## Слайд 2

В наше время видео вещание имеет широкую популярность. Будь то начинающий стример или профессиональная команда телеканала: все сталкиваются с проблемой генерации медиапотока.

Выбор бесплатных способов создания медиапотока ограничивается двумя категориями. Ими являются либо программы для видеоконференций, либо программные энкодеры, которые позволяют создавать видео сцену на стороне вещателя.

При этом, среди бесплатных способов, отсутствует такой, который помог бы совместить функции вышеперечисленных методов, а именно композиция сцены и возможность участия нескольких пользователей в данной сцене.

## Слайд 3

Исходя из указанной проблемы была поставлена цель: обеспечить возможность динамической композиции транслируемого видеопотока под управлением нескольких пользователей.

Для достижения данной цели были поставлены задачи представленные на слайде.

## Слайд 4

На данном слайде представлены примеры решений, которые позволяют генерировать медиапоток. Представителями категории программных энкодеров являются OBS и XSplit.

Видеоконференции довольно распространены в наше время, и все имеют представление о них. Представители – Zoom и Google Meet

## Слайд 5

Несмотря на то, что бесплатных решений решающих выявленную проблему нет, существуют платные решения, которые нацелены на профессиональное вещание. Наиболее популярные профессиональные решения представлены на слайде.

## Слайд 6

На данном слайде приведен анализ вышеперечисленных решений. В таблице видно, что возможность композиции сцен присуща только программным энкодерам. Возможность проведения же совместных трансляций присутствуют в энкодерах в случае с vMix и Wirecast. Но так как для этого необходим доступ в интернет, было отмечено, что данные решения не могут работать в сети интранет.

В результате анализа было выявлено отсутствие необходимого решения указанной проблемы.

## Слайд 7

Для решения проблемы необходимо было провести анализ использующихся протоколов и стандартов для удобной доставки медиапотока в разрабатываемое приложение. Были рассмотрены

* протоколы управления потоком данных, которые описывают то, как два участника видеосвязи будут обмениваться данными с целью получения данных
* протоколы передачи данных, которые отвечают за организацию передачи данных
* Протокол описания данных предназначен для описания сессии и потоковых данных, которые передаются в ее контексте

Стандарты совмещающие протоколы агрегируют низкоуровневые функции протоколов.

Так, например, используя стандарт WebRTC разработчику не стоит волноваться о том, как следует упаковать и отправлять данные, ведь все это может быть выбрано на основе окружения пользователей автоматически.

## Слайд 8

После получения медиапотока необходимо предоставить возможность транскодировки, а также композиции медиапотоков. Средства, решающие данную задачу, должны иметь API, а также возможность получения удаленных медиапотоков.

Среди таких решений наиболее ценен фреймворк GStreamer, который помимо базовых манипуляций, имеет возможность организации сложных конвейеров по обработке видео. Такие конвейеры позволяют разделять, клонировать медиапотоки, совмещать их, и отправлять на разное число удаленных серверов.

## Слайд 9

Для обеспечения возможности совместного видеовещания было решено использовать стандарт WebRTC, так как он не только позволяет агрегировать низкоуровневые функции протоколов, но и не требует наличия дополнительного ПО у пользователей.

Для манипуляции на медиапотоками был выбран фреймворк GStreamer из-за его гибкости и большого количества плагинов. Большим плюсом было также наличие удобной документации и большого количества примеров работы с ним.

## Слайд 10

После того, как были определены необходимые технологии, была составлена программная архитектура решения, которая помогла бы достичь поставленную цель. Ядром решения является контроллер, который опирается на 3 компонента:

* веб приложение, которое обеспечивает возможность участия в совместном видеовещании
* компонент, который является участником видеосвязи со стороны сервера, то есть разработанного приложения
* А также компонент, который обеспечивает манипуляцию над медиапотоком, то есть его композицию, обмен и транскодирование.

## Слайд 11

Была реализована процедура сопряжения для webrtc-соединений, проиллюстрированная данной диаграммой последовательностей.

После начала WebRTC сессии, происходит выделение вебсокета, после чего происходит создание классов WebRTC соединения на стороне веб-приложения и сервера.

После этого запрашиваются пользовательские медиа (микрофон и камера) на стороне веб приложения, которые связываются с классом WebRTC соединения.

Далее происходит обмен пакетами Offer SDP и Answer SDP, которые поставляются через вебсокет.

После этого происходит отправка видео кадров, сгенерированных в компоненте UserMedia без использования вебсокета на обработку в компонент StreamReceiver.

## Слайд 12

После получения нового кадра компонентом StreamReceiver, данный кадр передается в компонент GstInteractor, который кладет данный кадр в очередь. В момент, когда конвейеру понадобится новый кадр, он вызывает соответствующий метод у компонента GstInteractor, который забирает видеокадр из очереди и передает его на вход элементу GstPipeline.

Следует отметить то, что получение и запрос кадра асинхронны относительно друг друга.

## Слайд 13

Используя приложение с указанной архитектурой, удалось осуществить сценарий WebRTC-RTMP, в процессе которого каждый захваченный у пользователя медиапоток направлялся в разработанное решение при помощи стандарта WebRTC, где далее все потоки комбинировались в один единственный поток, который далее публиковался на удаленный RTMP сервер.

## Слайд 14

Так же был реализован сценарий RTMP – WebRTC, в процесс которого видео было ретранслировано с RTMP сервера по стандарту WebRTC всем участникам соединения.

## Слайд 15

По итогам данной работы была разработана программная архитектура, сочетающая возможности WebRTC и технологий видео-стриминга без использования внешних служб

Реализовано и протестировано на примере двух сценариев приложение, обеспечивающее возможность создания совместных видеотрансляций под управлением нескольких пользователей

## Слайд 16

Что касается возможных путей совершенствования данного решения, то можно добавить возможности администрирования.

Так, например, в данный момент все входящие соединения регулируются только на стороне веб приложения, то есть отсутствует возможность отключить пользователя на стороне приложения.

Также отсутствует графический веб интерфейс, который позволил бы администрировать работу приложения удаленно несколькими людьми

Также приложения работает только с протоколом RTMP. Но, благодаря плагинам, в Gstreamer-е можно с легкостью добавить поддержку других протоколов или запись на жесткий диск.