## Слайд 2

В наше время интернет имеет большое значение в жизни большого числа людей. 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 видео. Сценарии доставки видео можно разделить на 4 группы:

* Просмотр видео в интернете
* Видеосвязь, конференции
* Трансляции с веб-камер
* Стриминг

Выбор бесплатных способов создания медиапотока ограничивается двумя категориями. Ими являются либо программы для видеоконференций (Skype, Zoom, Google Meet), либо так называемые программные энкодеры, которые позволяют создавать видео сцену на стороне поставщика услуг и вещать сгенерированный медиапоток на широкий круг пользователей.

При этом, среди бесплатных способов, отсутствует таковой, 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 было поставлена цель: обеспечить возможность динамической композиции транслируемого видеопотока под управлением нескольких пользователей.

Для достижения данной цели были поставлены следующие задачи:

* Проанализировать существующие средства доставки потокового видео;
* Выявить способы создания совместных трансляций и их недостатки;
* Выбрать пути реализации различных компонентов создаваемого решения;
* Разработать решение для совместного видео-стриминга.

## Слайд 4

На данном слайде представлены решения, которые позволяют генерировать медиапоток. Представителями категории программных энкодеров являются OBS и XSplit, которые позволяют производить композицию сцены, то есть размещения на ней множества элементов и определения их свойств (например, позиция элемента и его прозрачность). При этом, среди элементов могут быть веб камера, камера захвата экрана, текст и изображения. Следует отметить, что в целом данные решения имеют схожий функционал за исключением того, что программа XSplit ориентирована на игровой стриминг.

Видеоконференции довольно распространены в наше время, и все имеют представление о них. В рамках данной работы были рассмотрены Zoom и Google Meet как наиболее яркие представители данного сегмента решений.

## 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.

Тем не менее, у данных решений так же есть изъяны, помимо их цены. Так, например, для организации совместных трансляций необходим доступ в интернет, что исключает возможность функционирования данных решений в сети интранет.

## Слайд 6

На данном слайде приведен анализ вышеперечисленных решений. В таблице видно, что возможность композиции сцен присуща только программным энкодерам, которые объединены под категорией стриминга. 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, что не совсем верно. Безусловно, данные решения могут работать в сети интранет и вещать на локальный сервер, как например и OBS или XSplit, но возможность совместного вещания при этом отпадает.

Помимо указанного на таблице, стоит отметить то, что Google Meet использует стандарт WebRTC, который позволяет устанавливать пиринговые соединения и вести конференции без необходимости в сторонних серверах для передачи медиа потока.

## Слайд 7

Выявив все достоинства и недостатки существующих решений, необходимо было рассмотреть существующие протоколы и способы передачи видео, для выявления наиболее удобного способа обмена видео.

Были просмотрены следующие типы протоколов:

* 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 следует упаковать данные и по какому протоколу стоит данные передавать и как при этом следует описывать сессию передачи данных, веди все это уже может быть выбрано на основе окружения пользователей автоматически.

Среди всех описанных стандартов, именно WebRTC был разработан для создания видеоконференций, что, несомненно, выделяет его среди других решений в рамках данной работы.

## Слайд 8

В рамках поставленной цели получение потока является лишь частью необходимого процесса манипуляции. После его получения необходимо предоставить возможность кодировки/декодировки, а также композиции медапотоков. Существуют специальные средства, которые помогают выполнить данную задачу. Прмеры данных средств приведены на таблице. Из всех средств можно выделить те, которые имею API и не имеют его. Последние интереса в рамках данной работы не представляют. Так же средства должны иметь возмодность подучения удаленных медаипотокв для организации видеосвязи и публикации медиапотокв на удаленные сервера. Среди таких решений наиболее ценен фреймворк GStreamer, которые позволяет не только получать видео, транскодировать видео и осуществлять композицию, но также 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 наличие дополнительного ПО у пользователей.

Для манипуляции на медиапотоками был выбран фреймворк GStreamer из-за его гибкости и большого наличия плагинов. Большим плюсам 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 компонента, который обеспечивает манипуляцию над медиапотоком, то есть его композицию, обмен и транскодирование.

Решение было разработано на языке C#, при этом оно зависит от библиотеки MS.MixedReality.WebRTC, которая обеспечивает возможность использования стандарта WebRTC, а также зависит от библиотек GlibSharp и GstSharp которые вместе предоставляют доступ к фреймворку GStreamer.

## Слайд 11

Данная диаграмма показывает процесс установления WebRTC соединения, которые было инициировано веб приложением. В рассматриваемой диаграмме используется только одно направление связи – передача медиа пользователя на сервер. В качестве компонента, который обеспечивает возможность установления соединения, выступает WebSocket. Он передает сообщения между участниками, которые помогает установить соединение.

После начала WebRTC сессии, происходит выделение вебсокета, после чего происходит создание классов, отвечающих за пиринговое соединение WebRTC на стороне веб-приложения и сервера. После этого запрашиваются пользовательские медиа (микрофон и камера) на стороне веб приложения, которые связываются с классом соединения. После этого происходит генерация Offer SDP пакета, который говорит о том, что веб приложения хочет инициировать соединение, а также о возможных путях его установления. В ответ на это сообщение может быть сгенерирован Answer SDP пакет, который говорит о том, что сервер готов к установлению соединения. После того, как веб приложения получило данный пакет происходит отправка видео кадров, сгенерированных в компоненте UserMedia, которые поставляются в компонент StreamReceiver при помощи компонентов пирингового соединения.

## 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 для потребления, он вызывает соответствующий метод у компонента GstInteractor, который забирает видеокадр из очереди и передает его на вход элементу GstPipeline. Следует отметить то, что получение кадра и событие, сигнализирующее о необходимости передачи кадра в конвейер асинхронны относительно друг друга.

## Слайд 13

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

На скриншоте справа можно увидеть то, как бал захвачен поток пользователя и далее был воспроизведён итоговый поток с RTMP сервер при помощи программы VLC player.

## Слайд 14

Так же был реализован сценарий RTMP – WebRTC, в процесс которого видео было ретранслировано с RTMP сервера по стандарту WebRTC всем участникам соединения. На скриншоте справа приведен пример того, как поток генерируется в программе OBS. Далее этот поток публикуется на RTMP сервер. Среди подписчиков сервера присутствует VLC player, а также разработанное решение, которое ретранслирует данный поток двум 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 возможный 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.

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

Также в данном решение безопасность осуществляется только на уровне https при общении с вебсокетом. При этом любой желающий, имеющий доступ к веб приложению может быть участником соединения.

Также приложения работает только с протоколом RTMP. Но благодаря большому количеству плагинов в Gstreamer можно с легкостью добавить поддержку RTSP, TCP и других протоколов. Существуют также плагины, которые позволят записывать медиапоток на жесткий диск.