Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

(Университет ИТМО)

Факультет **Программной инженерии и компьютерной техники**

Образовательная программа **Программно-информационные системы 2017**

Направление подготовки (специальность) **09.03.04 Программная инженерия**

О Т Ч Е Т

о производственной, преддипломной практике

Тема задания: Разработка платформы для совместного видео-стриминга

Обучающийся: Ореховский Антон, группа P3417

Руководитель практики от университета: Кореньков Юрий Дмитриевич, к.т.н. ассистент

Практика пройдена с оценкой _____

Подписи членов комиссии:

_____
(подпись)

_____
(подпись)

_____
(подпись)

Дата _____

Санкт-Петербург
20 21

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Сложно представить жизнь современного человека без интернета и технологий, которые его образуют. Существует огромное количество возможностей, которые предоставляет нам интернет: доступ к свободной информации в любое удобное время, возможность отсылать и получать электронные письма, возможность участвовать в телеконференциях и удаленных встречах, возможность совершать покупки, не выходя из дома, играть в сетевые компьютерные игры и многое другое.

Среди прочего большую значимость для людей имеет потоковое вещание. Потоковое вещание — это мультимедиа контент (аудио-, видео-, а также текстовые данные), которые пользователи непрерывно получают от поставщика услуг [1].

В рамках данной работы наибольшее значение имеет потоковое видео вещание в режиме реального времени. Наиболее важные решения, интересующие нас в данной работе таковы:

1) Массовый онлайн стриминг. Под данной категорией подразумеваются платформы способные транслировать видео в реальном времени во всемирную паутину обширному, условно неограниченному количеству пользователей (ограничивают количество пользователей технические возможности платформы). Наиболее популярные сервисы — YouTube, Twitch, Facebook Live и прочие.

2) Видеоконференции. Под данной категорией подразумеваются решения, обеспечивающие двустороннюю передачу, обработку, преобразование и представление видеоданных между несколькими пользователями, количество которых ограничено, независимо от их месторасположения в режиме реального времени. Skype, Zoom, Google Hangouts Meet — наиболее яркие решения, которые предоставляют данный формат вещания.

Среди существующих платформ нет такой, которая сочетала бы свойства обеих категорий платформ потокового вещания в режиме реального времени: возможность участия нескольких пользователей, медиа-поток которых бы мог собираться в единый, возможность администрирования потоков пользователей и создания сцены (например, динамическое размещение потоков в одной сцене при изменении количества потоков). Такие платформы здесь и далее будем называть виртуальной аппаратной.

Целью данной работы является обеспечение возможности динамической композиции транслируемого видеопотока под управлением нескольких пользователей.

Для достижения данной цели были поставлены следующие задачи:

1. проанализировать существующие средства организации передачи потокового видео в реальном времени (массовый онлайн стриминг и конференции);
2. изучить способы организации совместных трансляций с помощью существующих средств и выявить недостатки;
3. проанализировать средства, используемые для организации передачи потокового видео (композиция сцены, библиотеки кодеков для транскодирования, протоколы передачи);
4. разработать решение для совместного видео-стриминга.

Работа была реализована по следующим этапам:

1. инструктаж обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка,
2. изучение способов организации совместных трансляций с помощью существующих средств, изучить средства организации передачи потокового видео,
3. разработка решения для совместного видео-стриминга,
4. анализ полученных навыков и компетенций, оформление отчёта.

1 Инструктаж обучающегося

Первостепенную важность имела необходимость прохождения инструктажей. Этот этап достаточно важен, так как объясняет все возможные нестандартные ситуации и способы реагирования на них.

Часть инструктажей я прошел в Университете ИТМО в рамках подготовки к работам по написанию ВКР (охрана труда, техника безопасности, пожарная безопасность), часть инструктажей была проведена моим научным руководителем (правила внутреннего трудового распорядка, первичный инструктаж на рабочем месте).

2 Изучение теоретической части

Данный этап начался с изучения сценариев доставки видео контента. В самом начале были выделены следующие сценарии:

- просмотр видео в интернете,
- видеосвязь, конференции,
- трансляции с веб-камер,
- стриминг.

Среди выделенных сценариев только два представляют интерес. Стриминг, а также видеосвязь и конференции интересны тем, что только в них присутствует генерация потока, во время которой возможно применение композиции сцены.

Дальнейший интерес представляли решения, использующие эти сценарии. Среди рассмотренных решений наиболее близким к обеспечению поставленной цели является OBS с использованием плагинов.

Был произведен детальный анализ возможностей ПО OBS, который дал понять, что в целом, данное решение позволяет собирать локальные элементы и комбинировать их в единый медиа поток и публиковать его на rtmp сервер из коробки, но может быть расширено плагинами. Например, можно вещать по другим транспортным протоколам или подключать внешние источники данных. Стоит упомянуть плагин OBS-studio-webrtc [2], который способен получать данные из видеоконференций, основанных на стандарте WebRTC реализованных в медиа сервере Janus. С подобными плагинами возможность создания виртуальной аппаратной становится реальной, хотя и требует большого количества накладных ресурсов и сторонних серверов.

Однако данное решение не является оптимальным, так как требует внешних серверов либо для подключения к ним и сбора данных, либо для публикации одной из возможных композиций для обеспечения возможности удаленного администрирования каждой из них.

Был произведен их анализ, после которого стало понятно, что использовать композицию сцен в рамках видеоконференций возможно, при условии композиции сцены

программным энкодером и генерации виртуальной камеры, на основе которой будет генерироваться поток для видеосвязи.

Дальнейшее изучение теоретической части было сфокусировано на средствах передачи потокового видео. Для начала были рассмотрены средства композиции сцены, среди которых были выделены 2 способа:

- композиция вручную,
- композиция с использованием специализированных библиотек.

В целом, после изучения данной области, стало понятно, что, зная то, как расположены данные для соответствующих пикселей, и что они означат, можно с легкостью манипулировать ими:

- масштабировать,
- изменять размер добавляя или обрезая граничные пиксели,
- заменять пиксели другими с целью создания наложения видео компонентов друг на друга

Было отмечено то, что оба способа выполняют одинаковые функции, за исключением того, что вручную необходимо выполнять все самому, что с легкостью может привести, а также то, что готовые могут использовать более эффективные алгоритмы манипуляции над изображением. Так, например, можно вручную с легкостью изменить масштаб изображения методом ближайшего соседа, но он может иметь результат хуже, чем, например, билинейный метод [3].

Далее были рассмотрены методы доставки видео контента. Рассмотрев данную область, были получено представление в разнице между несколькими подходами. Наиболее интересные это:

- протоколы управления потоком, передачи и описания данных,
- WebRTC.

Отличие способов состоит в том, что протоколы лишь описывают стандарты, которе можно использовать, а WebRTC может обеспечивать коммуникацию между участниками и обмен медиаданными, лишая нас необходимости изобретать велосипед [4].

Последним этапом изучения практической части было изучение библиотек кодеков для транскодирования. Среди рассмотренных библиотек были описаны лишь 3:

- libav (FFmpeg)
- vlclib
- Gstreamer

Было выявлено, что в данных библиотеках процесс работы на видео очень схож. Пример работы над видео приведен на рисунке 1. Различие между данными библиотеками состоит в разности гибкости их использования.

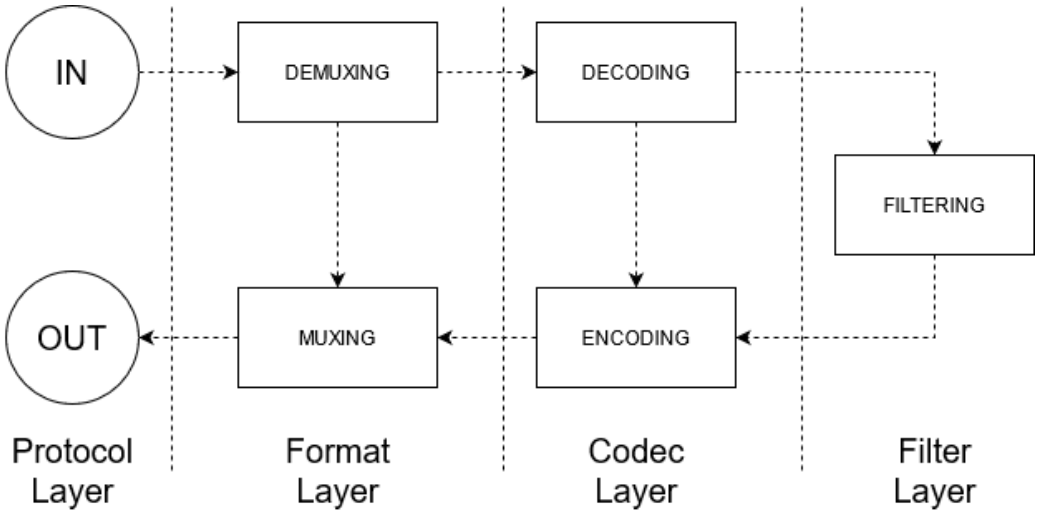

Рисунок 1 – шаблон процесса работы над видео на примере библиотеки libav

Данный шаблон говорит, о том, что у потока работ над данными всегда присутствует входные и выходные точки. Данные которые поступают в эти точки должны быть завернуты в контейнер, например mp4. Уровнем ниже определяется то, что видео должны быть закодированы, например кодировщиком x264 [5]. Только лишь на уровне фильтрации происходит манипуляция над видео.

По окончанию данного этапа, был определен набор инструментов, которые будут задействованы в реализации решения. В качестве ядра программы решено использовать фреймворк Gstreamer из-за его универсальности и гибкости [6]. Так как решение, разрабатываемое в рамках работы, должно быть максимально гибким и иметь возможность собирать медиа потоки из всевозможных источников, среди которых могут быть и источники видеоконференцсвязи, конкуренты фреймворка Gstreamer отпадают. Как упоминалось в ходе анализа, во фреймворке Gstreamer существуют реализации элементов на любую стандартную

задачу. К сожалению, реализация конференцсвязи не является стандартной проблемой. Среди доступных плагинов не было найдено такого, который позволил бы получать медиа поток от каждого нового участника видеосвязи, в связи с чем возникает необходимость в удобном способе организации видеосвязи, у которого есть простой и надежный API, который желательно был бы широко используемым и унифицируемым. Среди всех проанализированных решений только WebRTC подходит под эти условия. Таким образом были определены используемые решения для данной работы, основанное на комбинации двух технологий:

- Gstreamer – для удобного манипулирования видео,
- WebRTC – для поддержки видеоконференцсвязи.

3 Выполнение практической части

В ходе данного этапа было построена архитектура, ядром которой является фреймворк Gstreamer, который отвечает за манипуляцию над видео, а также было реализована связь между двумя видами приложений: веб-приложений и конвейерного приложения (здесь и далее под конвейерным подразумевается разрабатываемое решение, которое предоставляет медиапоток на вход конвейеру Gstreamer или получает данные из него).

С использованием выше рассмотренных компонентов, разработана программная архитектура, основывающаяся на идее «конвейерного приложения». Данное приложение, основываясь на WebRTC и библиотеке Gstreamer позволяет осуществлять совместный видеостриминг – то есть видео-трансляции с участием многих пользователей одновременно. Предложенное решение отличается от существующих тем, что, прямое объединение медиапотоков всех участников трансляции на стороне одного экземпляра приложения позволяет каждому из пользователей участвовать в композиции сцены транслируемого видео. После этого сформированный с участием нескольких пользователей медиа-поток может быть транслирован на широкую аудиторию посредством публичных сервисов ретрансляции, как показано на рисунке 2.

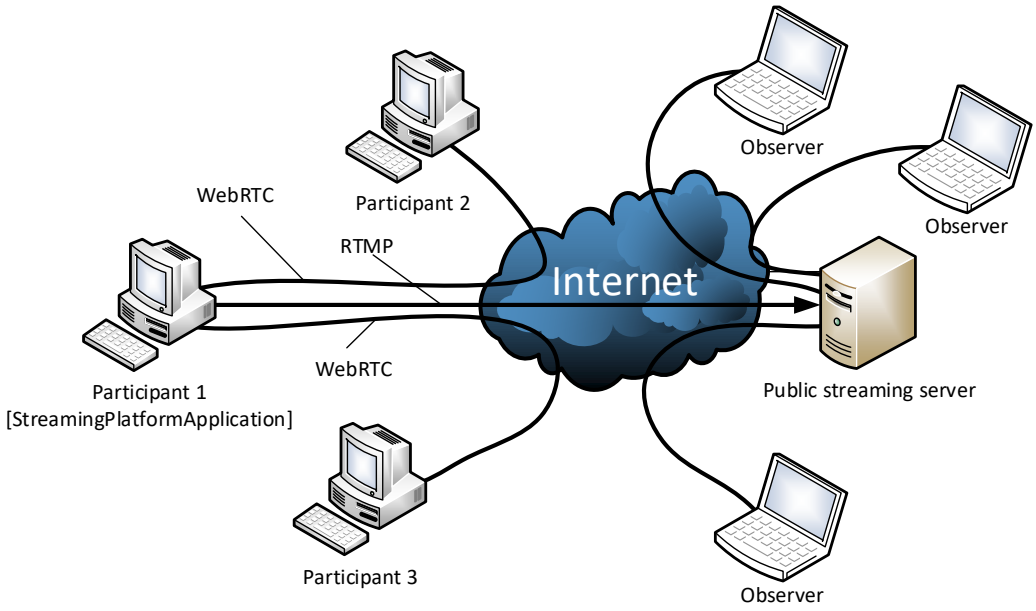

Рисунок 2 – Трансляция динамически объединенного видеоряда посредством публичного сервиса.

Также возможны и другие сценарии использования предложенного решения на основе «конвейерного приложения», например, дополнительная композиция и публикация существующего медиапотока нескольким пользователям без участия вторичной ретрансляции, как показано на рисунке 3.

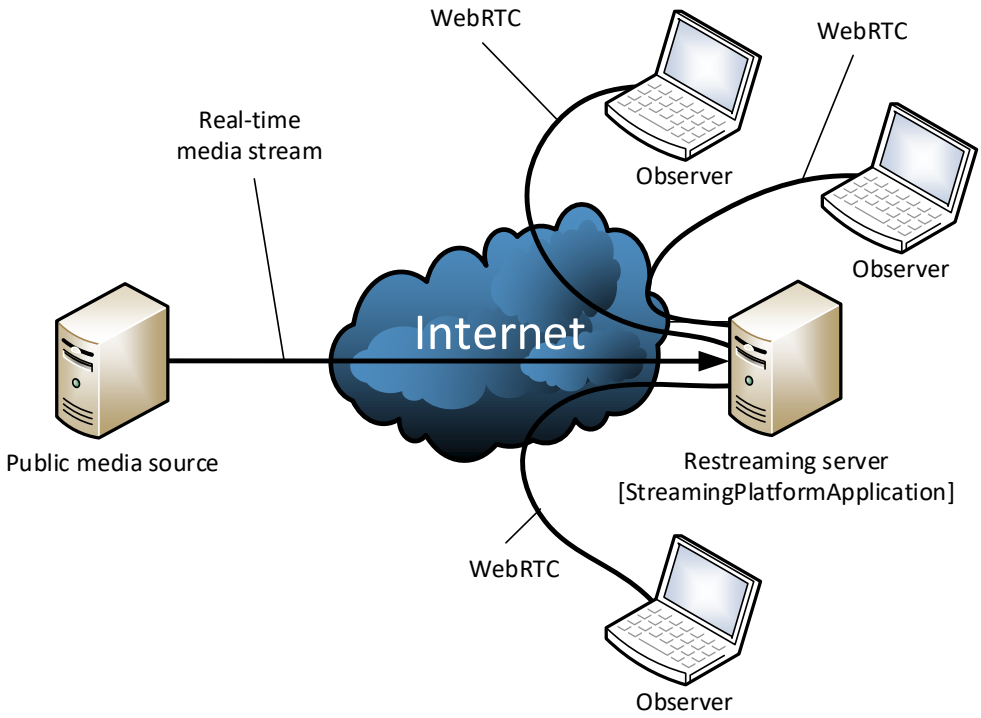

Рисунок 3 – Трансляция существующего видеоряда нескольким пользователям без участия дополнительных средств

Таким образом, разработанное решение позволяет реализовывать различные сценарии совместного видеостриминга без дополнительных требований к участникам трансляции и используемым ими программным обеспечением.

Следующим действием в данном этапе стала разработка решений, которые смогли бы обеспечить поток данных, описанный на рисунке 2 и 3.

Для начала был реализован сценарий использования WebRTC – RTMP. Данный сценарий представляет из себя процесс генерации видеопотоков на стороне пользователей веб-приложения. Веб-приложение при этом отправляет сгенерированные медиа-потоки посредством WebRTC на сторону службы, которой является конвейерное приложение. Приложение, получая медиаданные от каждого пользователя, передает их в конвейер, где далее

все медиапотоки будут совмещены в один единственный, который далее будет передан в rtmp сервер.

В рамках разработанного прототипа, демонстрирующего предложенный подход к реализации совместного видео-стриминга, конвейерное приложение конфигурирует конвейер и WebRTC соединения исходя из переданного ему на вход XML файла, в котором описаны все участники. Результат работы данного сценария представлен на рисунке 4.

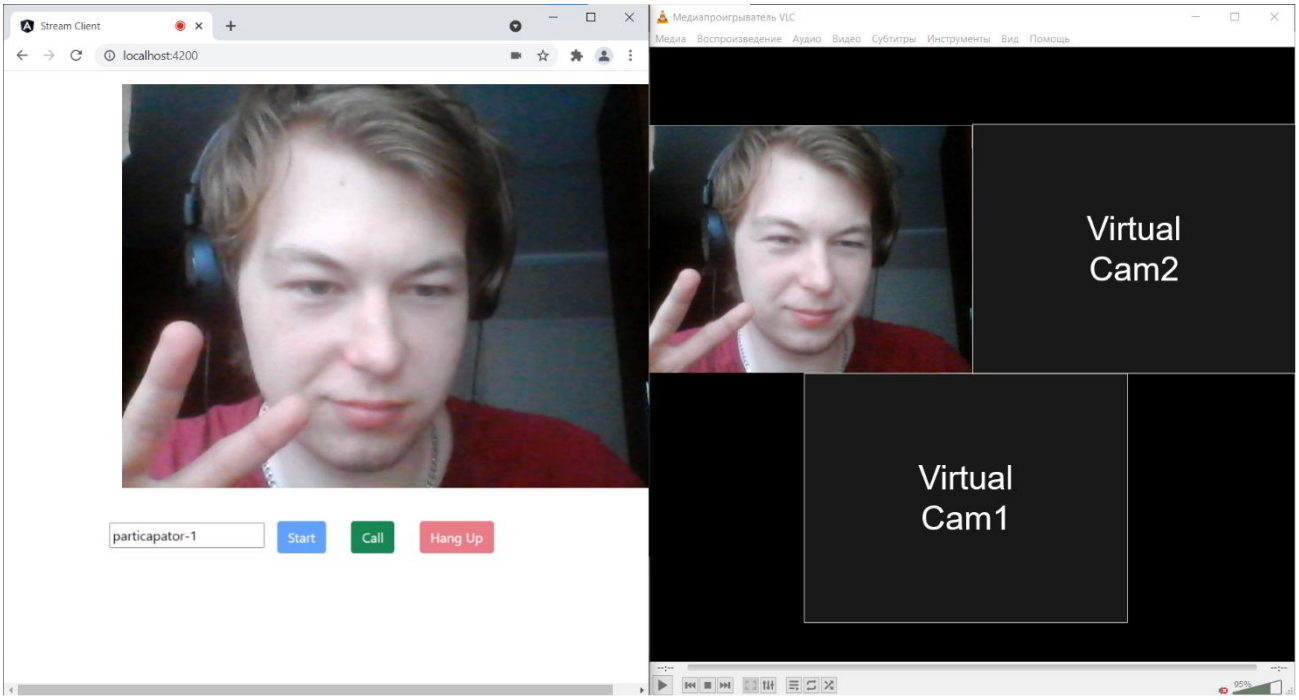

Рисунок 4 – результат работы сценария WebRTC – RTMP

Человек, представленный на изображении, является автором данной работы, то есть мной. К сожалению, количество физических камер в моем распоряжении было ограничено одной штукой, поэтому мне пришлось использовать виртуальные камеры, хотя это никаким образом не повлияло на работу данного решения. Можно заметить, что кадр со мной, сгенерированный на стороне клиента, отличается от кадра со мной на стороне RTMP сервера. Это обусловлено принципом работы RTMP протокола. Дело в том, что перед тем, как видео можно будет отдать потребителю RTMP трафика, необходимо накопить буфер, который и будет отправлен потребителю. Накопление этого буфера сильно отличается, и может даже доходить до нескольких секунд.

Последней на очереди в рамках данного этапа была реализация сценария RTMP - WebRTC. Данный сценарий использования разработанного решения позволяет получать

видеопоток из RTMP сервера, и затем направлять данный поток каждому из всех подключенных к конвейерному приложению посредством стандарта WebRTC пользователей.

В данном случае также используется файл конфигурации XML описанный в предыдущем подразделе, но используется он только с целью добавления нового адреса для прослушивания веб-сокетом с целью обеспечения общения каждой пары клиентов. Результат работы данного сценария приведен на рисунке 5.

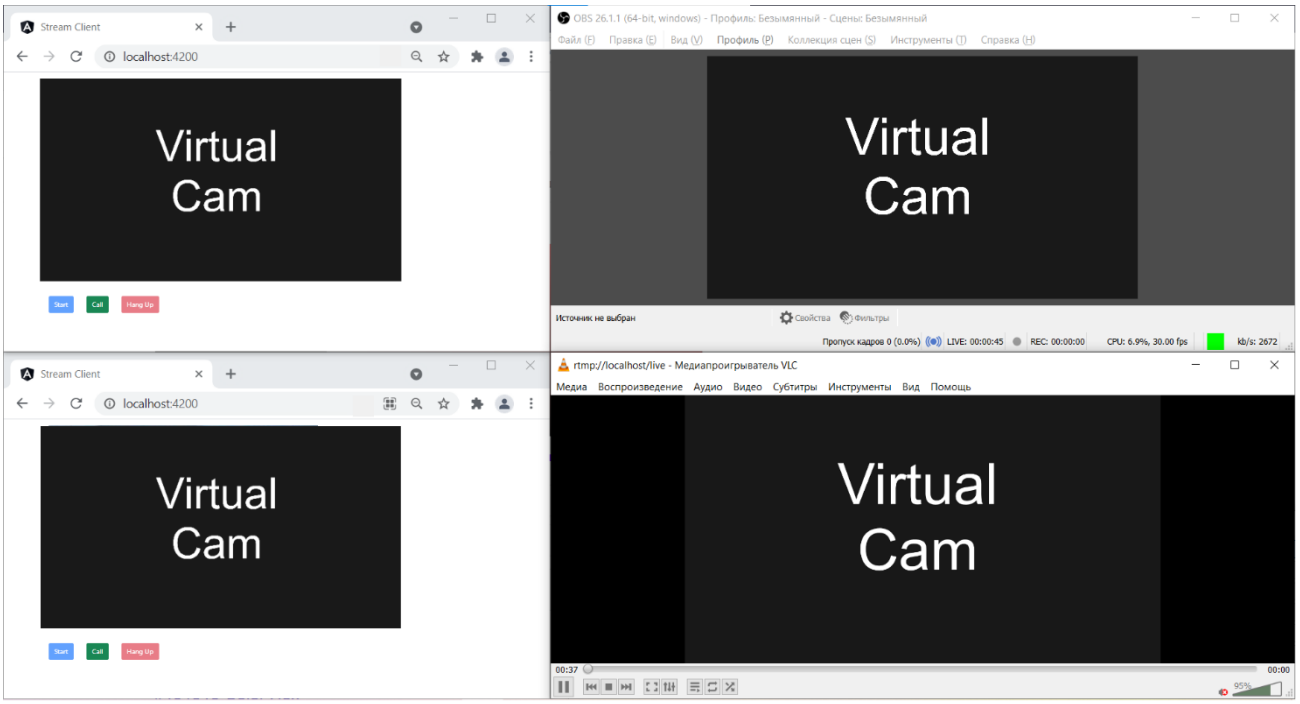

Рисунок 5 – результат работы сценария RTMP – WebRTC

На рисунке 5 видны следующие элементы:

- программа OBS, которая генерирует видеопоток, схожий с тем, что использовался в виртуальных камерах, и далее его публикует,
- проигрыватель vlc, который подключен к серверу, на который идет публикация видеопотока,
- 2 клиента веб-приложения, которое отличается от использованного в предыдущем сценарии, получают данные от конвейерного приложения, общаясь с ним по стандарту WebRTC.

4 Документирование

По итогам выполнения преддипломной практики с работ, связанных с написанием выпускной квалификационной работы, мною были получены навыки работы с фреймворком Gstreamer, преимущественно с его программным интерфейсом, а также был получен опыт работы со стандартом видеоконференцсвязи WebRTC.

Работая с фреймворком Gstreamer, я научился основным концепциям и методам манипулирования медиапотоками внутри конвейерной среды данного фреймворка. Наиболее важным навыком я считаю умение связывать созданные конвейеры и оконечные элементы, которые оперируют данными входящими или исходящими из приложения.

Опыт работы в рамках работы со стандартом WebRTC был получен во время конструирования архитектуры для обеспечения пиринговой видеосвязи. Стоит отметить, что данная архитектура связывает 2 реализации:

- реализация веб-приложения на языке TypeScript,
- реализация приложения для платформы .NET, выполненная на языке C#.

Именно реализацию под 2 разные платформы я считаю наиболее важной и значимой в контексте оценивания полученных навыков.

Очевидно, во время и после выполнения практической части выпускной квалификационной работы, необходимо было документировать прогресс и полученные результаты. Документирование производилось в соответствии с требованиями [8].

Подготовку к написанию отчета я начал с описания целей и задач, так как было необходимо понять, что в процессе работы над практическим заданием необходимо получить. Также было решено описать актуальность цели – почему необходимо ее решить и чем разработанное решение может помочь. Помимо этого, было размечено содержание, так как это помогает наглядно увидеть структуру будущей работы, и помогает определиться со значимостью той или иной ее части.

Затем, после изучения всей необходимой документации, была разработана архитектура, которая решает поставленные задачи. После того как я убедился в работоспособности данной архитектуры, я подробно описал все проделанные шаги и их результаты, которые я сделал, а

также я описал основную часть, а именно то, что, как и почему я делал так как я сделал, а не иначе.

После того, как основная часть была готова, я сформулировал и записал выводы о проделанной работе, после чего оформил весь написанный текст в соответствии с требованиями.

ЗАКЛЮЧЕНИЕ

В ходе данной работы были проанализированы существующие решения осуществляющие различные сценарии по доставке видео, среди которых не было выявлено тех, которые могут совмещать в себе сценарии стриминга, а также видеоконференцсвязи. Возможности по созданию виртуальной аппаратной были выявлены только у одного решения, но только с существенными ограничениями. Таким образом было подтверждена актуальность проблем и важность поставленной цели.

Среди проанализированных решений организации потокового видео были выявлены 2 решения, которые помогли бы достичь поставленной цели. На стыке выбранных технологий было разработано решение, которое может осуществлять получение данных из любого необходимого источника, будь то удаленный rtmp-сервер или сырые данные из приложения. Важно отметить то, что среди данных из приложения могут быть данные, полученные из видеоконференцсвязи по стандарту WebRTC.

Работоспособность данного решения также была установлена и продемонстрирована на примере двух сценариев:

- WebRTC – RTMP
- RTMP – WebRTC

СПИСОК ЛИТЕРАТУРЫ

1. E. P. J. Tozer. Broadcast Engineer's Reference Book –2013 –1049 с.
2. Исходный код расширения для ПО OBS–studio WebRTC–URL: https://github.com/CoSMoSoftware/OBS-studio-webrtc
3. А. О.Трубаков, М. О.Селейкович. Сравнение интерполяционных методов масштабирования растровых изображений–2017 –97с.
4. Altanai. WebRTC Integrator's Guide -2014 -382 с.
5. Iain E. G. Richardson. The H.264 Advanced Video Compression Standard -2011 - 346 с.
6. Документация фреймворка Gstreamer–URL: https://gstreamer.freedesktop.org
7. Эндрю Троелсен. Язык Программирования C# 5.0 и платформа .NET 4.5 -6-е издание –2013–1310 с.
8. Требования к выпускным квалификационным работам – 2020 – 35 с.