работали под управлением Macintosh OS El Capitan и Linux Ubuntu 14.04. Данный выбор был произведен с целью охвата основных широко используемых операционных систем и тестирования приложения на работоспособность, нагрузку и быстродействие [105, 106]. В таблице 4.3 представлены основные характеристики и названия устройств для тестирования.

Таблица 4.3. Основные характеристики тестируемых устройств

	MacBook Pro	Le№ vo t430	Стационарный	
Экран (размер в дюймах, разрешение)	13,3, 2560×1600	14, 1600 x 900	23, 1920x1080	15.6, 1600 x 900
Процессор	Intel Core i5-3210M (2 ядра, 2,5 ГГц)	Intel Core i7 (4 ядра, 2,9 ГГц)	Intel Core i5 6600 (4 ядра, 3.3 ГГц)	Intel i5-4210M (2 ядра, 2.60гц)
Графическая система	Intel HD Graphics 4000	Intel HD Graphics 4000 и Nvidia NVS 5200M 1 Гб	NVIDIA GeForce GTX 970 4 Гб	Geforce 840M
ОЗУ	8 Гб DDR3L	8 Гб DDR3	16 Гб DDR4	4 Гб DDR3
SSD / HDD	128 Гб SSD	20 Гб SSD, 1 Тб HDD	2 Тб HDD	500 HDD
Интернет, Bluetooth	Wi-Fi 802.11b/g/n, Bluetooth 4.0	Wi-Fi 802.11b/g/n, Bluetooth 4.0	Нет	Wi-Fi 802.11b/g/n, Bluetooth 4.0
Веб-камера	есть (720p)	есть (720p)	есть (720p)	есть (720p)
Операционная система	Macintosh OS El Capitan	Windows 7 Professional	Linux Ubuntu 14.04	Windows 10 Professional

В ходе нагрузочного тестирования с камер и микрофонов каждого из устройств был произведен захват аудио- и видеопотоков и соединение по протоколу WebRTC. Для организации работы в клиентской части приложения были использованы браузеры Firefox и Chrome. Данные браузеры для кодирования аудио данных используют кодек Opus [107], который сжимает данные с потерями и используется в приложениях реального времени в Интернете. Основное преимущество данного кодека — низкая задержка кодирования (от 2,5 до 60 мс, настраиваемо), более сильное сжатие аудиоданных, поддержка многоканального звука (до 255 каналов). Каждый из используемых браузеров имеет свой собственный кодек для передачи видеоданных, именно поэтому невозможно организовать пиринговую связь для более чем двух клиентов с передачей мультимедийных потоков через промежуточное звено цепочки пиринговой сети для разных браузеров.

Для браузера Chrome используется кодек VP8 [108], созданный компанией On2 Tech№ logies. Данный кодек имеет следующие преимущества: 1) повышенная устойчивость к потере пакетов; 2) опорные кадры, хранящиеся в отдельном буфере и допускающие ссылку на себя спустя значительное время после их декодирования; 3) фильтрация артефактов от DCT-кодирования; 4) возможность кодирования со множеством слабо зависимых подпотоков, позволяющее масштабировать декодирование на многоядерных архитектурах; 5) декодирование адаптировано как к SIMD-расширениям, так и к процессорам без них, со слабой (медленной) поддержкой байтовых операций, упрощенное по сравнению с предыдущими кодеками On2 энтропийное кодирование и субпиксельное предсказание для ускорения декодирования, кодек имеет профили, оптимизированные для проведения видео-конференций в реальном времени [109].

Для браузера Firefox используется кодек H.264 [110]. Данный кодек предназначен для достижения высокой степени сжатия видеопотока при сохранении высокого качества. H.264 обладает следующими возможностями: использование сжатых ранее кадров в качестве опорных; независимость порядка воспроизведения изображений и порядка опорных изображений; независимость методов обработки изображений и возможности их использования для предсказания движения; компенсация движения с

переменным размером блока; векторы движения, выводящие за границы изображения; шеститочечная фильтрация компонента яркости для полупиксельного предсказания с целью уменьшения зубчатости краев; точность до четверти пиксела (Qpel) при компенсации движения; взвешенное предсказание; пространственное предсказание от краев соседних блоков для I-кадров; сжатие макроблоков без потерь; гибкие функции чересстрочного сжатия; новые функции преобразования; квантование; внутренний фильтр деблокинга в цикле кодирования [109]. Размеры захватываемого на клиенте изображения составляют 720 на 480 пикселей, что позволило передавать качественный для человеческого восприятия видеопоток. Средний битрейт аудиопотока для WebRTC составляет примерно 40-60 килобит в секунду.

Тестирование проводилось в сравнении с наиболее популярной веб-версией многоканальной системой видеоконференцсвязи Skype, которая не поддерживается ни в одном мобильном браузере, в отличие от разработанного приложения, которое также имеет поддержку в браузере Chrome на Android платформах. Для работы Skype системы в браузерах на декстопных платформах необходимо было установить дополнительный плагин. Разработанная система работает без дополнительных средств и обеспечивает такой же функционал по передаче аудио- и видеоданных. Данные нагрузочного тестирования были получены на основе вышеприведенных параметров и усреднены для упрощения представления их в таблице 4.4.

Таблица 4.4. Результаты нагрузочного тестирования клиентской части разработанной системы и приложения Skype

Режим работы клиентской части приложения	Объем потребляемой оперативной памяти клиентской частью приложения		Загруженность процессора клиентской части приложения	
	Разработанная система	“Skype”	Разработанная система	“Skype”
Режим проверки логина и пароля	~20 Мб	~25 Мб	~1%	~1%
Режим ожидания соединения	~22 Мб	~30 Мб	~2%	~2%
1 входящий мультимедийный поток	~100 Мб	~110 Мб	~10%	~12%
2 входящих мультимедийных потока	~120 Мб	~120 Мб	~30%	~29%
3 входящих мультимедийных потока	~140 Мб	~140 Мб	~68%	~69%

Усредненные результаты измерений, потребляемых ресурсов центральным процессором (ЦП) и оперативной памятью (ОП) в разработанном приложении и системе Skype на различных этапах работы приложений при одинаковых входных и выходных параметрах – одинаковы. Необходимо отметить, что приложения Skype на декстопных платформах работало только в браузере Firefox, разработанное приложение было работоспособно в трех браузерах: Chrome, Opera и Firefox. Таким образом при большей функциональности, отсутствии дополнительной установки плагинов и поддержки большего количества браузеров в том числе и на мобильной платформе Android, разработанное приложение видеоконференцсвязи обеспечивает такое же потребление ресурсов центрального процессора и оперативной памяти, как и система Skype [111].

4.4. Анализ потребляемых серверной частью ресурсов в ходе работы приложения

В связи с тем, что серверное оборудование обычно имеет значительно большую производительность по сравнению с клиентскими устройствами, для того, чтобы убедиться, что приложение работает корректно под нагрузкой и не потребляет большое количество ресурсов, был использован обычный стационарный компьютер под управлением Linux Ubuntu 14.04 со следующими характеристиками: процессор Intel Core i5 6600 (4 ядра, 3.3 ГГц), видеокарта NVIDIA GeForce GTX 970 4 Гб, оперативная память 16 Гб DDR4, жесткий диск емкостью 2 Тб. Для тестирования разработанной серверной части был создан специальный набор тестов, который позволил осуществлять процессы соединения с сервером по различным протоколам, отправлять данные на сервер для их передачи другим пользователям и сохранения в базе данных, использовать возможности API облачного взаимодействия и осуществлять поиск и фильтрацию данных на сервере и в базе данных созданных тестовых клиентов, каждые 10 миллисекунд. Тестирование проходило в три стадии: запуск сервера, работа с четырьмя клиентами, работа со 100 клиентами.

Результаты максимальных показателей потребления ресурсов на всех трех этапах отображены в таблице 4.5.

Таблица 4.5. Максимальные показатели потребления ресурсов устройства при нагрузочном тестировании серверной части приложения видеоконференцсвязи

Устройства	Данные нагрузки		
	Запуск сервера	Минимальная нагрузка	Пиковая нагрузка
Центральный процессор (ЦП)	5%	12%	47%
Оперативная память (ОЗУ)	10 MB	307 MB	1202 MB

Первый этап тестирования проходил на протяжении 10 минут. В ходе него были выявлены незначительные отклонения от результатов, представленных в таблице 4.4, которые подтверждают стабильную работу сервера без нагрузки и отсутствие утечек памяти. Второй и третий этап проходили на протяжении 30 минут, каждые 20 секунд производилось снятие показаний о нагрузке на сервер, которые отображены на графиках, представленных на рисунке 4.5 и 4.6.

На графиках, представленных на рисунке 4.5, видны изменения, происходящие в момент тестирования приложения на нагрузку при пяти подключенных клиентах. Изменения происходят в определенном диапазоне значений, что означает стабильную работу сервера и отсутствие утечек памяти. Анализируя данные, представленные на графиках, можно сделать следующий вывод: на протяжении всего тестирования пик нагрузки приходится на центральный процессор при пяти клиентах в момент запуска приложения. Данный пик обусловлен одновременным подключением всех тестовых клиентов и выполнением операций для их обслуживания.

Дальнейшая нагрузка на процессор не достигает первого пикового значения и имеет равномерное изменение значений в диапазоне от 5% до 10% на оставшемся промежутке времени, что позволяет судить о стабильности работы серверной части приложения и своевременной обработке данных. Оперативная память приложения равномерно изменяется в диапазоне от 156 Мб до 308 Мб, что позволяет судить об отсутствии утечек памяти и стабильной

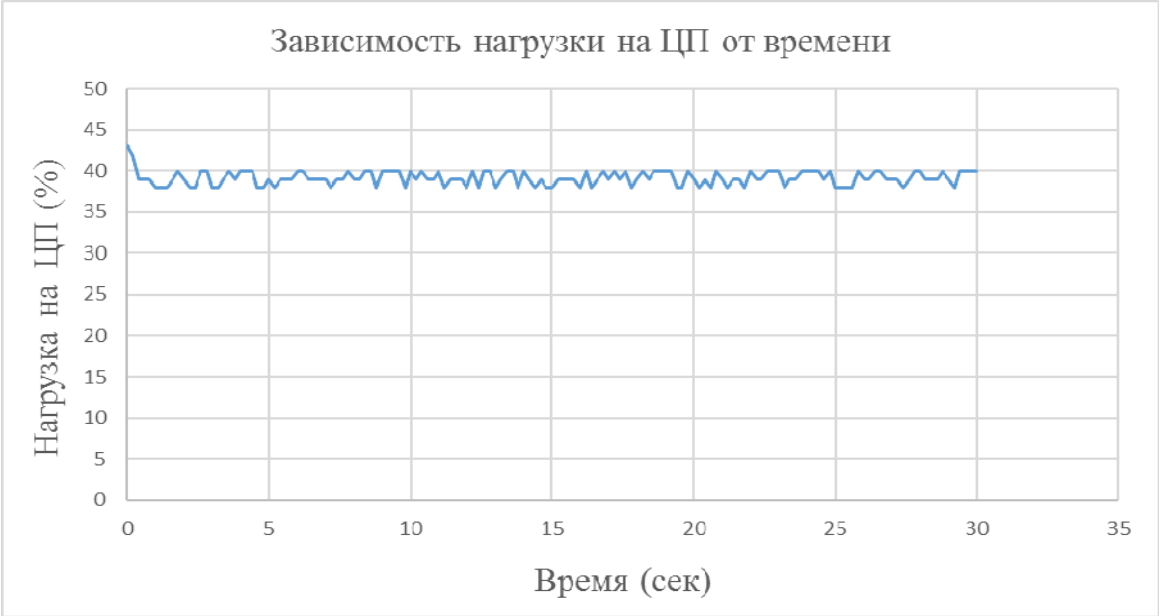
работе системы Garbage Collector встроенной в Node.js, который при достижении пиковых отметок начинает работу по очистке памяти.

а)

б)

Рис. 4.5. Зависимость потребляемых ресурсов центрального процессора и оперативной памяти от времени при пяти клиентах

На рисунке 4.6 представлены графики тестирования нагрузки при 100 клиентах. Как видно из графика 4.6а, наибольшая пиковая нагрузка достигается также при запуске серверной части приложения и одновременного соединения с ней множества клиентов.



а)

б)

Рис. 4.6. Зависимость потребляемых ресурсов центрального процессора и оперативной памяти от времени при 100 клиентах

Далее происходит еще одна менее значительная пиковая нагрузка, которая обусловлена большим количеством запросов к серверу на сохранение и получения информации из базы данных. Данная пиковая нагрузка имеет значение в 45%, что не превышает пиковую нагрузку на старте. Потребление оперативной памяти на всем промежутке времени, представленное на рисунке 4.6 б, колеблется в диапазоне от 803 Мб до 1202 Мб, что также позволяет судить об отсутствии утечек памяти и стабильной работе серверной части приложения. В целом работа сервера при отсутствии клиентов и во время

различных нагрузок стабильна, потребляет удовлетворительное количество ресурсов и позволяет одновременно обслуживать большое количество клиентов в режиме реального времени. Отсутствие утечек памяти позволяет предотвратить некорректное завершение работы серверной части и случаи использования всей оперативной памяти устройства.

4.5. Выводы

1. При проведении экспериментальной проверки разработанного приложения видеоконференцсвязи были разработаны алгоритмы функционального тестирования частей приложения, проведена оценка новых графических интерфейсов для модуля контролируемого аккаунта и модуля интерактивной доски, а также выполнен анализ потребляемых ресурсов клиентскими устройствами и серверной частью в ходе видеоконференцсвязи.

2. При функциональном тестировании разработанного приложения отдельно верифицировались несколько функциональных модулей по трем методикам, включающим: 1) тестирование графического интерфейса в клиентской части приложения, 2) тестирование логики работы клиентской части; 3) тестирование логики серверной части приложения. Применение разработанных методик позволило выявить и исправить программные ошибки и перейти к тестированию поведения системы и ее отклика в момент нагрузки при многопользовательском сеансе связи.

3. В разработанной системе существуют два дополнительных модуля: модуль контролируемого аккаунта и модуль интерактивной доски, позволяющие создавать аккаунты, управлять которыми может один пользователь-администратор и обмениваться графическими данными в режиме реального времени.

4. Для анализа ресурсов, потребляемых клиентской частью, запускаемой на гетерогенных мобильных устройствах, в ходе видеоконференцсвязи было проведено нагрузочное тестирование приложения на различных этапах работы с проверкой объема потребляемой оперативной памяти и загруженности центрального процессора. На двух из четырех устройств была установлена

операционная система Windows 7-ой и 10-ой версий, остальные устройства работали под управлением Macintosh OS El Capitan и Linux Ubuntu 14.04. В ходе нагрузочного тестирования с камер и микрофонов каждого из устройств был произведен захват аудио- и видеопотоков и соединение по протоколу WebRTC. Тестирование проводилось в сравнении с наиболее популярной многоканальной системой видеоконференцсвязи Skype и показало, что разработанная система потребляет меньшее количество оперативной памяти, чем Skype, на всех этапах работы приложения.

5. Для тестирования потребляемых ресурсов разработанной серверной части системы видеоконференцсвязи был создан специальный набор тестов, который позволил осуществлять процессы соединения с сервером по различным протоколам, отправлять данные на сервер для их передачи другим пользователям и хранения в базе данных, использовать возможности API облачного взаимодействия и осуществлять поиск и фильтрацию данных на сервере и в базе данных созданных тестовых клиентов каждые 10 миллисекунд. Тестирование проходило в 3 стадии: запуск сервера, работа с четырьмя клиентами, работа со 100 клиентами. Основной пик нагрузки на центральный процессор при пяти клиентах происходит в момент запуска приложения. Он одновременным подключением всех тестовых клиентов и выполнением операций для их обслуживания. Потребление оперативной памяти на всем промежутке времени тестирования при обслуживании 100 клиентов колеблется в диапазоне от 803 Мб до 1202 Мб. В целом тестирование показало стабильную работу сервера и отсутствие утечек памяти.

Заключение

Совокупность предложенных архитектур, алгоритмов и программных средств автоматической обработки и передачи потоков данных, а также их практическая реализация представляют собой решение актуальной научно-технической задачи поддержки многоканальной коммуникации удаленных абонентов на основе веб-приложений видеоконференцсвязи с пиринговой архитектурой, исполняемых на гетерогенных клиентских устройствах [112-114]. Их внедрение существенно влияет на современное развитие инфокоммуникационных технологий [115, 116]. При решении данной задачи были получены следующие результаты.

Совокупность предложенных архитектур, алгоритмов и программных средств автоматической обработки потоков данных, а также их практическая реализация представляют собой решение актуальной научно-технической задачи поддержки многоканальной коммуникации удаленных абонентов на основе веб-приложений видеоконференцсвязи с пиринговой архитектурой, исполняемых на гетерогенных клиентских устройствах. Их внедрение вносит значительный вклад в развитие инфокоммуникационных технологий. При решении данной задачи были получены следующие результаты.

1. Архитектуры клиентской и серверной частей в системах видеоконференцсвязи, отличающиеся применением пирингового принципа коммуникации, обеспечивающего сокращение объема передаваемых данных, снижение потребляемых ресурсов сервера и распределенных гетерогенных клиентских устройств, а также возможность построения речевых и многомодальных интерфейсов для данного типа телекоммуникационных приложений.

2. Алгоритмы взаимодействия клиентской и серверной частей системы видеоконференцсвязи и установления соединений между клиентами по протоколу WebRTC, обеспечивающие распределение сокетов и обработку их

потоков данных на сервере, обработку «сигнальных» данных, поддерживая взаимодействие между группами клиентов.

3. Программные средства клиентской и серверной частей веб-приложения видеоконференцсвязи обеспечивают кроссплатформенность за счет реализации веб-интерфейса с использованием языков HTML, CSS и JavaScript, а также сокращение передаваемых потоков мультимедийных данных по протоколам HTTP и WebSocket между клиентом и сервером, упакованных в формат JSON, за счет пиринговой передачи данных между клиентскими приложениями в процессе сеанса связи.

4. Методика тестирования пиринговых систем видеоконференцсвязи, включающая алгоритмы функционального тестирования и набор тестов, оценивающих потребляемые ресурсы, позволившая в ходе тестирования разработанного программного обеспечения улучшить эргономику пользовательского интерфейса и снизить объем передаваемых данных, подтвердить стабильную работу сервера и корректное использование памяти, а также провести сравнение с наиболее популярной системой видеоконференцсвязи «Skype», которое показало, что разработанная система потребляет меньшее количество оперативной памяти на всех этапах работы.

Дальнейшим перспективным направлением исследования является интеграция системы видеоконференцсвязи и интеллектуальных методов распознавания звуковых и графических образов, биометрической идентификации, цифровых программируемых технологий инфокоммуникационных систем [112-114], позволяющих существенно снизить объем передаваемых данных по пиринговым связям для распределения нагрузки на вычислительные и сетевые встроенные ресурсы мобильных гетерогенных клиентских устройств, использующихся при коммуникации в сетях с многоуровневой архитектурой [115-116].

Полученные результаты соответствуют п. 3 «Модели, методы, алгоритмы, языки и программные инструменты для организации взаимодействия программ и программных систем» п. 7 «Человеко-машинные интерфейсы, модели,

методы, алгоритмы и программные средства машинной графики, визуализации, обработки изображений, систем виртуальной реальности, мультимедийного общения» и п. 8 «Модели и методы создания программ и программных средств для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования» паспорта специальности 05.13.11 – «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей».

Литература

1. Ronzhin, A.L., Saveliev, A.I., Budkov, V.Yu. Context-Aware Mobile Applications for Communication in Intelligent Environment / A.L. Ronzhin, A.I. Saveliev, V.Yu. Budkov // Internet of Things, Smart Spaces, and Next Generation Networking. — Springer Berlin Heidelberg, 2012. — С. 307–315.
2. So, S. Mobile instant messaging support for teaching and learning in higher education / S. So // Internet and Higher Education. — 2016. — Т. 31. — С.32–42.
3. Chan, T. et al. Studying with the cloud: the use of online Web-based resources to augment a traditional study group format / T. Chan, S. Sennik, A. Zaki, B. Trotter // CJEM. — 2014. — Т. 16. — С.34–37.
4. Petrovčič, A et al. Landline and mobile phone communication in social companionship networks of older adults: An empirical investigation in Slovenia / A. Petrovčič, V. Vehovar, V. Dolničar // Tech№ logy in Society. — 2016. — Т. 45. — С.91–102.
5. Van der Ploeg, E. S. Internet video chat (Skype) family conversations as a treatment of agitation in nursing home residents with dementia / E. S. Van der Ploeg, B. Eppingstall, D. W. O'Con№ r // International Psychogeriatrics. — 2016. — Т. 28:4. — С. 697–700.
6. Dooly, M., Sadler, R. Filling in the gaps: Linking theory and practice through telecollaboration in teacher education / M. Dooly, R. Sadler // ReCALL. — 2013. — Т. 25(1). — С. 4–29.
7. Gerpott, T. J., Meinert, P. The impact of mobile Internet usage on mobile voice calling behavior: A two-level analysis of residential mobile communications customers in Germany / T. J. Gerpott, P. Meinert // Telecommunications Policy. — 2016. — Т. 40. — № 1. — С. 62–76.

8. Anshari, M., Alas, Y. Smartphones habits, necessities, and big data challenges / M. Anshari, Y. Alas // Journal of High Tech№ logy Management Research. — 2015. — Т. 26. — С. 177–185.
9. Gerpott, T. J. Personal characteristics and mobile Internet use intensity of consumers with computer-centric communication devices: An exploratory empirical study of iPad and laptop users in Germany / T. J. Gerpott, S. Thomas, M. Weichert // Telematics and Informatics. — 2013. — Т. 30. — С. 87–99.
10. Erol, B., Li, Y. An overview of tech№ logies for e-meeting and e-lecture / B. Erol, Y Li. // Proc. IEEE International Conference on Multimedia and Expo (ICME»2005). — IEEE, 2005. — С. 6–12.
11. Ронжин, Ан. Л., Будков, В. Ю., Ронжин, Ал. Л. Технологии формирования аудиовизуального интерфейса системы телеконференций / Ан. Л. Ронжин, В. Ю. Будков, Ал. Л. Ронжин // Автоматизация и современные технологии. — № 5. — 2011. — С. 20–26.
12. Герасимова, В. В., Кирпичникова, М. Ю. Перспективы использования интернет-телевидения / В. В. Герасимова, М. Ю. Кирпичникова // Молодежный научный вестник. — 2016. — № 7(7). — С. 46–51.
13. Айдынбай, Т. Ж., Шуйтенов, Г. Ж. Технологии передачи данных в системах видеоконференцсвязи / Т. Ж. Айдынбай, Г. Ж. Шуйтенов // Наука, техника и образование. — 2015. — № 4(10). — С. 77–83.
14. Месяцев С. Мобильная конференцсвязь будущего / С. Месяцев // Мобильные системы. — 2007. — № 12. — С. 48–53.
15. Федотов, Е. А. Бондаренко, Т. В., Федотова, В. Н., Поляничка, М. И. Исследование протоколов обмена сообщениями в режиме реального времени / Е. А. Федотов, Т. В. Бондаренко, В. Н. Федотова, М. И. Поляничка // Вестник магистратуры. — 2016. — № 5-2(56). — С. 64–66.
16. Wu, J. et al. TRADER: A reliable transmission scheme to video conferencing applications over the internet / J. Wu, Y. Shang, C. Yuen,

B. Cheng, J. Chen. // Journal of Network and Computer Applications. — 2014. — Т. 44. — С. 161–171.

17. Бершадский А. М. Курилов Л. С., Бождай А. С., Гудков П. А., Гудков, А. А. Экспериментальное исследование процессов передачи голосовых данных в беспроводных сенсорных сетях / А. М. Бершадский, Л. С. Курилов, А. С. Бождай, П. А. Гудков, А. А. Гудков // Прикаспийский журнал: Управление и высокие технологии. — 2012. — № 3. — С. 24–29.
18. Терехов, А. Н. Метод интегральной оценки качества телефонного общения при модернизации сетей и/или введении перспективных услуг связи/ А. Н. Терехов // Т-СООМ: Телекоммуникации и транспорт. — 2016. — Т. 10. — № 4. — С. 67–72.
19. Воробьев, Л. В., Ткачев, Д. Ф. Предложение по реализации механизмов обеспечения качества передачи речи в инфокоммуникационных сетях специального назначения / Л. В. Воробьев, Д. Ф. Ткачев // Т-СООМ: Телекоммуникации и транспорт. — 2015. — № 1. — С. 28–31.
20. Семенов, С. С. Применение технологии распознавания образов как инструмент решения задач технической разведки техники связи и автоматизированных систем управления / С. С. Семенов, А. В. Педан, А. В. Смолеха // Системы управления, связи и безопасности. — 2015. — № 1. — С. 26–36.
21. Виноградено, А. М., Будко, Н. П., Юров, А. С. Моделирование процессов мультиплексирования в многоканальной измерительной системе методами теории массового обслуживания / А. М. Виноградено, Н. П. Будко, А. С. Юров // Успехи современной радиоэлектроники. — 2014. — № 4. — С. 67–69.
22. Легков, К. Е., Захарченко, Р. И. Система поддержки принятия решения автоматизированной системы управления связи на основе организации информационного хранилища с аналитической обработкой данных /

К. Е. Легков, Р. И. Захарченко // Т-СООМ: Телекоммуникации и транспорт. — 2013. — Т. 7. — № 6. — С. 28–34.

23. Матвеев, Ю. Н., Шулипа, А. К. Анализ возможности применения методов машинного обучения на основе многообразий в задачах распознавания дикторов / Ю. Н. Матвеев, А. К. Шулипа // Известия высших учебных заведений. Приборостроение. — 2014. — Т. 57. — № 2. —С. 70–76.
24. Симончик, К. К. Доступ к интернет-банкингу на основе бимодальной биометрии / К. К. Симончик, Д. О. Белевитин, Ю. Н. Матвеев, Д. В. Дырмовский // Мир измерений. — 2014. — № 3. — С. 6–10.
25. Матвеев, Ю. Н. Исследование информативности признаков речи для систем автоматической идентификации дикторов / Ю. Н. Матвеев // Известия высших учебных заведений. Приборостроение. — 2013. — Т. 56. — № 2. — С. 47–51.
26. Матвеев Ю.Н. Технологии биометрической идентификации личности по голосу и другим модальностям / Ю. Н. Матвеев // Инженерный журнал: наука и инновации. — 2012. — № 3(3). — С. 5.
27. Матвеев Ю. Н., Шулипа А. К. Гистограммная нормализация речевых признаков в задаче верификации дикторов / Ю. Н. Матвеев, А. К. Шулипа // Научно-технический вестник информационных технологий, механики и оптики. 2012. № 6(82). С. 85–88.
28. Алейник, С. В., Матвеев Ю. Н., Раев А. Н. Метод оценки уровня клиппирования речевого сигнала / С. В. Алейник, Ю. Н. Матвеев, А. Н. Раев // Научно-технический вестник информационных технологий, механики и оптики. — 2012. — № 3(79). — С. 79–83.
29. Taillefer, Lidia, Mu№ z-Luna, Rosa. Developing Oral Skills Through Skype: A Language Project Analysis / Lidia Taillefer and Rosa Mu№ z-Luna // Procedia - Social and Behavioral Sciences. — 2014. — Т. 141. — С. 260–264.

30. Ronzhin, A., Budkov, V., Karpov, A. Multichannel System of Audio-Visual Support of Remote Mobile Participant at E-Meeting / A. Ronzhin, V. Budkov, A. Karpov // Smart Spaces and Next Generation Wired/Wireless Networking. — Springer Berlin Heidelberg, 2010. — С. 62–71.
31. Городилов, А.В., Кононова, А.И., Шаньгин, В.Ф. Особенности передачи данных в децентрализованных пиринговых сетях / А. В. Городилов, А. И. Кононова, В. Ф. Шаньгин // Известия высших учебных заведений. Электроника. — 2012. — № 6(98). — С. 95–97.
32. H.R. Oh et al. An effective mesh-pull-based P2P video streaming system using Fountain codes with variable symbol sizes / H. R. Oh, D. O.Wu, H. Song // Computer Networks. — 2011. — Т. 55. — С. 2746–2759.
33. Шурховецкий, П. П. Моделирование децентрализованной структурированной P2P сети на основе mesh-топологии для задачи обнаружения ресурсов в глобально распределенных системах обработки данных / П. П. Шурховецкий // Вестник ПГТУ. Серия: Радиотехнические и Инфокоммуникационные системы. — 2011. — № 2. — С. 36–45.
34. Keralapura, R. A № vel self-learning architecture for p2p traffic classification in high speed networks / R. Keralapura, A. Nucci, C. N Chuah // Computer Networks. — 2010. — Т. 54. — С. 1055–1068.
35. Губарев, В. В., Обейдат, А. А. Алгоритм взаимного исключения одновременного доступа пользователей к общим ресурсам в пиринговых системах / В. В.Губарев, А. А. Обейдат // Вестник НГТУ. — 2009. — № 2. — С. 75–90.
36. Wang, Xu-hui. Research on P2P-SIP based VoIP system enhanced by UPnP tech№ logy / Xu-hui Wang // The Journal of China Universities of Posts and Telecommunications. — 2010. — Т. 17. — С. 36–40.
37. Hoßfeld, Tobias, Binzenhöfer, Andreas. Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE measurements / Tobias Hoßfeld, Andreas Binzenhöfer // Computer Networks. — 2008. — Т. 52. — С. 650–666.

38. Alvarez, C., Alarcon, R. Nussbaum, M. Implementing collaborative learning activities in the classroom supported by one-to-one mobile computing: A design-based process / C. Alvarez, R. Alarcon, M. Nussbaum // The Journal of Systems and Software. — 2011. — Т. 84. — С. 1961–1976.
39. Kleij, R., Jong, A., Brake, G., Greef, T. Network-aware support for mobile distributed teams / R. Kleij, A. Jong, G. Brake, T. Greef. // Computers in Human Behavior. — 2009. — Т. 25. — С. 940–948.
40. Skype [Электронный ресурс]. — Режим доступа: // http://skype.com. — Заглавие с экрана.
41. Madiraju, P. A methodology for engineering collaborative and ad-hoc mobile applications using SyD middleware / P. Madiraju, S. Malladi, J. Balasooriya, A. Hariharan, S. Prasad, A. Bourgeois // Journal of Network and Computer Applications. — 2010— Т. 33. — С. — 542–555.
42. Covili, J. A communication infrastructure to ease the development of mobile collaborative applications / J. Covili, S. Ochoa, J. Pi№ , R. Messeguer, E. Medina, D. Royo // Journal of Network and Computer Applications. — 2011. — Т. 34. — С. 1883–1893.
43. Голошубина, О. К. Модель речевого жанра «Разговор в мессенджере» / О.К. Голошубина // Омский научный вестник. — 2014. — № 5(132). — С. 101–103.
44. Голошубина, О. К. Разговор в мессенджере как специфический жанр интернет-коммуникации / О. К. Голошубина // Вестник Омского университета. — 2015. — № 1(75). — С. 208–2012.
45. Sultan, A. J. Addiction to mobile text messaging applications is № thing to “lol” about // Abdullah J. Sultan. / The Social Science Journal. — 2014. — Т. 51. — С. 57–69.
46. Ha, Y. W. et al. Use and gratifications of mobile SNSs: Facebook and KakaoTalk in Korea / Y. W. Ha, J. Kim, C. F. Libaque-Saenz, Y. Chang, M. C. Park, // Telematics and Informatics. — 2015. — Т. 32. — № 3. — С. 425–438.

47. Попова А. А. Сравнительный анализ мобильных мессенджеров / А.А. Попова // Информационные системы и технологии в образовании, науке и бизнесе (ИСИТ-2014). — 2014. — С. 242–243.
48. Гатиятуллин, Т. Р. К вопросу выбора безопасного мессенджера / Т. Р. Гатиятуллин // Наука, техника и образование. — 2016. — № 1(19). — С. 79–81.
49. Вахонин, С. Обеспечение информационной безопасности в Skype / С. Вахонин // Windows IT Pro/ Re. — 2013. — № 5. — С. 59.
50. Шумилова, Е. Д., Семенов, А. А. Разработка мобильного приложения для отправки экстренных сообщений о помощи в чрезвычайных ситуациях / Е. Д. Шумилова, А. А. Семенов // Сборник материалов Всероссийской научной студенческой конференции «Инновационное развитие легкой и текстильной промышленности». — 2015. — С. 124–126.
51. Dustdar, S., Gall, H. Architectural concerns in distributed and mobile collaborative systems / S. Dustdar, H. Gall // Journal of Systems Architecture. — 2003. — Т. 49. — С. 457–473.
52. Ruan, Z., Ngai, E., Liu, J. Wireless sensor deployment for collaborative sensing with mobile phones / Ruan, E. Ngai, J. Liu. // Computer Networks. — 2011. — Т. 55. — С. 3224–3245.
53. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W. A measurement study of a large-scale P2P IPTV system / X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross. // IEEE Transactions on Multimedia. — 2007. — Т. 9(8). — С. 1672–1687.
54. Hei, X., Liu, Y., Ross, K. W. Inferring network-wide quality in P2P live streaming systems / X. Hei, J. Liang, Y. Liu, K. W. // IEEE journal on Selected Areas in Communications. — 2007. — Т. 25. — № 9. — С. 1640–1654.
55. Wu, Di, Liang, Chao, Liu, Yong, Ross, Keith W. Redesigning multi-channel P2P live video systems with View-Upload Decoupling / Di Wu,

Chao Liang, Yong Liu, Keith W. Ross // Computer Networks. — 2010. — T. 54. — C. 2007–2018.

56. Civanlar, M. R. et al. Peer-to-peer multipoint videoconferencing on the Internet / M. R. Civanlar, Ö. Özkasap, T. Çelebi // Signal Processing: Image Communication. — 2005. — T. 20. — pp.743–754.
57. Tfin, Ilana, Cohen, Israel. Dominant speaker identification for multipoint videoconferencing / Ilana Tfin, Israel Cohen // Computer Speech and Language. — 2013. — T. 27. — C. 895–910.
58. Ronzhin, A., Budkov, V. Speaker Turn Detection Based on Multimodal Situation Analysis / A. Ronzhin, V. Budkov // Springer International Publishing Switzerland, SPECOM 2013, LNAI 8113. — 2013. — C. 302–309.
59. Ronzhin, A., Budkov, V., Kipyatkova I. PARAD-R: Speech Analysis Software for Meeting Support / A. Ronzhin, V. Budkov, I. Kipyatkova // In Proc. of the 9th International Conference on Information, Communications and Signal Processing ICICS-2013. — 2013. — C. 1–4.
60. Ronzhin, A. L., Saveliev, A. I., Budkov, V. Yu. Context-Aware Mobile Applications for Communication in Intelligent Environment / A. L. Ronzhin, A. I. Saveliev, V. Yu. Budkov // Springer-Verlag Berlin Heidelberg: NEW2AN/ruSMART 2012, LNCS 7469. — 2012. — C. 307–315.
61. Baskaran V. M., Chang, Y. C., Loo, J., Wong, K. Software-based serverless endpoint video combiner architecture for high-definition multiparty video conferencing / Vishnu Monn Baskaran, Yoong Choon Chang, Jonathan Loo, KokSheik Wong // Journal of Network and Computer Applications. — 2013. — T. 36. — C. 336–352.
62. Ramzan, N. et al. Video streaming over P2P networks: Challenges and opportunities / N. Ramzan, H. Park, E. Izquierdo // Signal Processing: Image Communication. — 2012 — T. 27. — C. 401–411.
63. Савельев, А. И. Оптимизация алгоритмов распределения потоков мультимедийных данных между сервером и клиентом в приложениях

видеоконференцсвязи / А. И. Савельев // Труды СПИИРАН. — 2013. — Вып. 31. — С. 61–79.

64. Ronzhin, A. L., Saveliev A. I., Budkov V. Yu. Context-Aware Mobile Applications for Communication in Intelligent Environment / A. L. Ronzhin, A. I. Saveliev, V. Yu. Budkov // Springer-Verlag Berlin Heidelberg: NEW2AN/ruSMART 2012, LNCS 7469. — 2012. — С. 307–315.
65. Павлов М. С. Реализация сервера для установления соединения и непрерывного обмена между парами компьютер-мобильное устройство / М. С. Павлов // Молодежный научно-технический вестник. — 2014. — № 10. — С. 18.
66. Герасимова, А. И. Оптимизация мобильной AD-HOC сети, использующей распределенный механизм Push-To-Talk / А. И. Герасимова // Приборы и системы. Управление, контроль, диагностика. — 2009. — № 10. — С. 23–25.
67. Левин, Ю. Разработка веб-интерфейса для сервера уведомлений / Ю. Левин // International Journal of Open Information Tech№ logies. — 2013. — Т. 1. — № 2. — С. 12–16.
68. Десницкий, В. А., Котенко, И. В., Резник, С. А. Разработка и верификация протокола обмена сообщениями для защиты программ на основе механизма «Удаленного доверия» / В. А. Десницкий, И. В. Котенко, С. А. Резник // Защита информации. Инсайд. — 2008. — № 4(22). — С. 59–63.
69. Виноградено, А. М. Формирование многоканальной телеметрической системы как многокритериальная задача распределения информационных ресурсов / А. М. Виноградено // Успехи современной радиоэлектроники. — 2015. — № 3. — С 179–185.
70. Bulut, H., Fox, G., Pallickara, S., Uyar, A., Wu, W. Integration of NaradaBrokering and Audio/Video Conferencing as a Web Service / H. Bulut, G. Fox, S. Pallickara, A. Uyar, W. Wu // In Proceedings of IASTED

International Conference on Communications, Internet, and Information Tech№ logy. — 2002. — C. 401–406.

71. Fox, G., Pallickara, S. The Narada Event Brokering System: Overview and Extensions / G. Fox, S. Pallickara // In Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications. — 2002. — C. 353–359.
72. Uyar, A., Pallickara, S., Fox, G. Towards an Architecture for Audio/Video Conferencing in Distributed Brokering Systems / A. Uyar, S. Pallickara, G. Fox // In Proceedings of International Conference on Communications in Computing. — 2013. — C. 17–23.
73. Fox, D. et. al. Peer-to-Peer Grids / D. Fox, S.-H. Gan№ n, S. Ko, S. Lee, M. Pallickara, X. Qiu, X. Rao, A. Uyar, M. Wang, W. Wu // Chapter 18 of Grid Computing: Making the Global Infrastructure a Reality. — 2013. — C. 471–490.
74. Balatskaya, L. et al. Software for Assessing Voice Quality in Rehabilitation of Patients after Surgical Treatment of Cancer of Oral Cavity, Oropharynx and Upper Jaw / L. Balatskaya, E. Choinzo№ v, S. Chizevskaya, E. Kostyuchenko, R. Meshcheryakov // Springer International Publishing Switzerland: SPECOM 2013, LNAI 8113. — 2013. — pp 294–301.
75. Ronzhin A., Karpov A., Kipyatkova I., Zelezny M. Client and Speech Detection System for Intelligent Infokiosk. // Springer-Verlag Berlin Heidelberg: TSD 2010, LNAI 6231. — 2010. — C. 560–567.
76. Ronzhin, A., Budkov, V. Multimodal Interaction with Intelligent Meeting Room Facilities from Inside and Outside / A. Ronzhin, V. Budkov // Springer-Verlag Berlin Heidelberg: NEW2AN/ruSMART 2009, LNCS 5764. — 2009. — C. 77–88.
77. Hsiao, K. L., Chiang, H. S., Huang, C. T. Continuous Usage Intention of Videoconferencing Software: A Case Study of One-to-Some Online Courses / K. L. Hsiao, H. S. Chiang, C. T. Huang // 27th International Conference on

Advanced Information Networking and Applications Workshops (WAINA). — 2013. —С. 990–995. — DOI:10.1109/WAINA.2013.24

78. Civanlar, M. R., Ozkasap O., Celebi T. Peer-to-Peer Multipoint Videoconferencing on the Internet / M. R. Civanlar, O. Ozkasap, T. Celebi // Signal Processing: Image Communication. — 2005. — Т. 20. — № 8. — С. 743−754. — DOI:10.1109/ICIP.2004.1421484
79. Ramdan, A. A., Munir R. Selective Encryption Algorithm Implementation for Video Call on Skype Client / A. A. Ramdan, R. Munir // 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA). — 2012. — С. 120–124. — DOI: 10.1109/TSSA.2012.6366035
80. Xu, Y., Yu, C., Li, J., Liu, Y. Video Telephony For End-Consumers: Measurement Study of Google+, iChat, and Skype / Y. Xu, C. Yu, J. Li, Y. Liu // IEEE/ACM Transactions on Networking. — 2014. — Т. 22. — С. 826–839. — DOI:10.1109/TNET.2013.2260354
81. Ma№ lache, F. B., McDowell, W., Rusu, O. Directory Cache Techniques for Efficient User Management / F. B. Ma№ lache, W. McDowell, O. Rusu // 10th Roedunet International Conference (RoEduNet). — 2011. — С. 1–5. — DOI:10.1109/RoEduNet.2011.5993696
82. Yan, S. et al. Infrastructure Management of Hybrid Cloud for Enterprise User / S. Yan, B. S. Lee, G. Zhao, D. Ma, P. Mohamed // 5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM). — IEEE. — 2011. — С. 1–6. — DOI:10.1109/SVM.2011.6096463
83. C. W. Lin Implementation of H.323 Multipoint Video Conference Systems with Personal Presence Control / C. W. Lin // International Conference on Consumer Electronics (ICCE). — 2000. — С. 108–109. — DOI:10.1109/ICCE.2000.854518
84. Потапова, Р. К., Собакина А. Н., Маслов А. В. Возможность идентификации говорящего по голосу в системе интернет-телефонии Skype / Р. К. Потапова, А. Н. Собакина, А. В. Маслов // Вестник

Московского Государственного Лингвистического Университета — 2013. — № 13. — С. 177–188.

85. Ронжин, Ал. Л., Будков, В. Ю., Ронжин, Ан. Л. Формирование профиля пользователя на основе аудиовизуального анализа ситуации в интеллектуальном зале совещаний / Ал. Л. Ронжин, В. Ю. Будков, Ан. Л. Ронжин // Труды СПИИРАН. — Вып. 23. — 2012. — С. 482–494.
86. Saveliev, A. I., Vatamaniuk, I. V., Ronzhin, A. L. Architecture of data exchange with minimal client-server interaction at multipoint video conferencing / A. I. Saveliev, I. V. Vatamaniuk, A. L. Ronzhin // International Conference on Next Generation Wired/Wireless Networking. — Springer International Publishing, 2014. — С. 164–174.
87. Barr E. T. et al. The oracle problem in software testing: A survey / E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, S. Yoo // IEEE transactions on software engineering. — 2015. — Т. 41. — № 5. — С. 507—525.
88. Jorgensen P. C. Software testing: a craftsman’s approach. — CRC press, 2016.
89. Mittal, H. Comparative Analysis of Automated Functional Testing Tools / H. Mittal // Journal of Network Communications and Emerging Tech№ logies. — 2016. — Т. 6. — № 6.
90. Erdogan, B. et al. Collaboration environments for construction: Management of organizational changes / B. Erdogan, C. J. Anumba, D. Bouchlaghem, Y. Nielsen // Journal of Management in Engineering. — 2013. — Т. 30. — № 3. — С. 04014002.
91. Banerjee, I. et al. Graphical user interface (GUI) testing: Systematic mapping and repository / I. Banerjee, B. Nguyen, V. Garousi, A. Memon, // Information and Software Tech№ logy. — 2013. — Т. 55. – № 10. – С. 1679–1694.

92. Wang, X., Zhou, B., Li, W. Model-based load testing of web applications / X. Wang, B. Zhou, W Li // Journal of the Chinese Institute of Engineers. — 2013. — Т. 36. — № 1. — С. 74–86.
93. Saveliev, A. I., Ronzhin, A. L. Algorithms and Software Tools for Distribution of Multimedia Data Streams in Client Server Videoconferencing Applications / A.I. Saveliev, A.L. Ronzhin // Pattern Recognition and Image Analysis. — Springer, 2015. — Т. 25. — № 3. — С. 517–525.
94. Басов О.О., Карпов А.А. Анализ стратегий и методов объединения многомодальной информации / О. О. Басов, А. А. Карпов // Информационно-управляющие системы. — 2015. — № 2. — С. 7–14.
95. Ронжин Ал. Л., Ронжин Ан. Л., Будков В. Ю. Каскадная система дистанционного аудиовизуального мониторинга и записи говорящих / Ал. Л. Ронжин, Ан. Л. Ронжин, В. Ю. Будков // Актуальные направления развития систем охраны, специальной связи и информации для нужд государственного управления: IX Всероссийская межведомственная научная конференция: материалы и доклады. — Орёл: Академия ФСО России. — 2015 — Ч. 9. — С. 93–95.
96. Карпов, А. А. Средства речевого и многомодального человеко-машинного взаимодействия для ассистивных информационных технологий / А. А. Карпов // Речевые технологии. — 2014. — № 3–4. С. 48–59.
97. Басов, О.О., Ронжин, А.Л. Методика поэтапного внедрения полимодальных инфокоммуникационных систем / О.О. Басов, А.Л. Ронжин // Научные ведомости БелГУ. Сер. История. Политология. Экономика. Информатика. — 2015. — Вып. 33/1. — № 1 (198). — С. 131–137.
98. Den Hertog D. Interior point approach to linear, quadratic and convex programming: algorithms and complexity / Hertog D. Den — Springer Science & Business Media. — 2012. — Т. 277.

99. Rodríguez, P. et al. Cross-device Videoconferencing based on Adaptive Multimedia Streams / P. Rodríguez, A. Alonso, J. Salvachúa, E. Barra, J. Cerviño // JMPT. — 2013. — T. 4. — № 1. — C. 14–24.

100. Ines, D. E. et al. Streaming mobile multimedia optimization for video-conferencing scenarios / D. E. Ines, K. Fujikawa, E. Kawai, H. Sunahara // Advances in Multimedia, 2009. MMEDIA'09. First International Conference on. — IEEE, 2009. — C. 80–85.

101. Ruiz, P. M., Garcia, E. Adaptive multimedia applications to improve user-perceived QoS in multihop wireless ad-hoc networks / P. M. Ruiz, E. Garcia //Proc. PIRMC. — 2002. — T. 3. — C. 1467–1471.

102. Ali, S., Hemmati, H. Model-based testing of video conferencing systems: challenges, lessons learnt, and results / S. Ali, H. Hemmati // 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation. — IEEE, 2014. — C. 353–362.

103. Joskowicz, J. Automation of subjective video quality measurements/ J. Joskowicz // Proceedings of the Latin America Networking Conference on LANC 2014. — ACM, 2014. — C. 7.

104. Anand, S. An orchestrated survey of methodologies for automated software test case generation / S. Anand // Journal of Systems and Software. — 2013. — T. 86. — №. 8. — C. 1978–2001.

105. Yuan, H., Du, H. The Design and Implementation of Qt-based Cross-platform Video Conferencing Remote Control / H. Yuan, H. Du // Communications and Network. — 2013. — T. 5. — №. 01. — C. 73.

106. Zhang, Y. Cross-Platform Product Usability and Large Screen User Experience: A Teleconference System U&E Research / Y. Zhang // International Conference of Design, User Experience, and Usability. — Springer International Publishing, 2014. — C. 469–479.

107. Valin, J. M., Vos K., Terriberry T. Definition of the Opus audio codec / J. M. Valin, K. Vos, T. Terriberry. — 2012. — №. RFC 6716.

108. Park, Y., Lee, H., Jeon, B. Performance Analysis of Open Web Video Codec VP8 / Y. Park, H. Lee, B. Jeon // IEIE Transactions on Smart Processing & Computing. — 2013. — Т. 2. — №. 2. — С. 86–96.

109. Choudhary, S., Varshney, P. A Study of Digital Video Compression Techniques / S. Choudhary, P. Varshney // PARIPEX-Indian Journal of Research. — 2016. — Т. 5. — №. 4.

110. Ismail, Y. FPGA Implementation of Fast and Efficient CODEC for H. 264/AVC Real Time Video Applications / Y. Ismail // International Journal of Tech№ logy Diffusion (IJTD). — 2016. — Т. 7. — №. 1. — С. 32–50.

111. Schwartz C. Performance analysis of the trade-off between signalling load and power consumption for popular smar tphone apps in 3G networks / C. Schwartz // University of Würzburg, Tech. Rep. — 2013. — Т. 485.

112. Александров, В. В., Кулешов, С. В. Программируемый мир / В. В. Александров, С. В. Кулешов // Материалы 1-ой Международной конференции Технологическая перспектива в рамках Евразийского пространства: новые рынки и точки экономического роста. — СПб: Издательство НПК «РОСТ», 2015. — С. 163–168.

113. Александров, В. В., Кулешов, С. В. Программируемый мир: информационные технологии XXI века / В. В. Александров, С. В. Кулешов // Сборник статей международной научно-практической конференции. — СПб: ООО «Книжный дом», 2015. — С. 19–24.

114. Александров, В. В. Формирование и развитие информационной инфраструктуры инновационного развития Санкт- Петербурга / В. В. Александров, В. И. Воробьев, С. В Кулешов, Д. К. Левоневский, В. С. Марков, Р. Р.Фаткиева, Р. М. Юсупов // в монографии «Перспективные направления развития науки в Петербурге». — СПб.: Изд-во ИП Пермяков С.А., 2015. — 543 с.

115. Одоевский, С. М., Калюка, В. И., Степаненко, В. В. Оптимизация распределения частотно-энергетических ресурсов сети широкополосного радиодоступа / С. М. Одоевский, В. И. Калюка,

В. В. Степаненко // XXI Международная научно-техническая конференция Радиолокация, навигация, связь. — 2015. — С. 1052–1059.

116. Одоевский, С. М., Яровикова, О. В., Оптимизация управления распределением трафика на смежных уровнях сетевой архитектуры / С. М. Одоевский, О. В. Яровкикова // III Международная научно-техническая и научно-методическая конференция «Актуальные проблемы инфотелекоммуникаций в науке и образовании». — 2015. — С. 323–328.

Приложение А. Копии актов внедрения результатов диссертационной работы

СТЭЛ КОМПЬЮТЕРНЫЕ СИСТЕМЫ
Общество с ограниченной ответственностью «Стэл – Компьютерные Системы»/ООО «Стэл КС»
ИНН 7705180443, КПП 770501001, ОКПО 48502969, ОГРН 1027739242310
Россия, 105082, г. Москва, ул. Б. Почтовая, д. 55/59, стр. 1.
тел./факс 8 (495) 775 5123, 8 (499) 261 1643, 261 3551
www.stel.ru stel@stel.ru

УТВЕРЖДАЮ
Генеральный директор ООО «Стэл КС»
М. Ю. Андреев
14 сентября 2015 г.

АКТ

внедрения результатов диссертационной работы Савельева Антона Игоревич «Алгоритмы и программные средства распределения мультимедийных потоков данных в клиент-серверных приложениях видеоконференцсвязи» на соискание ученой степени кандидата наук

Комиссия в составе: председателя Балакина А.В. и членов Широковой А.М. и Ямшинина А.А., рассмотрев представленные материалы диссертационных исследований Савельева А.И., установила следующее.

1. Основные положения диссертационной работы были использованы при проведении научно-исследовательских работ компании по проекту разработки системы «Аудиопротокол», обеспечивающей захват данных с различных медиа-источников и распознавание речевого потока.
2. На основе предложенных соискателем алгоритмов распределения мультимедийных потоков данных в клиент-серверных приложениях видеоконференцсвязи разработаны новые решения для видеоконференцсвязи, обеспечивающие быструю передачу мультимедийных данных от одного клиента к другому. Указанная возможность на программном уровне, в частности, реализована на основе предложенных Савельевым А.И. алгоритмов распределения

мультимедийных потоков в клиент-серверных приложениях видеоконференцсвязи.

3. Тестирование и обучение разработанной системы распределения мультимедийных потоков данных проводилось на современном мультимедийном комплексе аудио- и видеооборудования и технологической площадке, предоставленной компанией ООО «Стэл КС».
4. Разработанная система распределения мультимедийных потоков данных в клиент-серверных приложениях видеоконференцсвязи является перспективной и обладает заявленными техническими характеристиками по точности определения указанных свойств информации.
5. Система распределения мультимедийных потоков данных в клиент-серверных приложениях видеоконференцсвязи позволяет разрабатывать на ее основе новые высокотехнологичные интеллектуальные платформы с возможностью автоматизации функций абонента, а ее внедрение – повысить эффективность их коммуникативного взаимодействия.

Председатель комиссии: Балакин А.В.

Члены комиссии: Широкова А.М.

Ямшинин А.А.

УТВЕРЖДАЮ
Проректор по научной и
инновационной деятельности
Заслуженный деятель науки РФ
д.т.н., профессор,
_____Е.А. Крук
«05» 09 2016г.

АКТ

об использовании результатов диссертационной работы научного сотрудника лаборатории речевых и многомодальных интерфейсов Федерального государственного бюджетного учреждения науки Санкт-Петербургского института информатики и автоматизации Российской академии наук Савельева Антона Игоревича в учебном процессе Санкт-Петербургского Государственного университета аэрокосмического приборостроения

Мы, нижеподписавшиеся, начальник учебного управления, канд. техн. наук, доцент В.А.Матьяш, директор института инновационных технологий в электромеханике и робототехнике, д-р. техн. наук, профессор В.Ф.Шишлаков, доцент кафедры управления в технических системах, канд. техн. наук, доцент М.В. Бураков составили настоящий акт в том, что результаты диссертационной работы Савельева А.И. на тему «Архитектуры, алгоритмы и программные средства обработки потоков многомодальных данных в пиринговых веб-приложениях видеоконференцсвязи», представленной на соискание ученой степени кандидата технических наук,

внедрены в учебный процесс университета, а именно:

1. Архитектуры клиентской и серверной частей в системах видеоконференцсвязи, отличающиеся применением пирингового принципа коммуникации, обеспечивающие построение речевых и многомодальных интерфейсов для данного типа телекоммуникационных приложений.

2. Алгоритмы взаимодействия клиентской и серверной частей системы видеоконференцсвязи и установления соединений между клиентами по протоколу WebRTC, обеспечивающие распределение сокетов и обработку их потоков данных на сервере, поддерживая взаимодействие между группами клиентов.

3. Программные средства клиентской и серверной частей веб-приложения видеоконференцсвязи обеспечивают кроссплатформенность за счет реализации веб-интерфейса с использованием языков HTML, CSS и JavaScript, а также сокращение передаваемых потоков мультимедийных данных по протоколам HTTP и WebSocket между клиентом и сервером.

4. Методика тестирования пиринговых систем видеоконференцсвязи, включающая алгоритмы функционального тестирования и набор тестов, оценивающих потребляемые ресурсы при нагрузочном тестировании.

Разработанные программные и технические решения, реализующие обработку потоков многомодальных данных в пиринговых веб-приложениях видеоконференцсвязи используются в учебном процессе по направлению: 27.03.04 (220400) – «Управление в технических системах» при выполнении дипломного проектирования, в лекционном материале и лабораторном практикуме учебных курсов «Методы искусственного интеллекта», «Нейронные сети и экспертные системы».

Разработанный веб-приложение видеоконференцсвязи, использующееся в натурных экспериментах и лабораторном практикуме, позволил наглядно

продемонстрировать возможности естественного человеко-машинного взаимодействия, способы проектирования пользовательских интерфейсов к мобильных приложениям и повысить интерес обучаемых к предмету.

Начальник учебного управления,
канд. техн. наук, доцент В.А.Матьяш

Директор института инновационных технологий
в электромеханике и робототехнике,
д-р. техн. наук, профессор В.Ф.Шишлаков

Доцент кафедры управления в технических системах,
канд. техн. наук, доцент М.В. Бураков

САНКТ-ПЕТЕРБУРГСКИЙ
АКАДЕМИЧЕСКИЙ УНИВЕРСИТЕТ

Лермонтовский пр., д. 44 литер А, Санкт-Петербург, 190103
Тел. (812) 575-03-00 Факс (812) 575-02-70
ОКПО 39410814, ОГРН 1027810240260, ИНН/КПП 7826001459/783901001

***E-mail:** rector@spbume.ru URL: http//www.spbume.ru*

15 СЕН 2016 № 2594
На №_____ от _____

АКТ

об использовании результатов диссертационной работы на соискание ученой степени кандидата технических наук Савельева Антона Игоревича «Архитектуры, алгоритмы и программные средства обработки потоков многомодальных данных в пиринговых веб-приложениях видеоконференцсвязи»

Комиссия в составе проректора по науке Санкт-Петербургского академического университета Г.А. Костина, директора института электронного обучения В.В. Нарышевой и доцента кафедры информационных технологий и математики В. В. Курлова, составила настоящий акт о том, что результаты диссертационной работы «Архитектуры, алгоритмы и программные средства обработки потоков многомодальных данных в пиринговых веб-приложениях видеоконференцсвязи» были использованы в процессе обучения студентов Санкт-Петербургского академического университета в следующем виде:

1. Основные положения диссертационной работы Савельева А. И. были использованы при подготовке лекционного материала для студентов Санкт-Петербургского академического университета.
2. Веб-приложение пиринговой видеоконференцсвязи, демонстрирующее возможности разработанного математического и программного обеспечения обработки потоков многомодальных данных, было использовано при проведении практических работ с целью более качественного усвоения студентами обучающихся по направлению 09.03.03 Прикладная информатика.
3. Алгоритмы и программные средства обработки потоков многомодальных данных использовались при проведении занятий со студентами института электронного обучения с применением дистанционных технологий.

Проректор по науке САУ
доктор технических наук, доцент Г.А. Костин

Директор института
электронного обучения САУ В.В. Нарышева

Доцент кафедры информационных
технологий и математики,
кандидат технических наук, доцент В. В. Курлов

15.09.2016

ФАНО РОССИИ

Федеральное государственное бюджетное учреждение науки Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН)

14 линия, 39, Санкт-Петербург, 199178
Телефон: (812) 328-33-11, факс: (812) 328-44-50
E-mail: spiiran@iias.spb.su, http://www.spiiran.nw.ru
ОКПО 04683303, ОГРН 1027800514411
ИНН/КПП 7801003920/780101001

«28» сентября 2016 г. № 073- 09/211/399

На № _____

УТВЕРЖДАЮ
Директор СПИИРАН
Член-корреспондент РАН
_____Р.М. Юсупов
«28» 09 2016 г.

А К Т

об использовании результатов диссертационной работы Савельева Антона Игоревича «Архитектуры, алгоритмы и программные средства обработки потоков многомодальных данных в пиринговых веб-приложениях видеоконференцсвязи» в научно-исследовательских работах СПИИРАН

Комиссия в составе: председателя д.т.н., проф. Б.В. Соколова, членов комиссии: к.т.н. В.П. Дашевского и к.воен.н., доцента Е.П. Силлы, рассмотрев представленные материалы:

1. автореферат и диссертационную работу Савельева Антона Игоревича;
2. отчетную документацию научно-исследовательских работ лаборатории речевых и многомодальных интерфейсов;

установила, что:

1. Основные положения диссертационной работы Савельева Антона Игоревича были использованы при проведении НИР, выполняемых:
   - по государственному контракту № 07.514.11.4139 «Математическое и программное обеспечение автоматического анализа и распознавания разговорной русской речи и диаризации дикторов», 2012-2013 гг.;
   - по государственному контракту № 14.740.11.0357 «Разработка принципов и инновационных информационных технологий для взаимодействия пользователей с интеллектуальным пространством», 2010-2012 гг.;
   - по государственному контракту № 14.616.21.0056 «Исследование и разработка системы аудиовизуального распознавания речи на базе микрофона и высокоскоростной видеокамеры», 2015-2016 гг.;
   - по гранту РФФИ № 15-07-06774 «Разработка методов обработки и обмена мультимедийными данными в пиринговых веб-приложениях многоточечной видеоконференцсвязи», 2015-2017 гг;
   - по гранту РФФИ № 13-08-00741 «Разработка методов и кроссплатформенных программных средств аудиовизуального сопровождения мобильных мероприятий», 2013-2015 гг.
2. Разработанные в ходе НИР подходы, архитектуры, алгоритмы и программное обеспечение основаны на научных результатах, представленных в диссертационной работе, в том числе:

- Архитектура пирингового веб-приложения многоточечной видеоконференцсвязи, включающая методы автоматической обработки аудиовизуальных данных участников сеанса связи, обеспечивающие сокращение передаваемого трафика, оптимизацию конфигурационной сети пиринговых клиентских соединений и снижение нагрузки на серверную часть веб-приложения, отвечающего в основном только за соединение клиентов, при этом передача самих мультимедийных потоков осуществляется напрямую между устройствами участников связи.
- Алгоритм соединения и обмена данными в пиринговом веб-приложении видеоконференцсвязи, исключающий потери данных и использующий серверное приложение прежде всего для буферизации служебных сообщений и позволяющий сформировать конфигурационную динамическую сеть пиринговых соединений для передачи требуемого мультимедийного контента от ближайших в сети клиентов.
- Алгоритм многомодального анализа поведения пользователя, использующий передаваемый аудиовизуальный поток данных от клиента для определения его речевой активности, оптимизации соединений конфигурационной пиринговой сети для передачи данных выступающих участников слушателям и персонифицированной настройки пользовательского интерфейса с учетом истории сеанса связи.
- Кроссплатформенное приложение многоточечной видеоконференцсвязи с пиринговой архитектурой обмена данными между клиентскими приложениями, учитывающими активность участников на основе многомодального анализа их поведения для оптимизации объемов передаваемого мультимедийного трафика между клиентскими устройствами.

3. По результатам НИР и диссертационного исследования опубликовано 30 научных работ, включая 6 статьи в журналах из перечня ВАК (Автоматика и телемеханика, Информационно-управляющие системы, Труды СПИИРАН, Automation and Remote Control и др.), 9 статей в зарубежных изданиях, индексируемых в базе Scopus (Pattern Recognition and Image Analysis, Speech and Computer, Interactive Collaborative Robotics, Human-Computer Interaction, Internet of Things, Smart Spaces, and Next Generation Networking).

Председатель комиссии
Заместитель директора по научной работе,
д.т.н. профессор — Б.В. Соколов

Члены комиссии
Старший научный сотрудник
лаборатории автономных
робототехнических систем, к.т.н. — В.П. Дашевский

Ученый секретарь,
к.воен.н. доцент — Е.П. Силла