УДК 004.773.5

И. В. Лобов, канд. физ.-мат. наук, ст. науч. сотр., e-mail: lobov@ihep.ru,
В. Г. Готман, мл. науч. сотр., e-mail: vladislav.gotman@ihep.ru,
НИЦ "Курчатовский институт" ФГБУ ГНЦ РФ —
Институт физики высоких энергий, Московская обл., г. Протвино

Использование контейнера Ogg для организации потоковой трансляции в реальном времени над протоколом HTTP методом опережающей загрузки

Рассмотрены характеристики и общая схема организации потоковой трансляции в реальном времени над протоколом HTTP. Применительно к общей схеме дан обзор существующих лидирующих технологий в этой области. Рассмотрена технология использования контейнера Ogg для организации трансляции в реальном времени над протоколом HTTP методом опережающей загрузки. Проведено ее сравнение с существующими основными технологиями (Apple HLS, Adobe HDS, Microsoft Smooth Streaming, MPEG DASH).

Ключевые слова: *трансляция, поток, опережающая загрузка, Ogg, Apple HLS, Adobe HDS, Microsoft Smooth Streaming, MPEG DASH*

Введение

В последнее время в мире приобрели широкую популярность веб-трансляции презентаций в реальном времени, основой чему послужило бурное развитие технологий потоковых трансляций над протоколом HTTP. В настоящей работе делается сравнение предложенной в работе [1] идеи использования метода опережающей загрузки медиапотока формата Ogg [2] для организации живой трансляции с лидирующими технологиями живого потокового вещания. С этой целью выявлены общие свойства, присущие любой живой трансляции, а именно: разбиение трансляции на отдельные фрагменты, наличие у фрагмента метки времени и наличие метаданных для всей трансляции. Эти свойства определяют общую схему организации живой трансляции. Выполнен обзор существующих лидирующих технологий живого потокового вещания с точки зрения соответствия общей схеме. Показано, что технология трансляции медиапотока формата Ogg методом опережающей загрузки удовлетворяет общей схеме организации живой трансляции. Сделан вывод о логической эквивалентности всех рассмотренных технологий потоковых трансляций.

1. Общая схема организации трансляции в реальном времени

Под презентацией будем далее понимать действие, самостоятельно происходящее во времени, это может быть программа новостей, лекция, мастер-класс и т.п. Далее будет рассматриваться случай односторонних трансляций, когда клиент может только смотреть и слушать презентацию, но не участвовать в ней.

Задача трансляции презентации в реальном времени заключается в неискаженной передаче медиапотоков (синхронизованных между собой временных последовательностей видеокадров и аудиосэмплов) от источника трансляции к конечному потребителю (клиенту) по каналу связи. Напрямую, в необработанном виде, эти потоки посылать нельзя ввиду большого объема информации в них и ограничений на пропускную способность канала связи. Поэтому используют различные алгоритмы сжатия медиапотоков. Соответственно, на стороне клиента после приема сжатых данных необходимо их декодировать и превратить в обычные видеокадры и аудиосэмплы, которые уже можно воспроизводить.

На технологию трансляции презентации налагают различные требования, наиболее важные

из них: 1) возможность одновременного подключения к трансляции нескольких клиентов в произвольные моменты времени; 2) возможность продолжения воспроизведения трансляции после потери ее части при передаче по каналу связи; 3) синхронное воспроизведение аудио- и видеопотоков на стороне клиента. Из этих требований следует несколько важных выводов.

Во-первых, трансляция презентации не должна осуществляться непрерывным потоком октетов без дополнительной их организации. В противном случае клиент, подключающийся к середине трансляции, не сможет понять, где закончился один видеокадр (аудиосэмпл) и где начался другой. Это же относится и к случаю потери по линии связи части октетов. Если после потери части трансляции клиент может понять, где начинается первый кадр или сэмпл, то он сможет продолжить воспроизведение. Поэтому трансляцию удобно проводить порциями, имеющими четкие границы и снабженными (желательно) контрольной суммой. Далее будем называть эти порции *обобщенными фрагментами*.

Обобщенный фрагмент это передаваемая по каналу связи единица медиаконтейнера, содержащая часть закодированного медиапотока. Обобщенный фрагмент (далее просто *фрагмент*) может содержать как целое число медиаэлементов (видеокадров, аудиосэмплов), так и часть медиаэлемента. Фрагменты могут содержать один тип медиаэлементов (например, только видео) или несколько типов вместе (например, аудио и видео в одном фрагменте). Фрагменты являются независимыми друг от друга в том смысле, что фрагмент содержит всю необходимую информацию для извлечения из него закодированных медиаэлементов. Фрагмент можно передать клиенту за одну транзакцию (например, за один запрос GET [3, С. 35]), но это необязательно. Клиент может получить фрагмент от сервера по частям, а затем "склеить" полученные части между собой.

Во-вторых, клиент должен знать, в какие моменты времени необходимо воспроизводить содержащиеся во фрагментах медиаэлементы. Для этого фрагмент снабжается *меткой времени*. При задержке или потере фрагментов метка времени предотвратит временной сдвиг и рассинхронизацию между различными медиапотоками (например, аудио и видео).

В-третьих, для возможности воспроизведения трансляции клиент должен получить *метаданные* медиапотока. Метаданные необходимы клиенту для первоначальной инициализации медиапотока. В метаданные входит информация о параметрах видео и аудио (размер кадра, частота кадров, частота дискретизации, применяемые кодеки и т.п.).

На рис. 1 изображена общая схема организации простейшей трансляции в реальном времени. От источника трансляции, например видеокамеры со встроенным микрофоном, на сервер поступают два потока медиаданных — видеокадры и аудиосэмплы. Каждый из этих потоков поступает со своей определенной частотой. Перед началом трансляции сервер создает метаданные трансляции и хранит их в отдельном буфере. По мере поступления медиаданных с источника трансляции, сервер их кодирует и помещает во фрагменты. Когда клиент подключается к трансляции, он вначале получает метаданные, затем начинает загружать фрагменты и воспроизводить их. На рис. 1 изображена гипотетическая ситуация, когда одному видеокадру всегда соответствуют два аудиосэмпла, а один фрагмент содержит три видеокадра и шесть аудиосэмплов.

2. Обзор важнейших технологий организации живого потокового вещания над протоколом HTTP

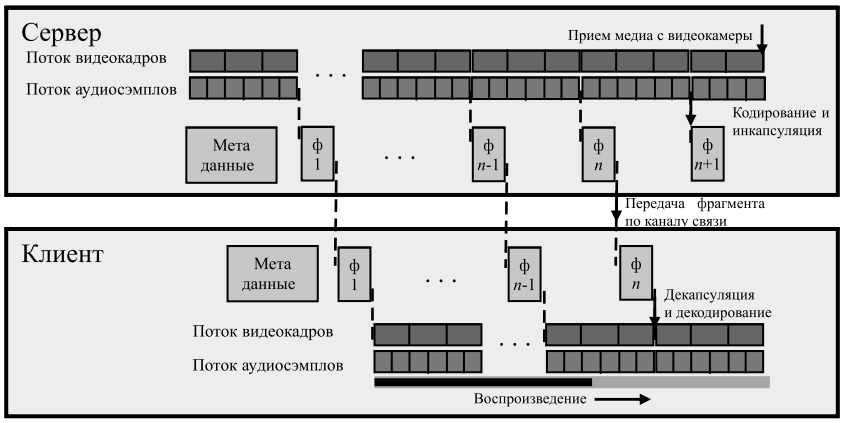

Рис. 1. Общая схема организации трансляции в реальном времени для случая, когда видео и аудио инкапсулируются в один фрагмент

На сегодняшний день лидирующими технологиями живого потокового вещания над протоколом HTTP являются *Apple HTTP Live Streaming* [4], *Adobe HTTP Dynamic Streaming* [5], *Microsoft Smooth Streaming* [6], *MPEG Dynamic Adaptive Streaming over HTTP* [7]. Дадим общую характеристику перечисленных технологий и проведем краткий обзор их существенных особенностей применительно к приведенной выше общей схеме. Адаптивное вещание в рамках настоящей статьи не обсуждается.

Общая характеристика обозреваемых технологий

Каждая из обсуждаемых далее технологий живого потокового вещания имеет свое собственное понятие минимальной единицы потока, передаваемой по каналу связи. Все эти определения соответствуют данному в разд. 1 понятию обобщенного фрагмента, дополненного требованием независимой загрузки. Приведем краткое описание процесса трансляции по этим технологиям. По мере прохождения трансляции сервер создает фрагменты. Клиент подключается к серверу и делает постоянные запросы для загрузки метаданных и фрагментов, используя HTTP-метод GET. С точки зрения клиента, каждый загружаемый фрагмент является файлом, хотя фрагменты необязательно должны физически находиться на сервере в виде отдельных файлов. Фрагмент содержит целое число видеокадров и/или аудиосэмплов, закодированных видео- и аудиокодеками и инкапсулированных в контейнер.

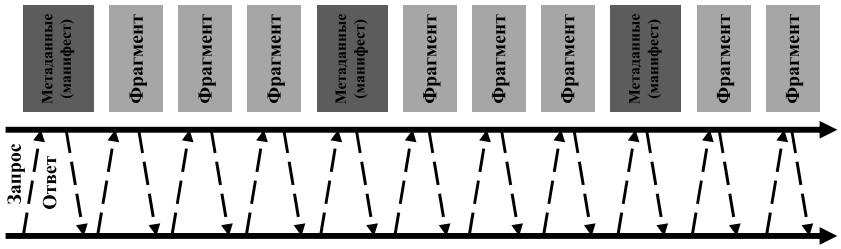

Рис. 2. Схема взаимодействия клиента и сервера, применяемая в существующих лидирующих технологиях организации живого потокового вещания над протоколом HTTP

Некоторые из этих технологий (*Apple HLS*, *Adobe HDS*) накладывают ограничение на обобщенный фрагмент — требование независимого декодирования фрагментов друг от друга.

В обсуждаемых технологиях роль метаданных играет файл, называемый манифестом. Сервер помещает в манифест общую информацию о свойствах воспроизводимого потока (применяемых кодеках, разрешении экрана и др.) и информацию о фрагментах (URL и метки времени). С каждым новым фрагментом сервер обновляет манифест.

Отличительной особенностью обозреваемых технологий является наличие у каждого фрагмента своего уникального URL, что делает фрагменты независимо загружаемыми. Манифест необходим клиенту не только для возможности декодирования принимаемого потока фрагментов, но также для получения URL отдельных фрагментов (посредством шаблонов или прямых ссылок). Поэтому для обновления списка доступных фрагментов клиенту необходимо периодически обновлять манифест. Отметим, что для технологии *Microsoft Smooth Streaming* манифест запрашивается клиентом только один раз — при подключении клиента к трансляции.

Схема взаимодействия клиента и сервера представлена на рис. 2.

Как видно из рис. 2, инициатором приема фрагментов у всех перечисленных технологий является клиент. Сервер при этом выполняет пассивную роль отсылки данных в ответ на запросы клиента.

Apple HTTP Live Streaming

Технология *HTTP Live Streaming (HLS)* разработана компанией *Apple* в 2009 г. и является одной из самых ранних и простых технологий ведения потоковых трансляций над *HTTP*. *HLS* входит составной частью в собственное программное обеспечение *Apple*, такое как *QuickTime, Safari, OS X* и *iOS*. *Apple HLS* может использовать в качестве клиента любые современные браузеры с поддержкой *HTML5* [8] и *Media Source Extensions*. Это дает возможность системам введения трансляций по технологии *HLS* поддерживать максимально широкое число клиентских устройств.

Во время ведения трансляции по технологии *HLS* презентация записывается в непрерывную цепочку файлов. В технологии *Apple HLS* данное в разд. 1 определение обобщенного фрагмента соответствует файлу формата *MPEG-2 TS* [9] с расширением .ts, называемым *медиасегментом* [4, С. 5]. Видео и аудио содержатся в одном фрагменте. Для сжатия чаще всего используют кодеки — H.264 [10] (видео) и AAC (аудио) [11].

Метаданные потока содержатся в манифесте — текстовом файле формата *m3u* или *m3u8*. Манифест содержит следующее:

- служебную информацию (EXT-X-STREAM-INF) о кодеках (CODECS), разрешении экрана (RESOLUTION), битрейте (BANDWIDTH) и т.д.;
- ссылки на фрагменты, доступные для скачивания клиентом с *web*-сервера, представляющие собой имена реальных файлов;
- метки времени (EXTINF), представляющие собой длительности фрагментов.

По мере прохождения презентации сервер кодирует новые видеокадры и аудиосэмплы во фрагменты. С каждым новым готовым фрагментом сервер формирует новый манифест. Сервер может ограничивать число фрагментов в манифесте, удаляя из списка самые старые фрагменты, и поэтому общее число фрагментов будет постоянным. Наименьшее число фрагментов равно 3 [4, С. 32]. Пример файла-манифеста *Apple HLS* для живой трансляции:

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=
150000,RESOLUTION=416x234,
CODECS = "avc1.42e00a,mp4a.40.2"
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:10,
fileSequence1.ts
#EXTINF:10,
fileSequence2.ts
#EXTINF:10,
fileSequence3.ts
```

Здесь заданы три фрагмента, длительностью 10 с каждый.

После подключения к трансляции клиент вначале загружает манифест и тем самым получает список ссылок на доступные в данный момент фрагменты. Затем клиент начинает загружать сами фрагменты. После завершения загрузки всех фрагментов из полученного списка клиент запрашивает новый манифест, получает новый список фрагментов и опять начинает их загружать. Этот процесс продолжается до конца трансляции. Клиент воспроизводит загруженные фрагменты непрерывно, без каких-либо задержек между ними.

Microsoft Smooth Streaming

Smooth Streaming (SS) является технологией ведения потоковых трансляций, разработанной компанией Microsoft. Впервые технология была анонсирована и применена для трансляции готового статического видео на Олимпийских играх в 2008 г. В 2010 г. была опубликована первая официальная версия протокола. Технология *Smooth Streaming* использует проигрыватель Microsoft Silverlight в качестве основного клиента.

Во время ведения трансляций по технологии Microsoft SS вся презентация записывается на сервере в один файл формата PIFF [12], который представляет собой расширение стандарта MPEG-4 Part 12 [13]. Формат файлов этого стандарта называют также фрагментированным MP4 (fMP4). Файлы формата fMP4 состоят из атомов — объектно-ориентированных строительных блоков, каждый из которых снабжен уникальным идентификатором типа этого блока [13, С. 2]. Файлы презентации содержат либо только видео .ismv, либо только аудио .isma, либо все вместе .ismv. Основными кодеками *Microsoft SS* являются H.264 и AAC.

Данному во введении определению обобщенного фрагмента в технологии *Microsoft SS* соответствует независимо загружаемая единица медиа, содержащая целое число видеокадров или аудиосэмплов, называемая в протоколе *Smooth Streaming* фрагментом [6]. Фрагмент состоит из двух атомов: moof с метаданными фрагмента и mdat с медиаданными. Физически фрагмент — это часть файла презентации. По мере прохождения презентации видеокадры и аудиосэмплы сервер кодирует во фрагменты и добавляет к файлу презентации. Различные потоки, например аудио или видео, в *Microsoft SS* именуют треками. Фрагмент содержит только один трек.

Временная метка фрагмента в *Microsoft SS* есть абсолютное время первого видеокадра или аудиосэмпла во фрагменте трека. Отличительной особенностью фрагмента *Microsoft SS* является то, что в нем содержатся две (или более) временные метки. Одна временная метка задает время самого фрагмента в треке. Она содержится в параметре "FragmentAbsoluteTime" атома Tfxd/moof [6]. Другая задает время начала следующего (или нескольких последующих) фрагмента трека. Она содержится в параметре "FragmentAbsoluteTime" атома Tfrf/moof фрагмента [6].

Метаданные потока содержатся в файле-манифесте клиента .ismc, имеющего формат XML [14]. Помимо метаданных, манифест *Microsoft SS* содержит шаблон построения ссылок для загрузки клиентом фрагментов с сервера. В общем виде шаблон имеет следующий вид:

$$\text{http://\{ service uri\}/QualityLevels(\{bitrate\})/Fragments(video=\{start time\}),} \quad (1)$$

где service uri — базовый URL сервера; bitrate — битрейт потока; start time — временная метка фрагмента. Также в манифесте клиента содержатся временные метки и длительности нескольких самых свежих фрагментов.

При старте трансляции сервер создает несколько первоначальных фрагментов с медиаданными и манифест клиента, содержащий информацию об этих фрагментах. По мере поступления новых медиаданных сервер создает новые фрагменты и обновляет манифест клиента. Подключившемуся клиенту сервер передает сначала манифест, а потом запрашиваемые клиентом фрагменты. Для связи фрагментов с файлом презентации сервер использует специальный файл (.ism), называемый серверным манифестом.

При подключении к трансляции клиент скачивает манифест клиента, затем начинает загружать описанные в манифесте фрагменты, формируя ссылки на фрагменты по шаблону (1) и делая запросы серверу. После того, как клиент скачает все указанные в манифесте фрагменты, ему необходимо сформировать ссылку на следующий фрагмент, для этого он должен узнать метку времени следующего фрагмента. Как было сказано

выше, это есть параметр "FragmentAbsoluteTime" атома Tfrf последнего загруженного клиентом фрагмента. После получения каждого очередного фрагмента клиент определяет таким способом метки времени следующих фрагментов этого трека, формирует запросы по шаблону (1) и получает новые фрагменты. Таким образом, клиент загружает файл-манифест только один раз, все остальное время он поддерживает свой внутренний манифест.

Adobe HTTP Dynamic Streaming

HTTP Dynamic Streaming (HDS) является технологией ведения потоковых трансляций, разработанной компанией *Adobe*. Финальная версия 3.0 официального стандарта опубликована в 2013 г., хотя существуют официальные упоминания об HDS с 2010 года. Для просмотра презентации *Adobe HDS* использует собственный проигрыватель *Flash Player*.

Спецификация *Adobe HDS* [5, С. 4] определяет фрагмент, как наименьшую адресуемую единицу медиа, позволяющую автономное декодирование. Фрагмент является базовой нумерованной единицей медиа, передаваемой по каналу связи. Это определение соответствует данному в разд. 1 определению *обобщенного фрагмента*. Основными кодеками являются H.264 и AAC. В одном фрагменте могут содержаться аудио и/или видеоданные [5, С. 28]. Смежные фрагменты объединяются в сегменты, которые хранятся на диске в виде файлов с расширением .f4v. Сегменты используются для повышения эффективности хранения и загрузки фрагментов. На сервере вся презентация выглядит как набор одного или нескольких сегментов.

Фрагменты и сегменты в *Adobe HDS* имеют формат F4V, представляющий собой расширение стандарта MPEG-4 Part 12. Для каждого сегмента на HTTP-сервере присутствует индексный файл .f4x, в котором содержится байтовое смещение каждого фрагмента в сегменте. Индексный файл представляет собой атом произвольного доступа (*random access*) afra. Индексный файл необходим серверу для поиска требуемого фрагмента в сегменте при выполнении запросов клиента. По ходу ведения презентации индексная информация постоянно обновляется.

Метаданные потока содержатся в паре манифестов — set-level и stream-level — файлов формата XML с расширением .f4m.

Манифест set-level необходим клиенту для информирования о существующих видеопотоках, а также для адаптивного воспроизведения. Манифест set-level содержит ссылки на манифесты stream-level для каждого описанного в нем потока. Манифест stream-level содержит:

- метаданные потока — тип кодека, разрешение экрана и т.п.;
- информацию bootstrap, включающую в себя: номера сформированных фрагментов, их метки времени и номера сегментов, в которых инкапсулированы эти фрагменты. Информация bootstrap представляет собой три атома: abst, asrt и afrt, сформированных на сервере и закодированных в формат Base64.

Ниже приведены примеры файлов-манифестов.

1. Манифест set-level:

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns = "http://ns.adobe.com/f4m/1.0" version = "3.0">
<id>myVideo</id>
<baseURL>http://www.example.com/myvideo/</baseURL>
<streamType>live</streamType>
<media href = "stream250.f4m" bitrate = "250" />
<media href = "stream500.f4m" bitrate = "500" />
</manifest>
```

В нем указано:

1) базовый URL сервера http://www.example.com/myvideo/;

2) параметр трансляции live, трансляция происходит в реальном времени;

3) ссылки на два манифеста stream-level с битрейтами 250 и 500.

2. Манифест stream-level для потока с битрейтом 250:

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns="http://ns.adobe.com/f4m/1.0"version= "3.0">
<id>myStream1</id>
<baseURL>http://www.example.com/data/</baseURL>
<bootstrapInfo profile="named" id="boot1" fragmentDuration ="4">
(BASE64 encoding of bootstrap information)
</bootstrapInfo>
<media url = "stream250" bootstrapInfoId = "boot1"/>
</manifest>
```

В нем присутствуют:

1) базовый URL сервера http://www.example.com/data/;

2) идентификатор потока myStream1 (MovieIdentifier);

3) медиа URL потока stream250 (QualitySegmentUrlModifier);

4) информация bootstrap.

По мере прохождения трансляции сервер создает фрагменты с медиаданными и добавляет их к сегменту. Число фрагментов в сегменте определяется параметрами длительностей фрагмента и сегмента, которые задаются в настройках сервера. Сервер создает информацию bootstrap, которая связывает между собой: а) время отображения видеокадров/аудиосэмплов во фрагменте; б) номера

фрагментов; в) номера сегментов. Вместе с метаданными потоков информацию bootstrap сервер помещает в манифест stream-level. При появлении новых фрагментов манифест stream-level и информация bootstrap в нем постоянно обновляются.

Для получения информации обо всех потоках клиент сначала единожды загружает с сервера set-level манифест, после чего запрашивает нужный stream-level манифест. Затем клиент расшифровывает информацию bootstrap и начинает запрашивать фрагменты выбранного потока в соответствии со следующим шаблоном:

http://<ServerBaseUrl>/<MovieIdentifier><QualitySegmentUrl Modifier>Seg<SegmentNumber>-Frag<FragmentNumber> (2)

Например, для того, чтобы получить фрагмент 210 сегмента 1, клиент должен сделать следующий запрос:

http://adobe.com/MyMovie/highSeg1-Frag210

HTTP-сервер, используя индексный файл .f4x, транслирует запрос в байтовую сдвижку запрашиваемого фрагмента (в примере это фрагмент 210) в сегменте (в примере это сегмент 1) и отправляет клиенту требуемый фрагмент. После загрузки всех доступных фрагментов, описанных в файле-манифесте, клиент загружает новый манифест stream-level, с bootstrap информацией о новых доступных фрагментах.

По умолчанию длительность фрагмента в технологии Adobe HDS составляет 4 с [15, С. 29], а рекомендованная длина буфера клиента — три фрагмента (утроенная длительность одного фрагмента) [5, С. 14]. С этими параметрами минимальная задержка живой трансляции будет отстоять от настоящего времени на 12 с.

Dynamic Adaptive Streaming over HTTP (MPEG-DASH)

MPEG-DASH является не готовой технологией, а стандартом [7], описывающим общие принципы организации ведения потовых трансляций над HTTP. Работа над созданием стандарта MPEG-DASH была начата в 2010 г., окончательный стандарт ISO/IEC 23009-1:2012 был опубликован в 2012 г. Существуют реализации стандарта MPEG-DASH в виде технологий, например *Azure Media Services* [16].

Данному в разд. 1 определению обобщенного фрагмента в терминологии MPEG-DASH соответствует сегмент — ассоциированная с URL единица медиаданных [7, С. 4].

Стандарт MPEG-DASH описывает два возможных формата фрагментов и вместе с тем позволяет расширять набор форматов сторонними разработчиками. Форматы предложенных фрагментов MPEG-DASH:

1) MPEG-2 TS (базируется на стандарте ISO/IEC 13818-1 [17]).

2) fMP4 (базируется на стандарте MPEG-4 Part 12).

Фрагменты содержат закодированные видеокадры и/или аудиосэмплы. Фрагменты могут содержать как видео и аудио вместе, так и только аудио или только видео. Хотя стандарт MPEG-DASH не предписывает использование определенных кодеков, наиболее распространенными кодеками в существующих реализациях являются H.264 и AAC.

Метаданные потока содержатся в файле-манифесте формата XML с расширением .mpd. Манифест содержит как общую информацию о потоках — битрейты дорожек, типы дорожек и т.д., — так и ссылки на фрагменты, временные метки и длительности сегментов.

Ссылки бывают трех видов:

1. Прямая ссылка на одиночный медиафайл презентации в файле-манифесте, например: <BaseURL>7657412348.mp4</BaseURL>

2. Список доступных фрагментов и их длительности, например:

```
<SegmentList duration = "10">
<SegmentURL media = "seg-m1-C2view-1.mp4"/>
<SegmentURL media = "seg-m1-C2view-2.mp4"/>
<SegmentURL media = "seg-m1-C2view-3.mp4"/>
</SegmentList>
```

3. План производства фрагментов, представляющий собой шаблон для построения ссылок, например:

http://cdn1.example.com/video/50000/$Time$.mp4v,
t = "0" d = "96000" r = "432",

где $Time$ время начала будущего файла фрагмента; t — время начала; d — длительность одного фрагмента; r — число фрагментов. Пример использования шаблона (время начала первого фрагменты ноль):

```
http://cdn1.example.com/video/500000/0.mp4v
http://cdn1.example.com/video/500000/180180.mp4v
http://cdn1.example.com/video/500000/360360.mp4v
```

Временная метка фрагмента есть время начала его воспроизведения относительно начала презентации. Она содержится в файле-манифесте.

Для воспроизведения трансляции клиент подключается к серверу, делая запрос на манифест. Манифест дает возможность клиенту получить или сгенерировать список ссылок на фрагменты.

Манифест изменяется в случае появления новых фрагментов или изменения местоположения (ссылок) прежних фрагментов. В параметре файла-манифеста "minimumUpdatePeriod" указан минимальный период потенциальных изменений манифеста. С помощью этого параметра можно оптимизировать процесс периодического обновления манифеста клиентом.

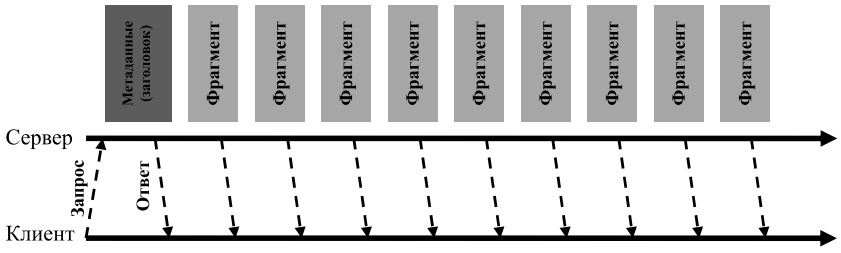

Рис. 3. Схема взаимодействия клиента и сервера, применяемая в технологии организации живой потоковой трансляции методом опережающей загрузки медиапотока поверх протокола HTTP

В качестве примера реализации стандарта MPEG-DASH рассмотрим технологию *Azure Media Service*. Для просмотра презентации клиент подключается к серверу *Azure* и загружает манифест этой трансляции. *Azure* использует адресацию фрагментов в реальном времени и задает время фрагментов в манифесте. Поэтому клиент всегда знает время последнего доступного фрагмента и может подключиться к трансляции при минимальной задержке трансляции от реального времени. *Azure* минимизирует эту задержку путем уменьшения длительности одного фрагмента до 2 с. Типичная задержка трансляции клиентом составляет от нескольких до 30 с [18].

Особенностью *Azure* являются то, что параметр манифеста "minimumUpdatePeriod" равен нулю, поэтому клиент без дополнительной информации никогда сам не обновляет манифест. Фрагменты могут содержать атом emsg с параметром "publishTime" указывающим время, в которое манифест должен быть обновлен.

Таким образом, клиент сначала один раз запрашивает манифест, а потом запрашивает только фрагменты. Если на сервере добавились новые фрагменты или изменились ссылки на старые фрагменты, сервер заранее сообщает клиенту во фрагменте, когда ему необходимо обновить манифест. Этим достигается оптимизация Интернет-трафика. Клиент загружает манифест только по необходимости.

3. Технология организации живой потоковой трансляции над протоколом HTTP методом опережающей загрузки медиапотока формата Ogg

Суть изложенной в п. 1 общей схемы организации трансляции медиа в реальном времени (см. рис. 1) заключается в том, что медиапоток сжимается (кодируется), инкапсулируется в небольшие фрагменты и пересылается клиенту. Клиент извлекает данные из фрагмента, декодирует кадры или сэмплы, склеивает их в непрерывный медиапоток и воспроизводит его. Во всех приведенных в п. 2 технологиях передача медиа от сервера к клиенту проводится только по инициативе последнего. Однако ничто не мешает использовать общую схему в случае, когда клиенту принадлежит только инициатива начала передачи медиа, после чего сервер начинает самостоятельную пересылку фрагментов клиенту. А это есть не что иное, как опережающая загрузка медиапотока с одновременным его воспроизведением (рис. 3).

Метод опережающей загрузки — это программная реализация возможности начала воспроизведения файла без необходимости ожидания завершения его полного скачивания. В рассматриваемом случае в роли загружаемого файла выступает презентация, т.е. медиапоток. Клиент начинает загрузку и воспроизведение презентации путем обращения к тегу <video>, в котором указывается URL презентации и атрибуты тега, которые передаются воспроизводящему презентацию проигрывателю браузера. При этом браузер клиента посылает однократно на сервер стандартный HTTP-запрос GET о подключении к трансляции. Получив запрос от клиента, сервер посылает метаданные, а затем начинает отсылать ему медиаданные презентации. Браузер клиента, в свою очередь, начинает фоновую загрузку презентации. После накопления в буфере браузера некоторого достаточного объема медиаданных презентации, начинается ее воспроизведение. Сравнивая рис. 2 и рис. 3, заметим, что метаданные отсылаются клиенту только однократно, а необходимости в периодической посылке манифестов нет, так как клиент в рассматриваемой схеме пассивен и ему не требуется знать URL загружаемых фрагментов.

Пересылаемые клиенту данные должны иметь фрагментированную структуру. После подключения клиента к серверу в середине презентации сервер попросту начнет транслировать медиа с очередного "свежего" фрагмента. Поскольку фрагмент снабжается меткой времени, воспроизведение клиентом презентации выполняется в "правильное" время, а нарушения синхронизации между видео и аудио при этом не будет. Для организации фрагментированного потока в настоя-

щей работе предлагается использование контейнера Ogg, который изначально разрабатывался как потоковый медиаконтейнер. Потоком в понимании Ogg является последовательность *пакетов*, которой присвоен уникальный идентификатор.

Пакет представляет собой данные, закодированные определенным кодеком. Например, для видеопотока пакетом может быть кадр, закодированный кодеком Theora [19]. Для аудиопотока пакетом может быть некоторое число аудиосемплов, закодированных кодеком Vorbis [20]. Размер пакета Ogg может быть любым, в том числе и нулевым.

Медиафайл формата Ogg может содержать один или несколько потоков, например два потока — видео и аудио. Каждый поток в контейнере Ogg содержится в виде упорядоченной последовательности объектов одинаковой структуры — *страниц*. Страница Ogg имеет заголовок и тело, содержащее собственно данные. Заголовок страницы содержит:

- идентификатор медиапотока, которому принадлежит страница;
- порядковый номер страницы в потоке;
- тип страницы (страница является началом потока, страница является концом потока, страница содержит незавершенный пакет);
- маркер позиции в своем потоке последнего завершенного пакета в странице (*granule position*).

Естественно принять содержащую медиаданные страницу Ogg за *обобщенный фрагмент* потоковой трансляции. Каждый фрагмент содержит только один тип медиаэлементов (например только аудио или только видео). Отличительной особенностью страниц Ogg является то, что максимальный их размер ограничен значением 65,307 байт [2, С. 8], в то время как размер видеокадра (и соответственно пакет, в который он закодирован) может быть больше этого значения. В этом случае один кадр будет содержаться в нескольких страницах Ogg, которые должны иметь один и тот же маркер позиции в своем потоке.

Маркер позиции сам по себе не является меткой времени. Но существует простой способ вычисления метки времени каждого кадра или сэмпла по известным маркеру позиции страницы и числу кадров (сэмплов) в секунду в потоке, схема вычисления зависит от кодека потока. Для кодека Theora маркер позиции является бинарной конструкцией из двух чисел-счетчиков ключевых (I) и разностных (P) кадров с начала трансляции медиапотока. Метка времени подсчитывается как число кадров от начала трансляции, деленное на число кадров в секунду. А в случае кодека Vorbis это просто число аудиосэмплов от начала трансляции, деленное на частоту дискретизации.

Метаданными потока являются пакеты, содержащие заголовки потока и инкапсулированные в несколько первых страниц Ogg. Они содержат общие данные о потоке, например разрешение экрана, число кадров (сэмплов) в секунду и т.д.

Для воспроизведения потоковой трансляции клиент подключается к серверу и делает однократный запрос, используя HTTP-метод GET, обращаясь ко всей презентации, как к файлу. В ответ клиент вначале получает страницы заголовка потока, а затем страницы с медиаданными, начиная с последней готовой страницы, сделанной сервером к моменту подключения клиента.

На рис. 4 изображена общая схема трансляции потока пакетов Ogg в реальном времени. В самом начале трансляции сервер создает первые три пакета (З), содержащие заголовки потока, и инкапсулирует их в страницы С1 и С2, которые хранит в памяти. При получении медиаданных от источника трансляции сервер кодирует их в пакеты (Д) и помещает в новые страницы, начиная с С3. В одной странице могут быть несколько пакетов, один пакет или часть пакета. На рис. 4 изображен случай, когда клиент подключается к серверу в момент, когда тот создал страницу $n-1$. После подключения клиенту сначала передаются страницы С1 и С2 с заголовками потока, а затем страницы с медиаданными, начиная с номера $n-1$.

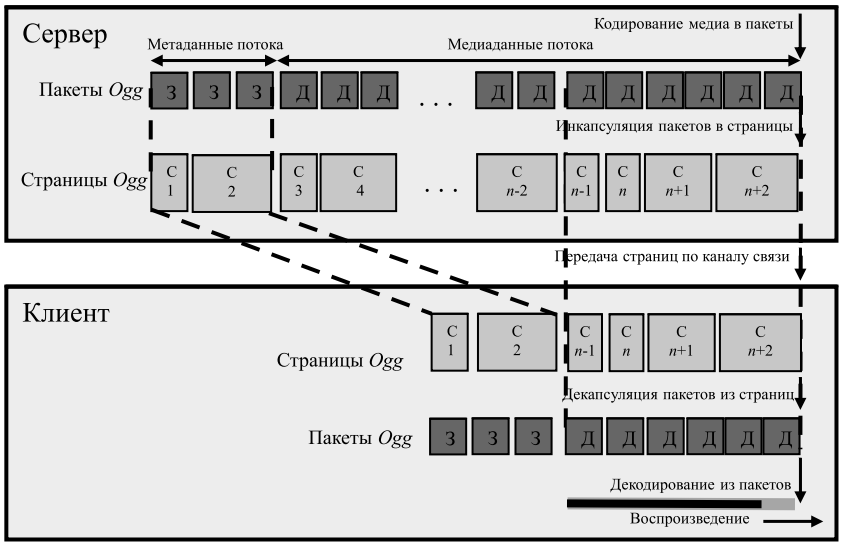

Рис. 4. Общая схема живой трансляции потока пакетов Ogg

Заключение

В настоящей работе предлагается использование контейнера Ogg для организации потоковой трансляции в реальном времени над протоколом

HTTP методом опережающей загрузки. Несомненным плюсом использования формата медиаконтейнера Ogg, кодеков Theora и Vorbis является то, что они обладают открытой лицензией и напрямую поддерживаются стандартом HTML5. Медиапотоки, закодированные кодеками Theora и Vorbis и помещенные в медиаконтейнер Ogg, доступны для воспроизведения во многих браузерах. Это позволяет клиенту отказаться от использования сторонних плагинов и расширений к браузерам.

В работе проведено сравнение предложенной в работе [1] технологии организации живой потоковой трансляции над протоколом HTTP методом опережающей загрузки медиапотока формата Ogg с лидирующими технологиями потоковых трансляций [4—7]. Для этой цели предложена общая схема, отражающая следующие важнейшие особенности потоковых трансляций:

- передача потока от сервера клиенту осуществляется фрагментами;
- трансляция имеет метаданные, предшествующие медиаданным потока;
- каждый фрагмент снабжен меткой времени.

Выявлено, что *Apple HLS*, *Adobe HDS*, *Microsoft Smooth Streaming*, *MPEG DASH* являются реализациями общей схемы и имеют следующие особенности:

- обобщенным фрагментом является имеющая URL единица медиа; инициатором начала трансляции и загрузки фрагментов является клиент, для чего ему необходимо предварительно получить от сервера манифест и затем периодически обновлять его;
- метаданные содержатся в файле-манифесте;
- метка времени фрагмента передается в манифесте.

Показано, что технология организации живой потоковой трансляции над протоколом HTTP методом опережающей загрузки медиапотока формата Ogg также удовлетворяет общей схеме и обладает следующими особенностями:

- фрагментом является страница Ogg с медиаданными; клиент пассивно принимает фрагменты от сервера;
- метаданные содержатся в заголовочных страницах Ogg и передаются клиенту только один раз после инициирования клиентом начала трансляции медиапотока;
- метка времени — это маркер позиции фрагмента (*granule position*).

Тот факт, что оба рассмотренных подхода ([4—7] и [1]) являются реализациями общей схемы, позволяет сделать вывод об их логической эквивалентности.

Заметим, что при использовании технологий потоковых трансляций клиент для плавного воспроизведения презентации должен иметь в буфере несколько фрагментов. Поскольку для технологий потоковой трансляции [4—7] размер фрагмента, как правило, составляет несколько секунд, суммарная задержка отображения презентации от реального времени в таком подходе будет весьма значительной, например, для *Azure* она может быть до 30 с. В предложенной в работе [1] технологии потоковой трансляции размер видеофрагмента составляет один кадр. С учетом 25 кадров в секунду и 4—5 фрагментов в буфере ожидаемая задержка составляет несколько сотен миллисекунд.

На базе технологии, описанной в работе [1], был разработан медиасервер [21] для организации потоковых трансляций между технологическими подсистемами Ускорительного комплекса У-70. В качестве источников трансляции были использованы IP-камеры D-Link DCS-3430. Проведенные эксперименты по выявлению задержек воспроизведения презентаций от реального времени дали результат 250...500 мс. Это дает возможность использования предложенной в работе [1] технологии не только для односторонних трансляций, но и для организации общения клиентов в реальном времени между собой.

Список литературы

1. **Лобов И. В., Готман В. Г.** Трансляция мультимедиа в реальном времени над протоколом HTTP методом опережающей загрузки // Технологии и средства связи. 2016. № 5. С. 36—40.

2. **Pfeiffer S.** The Ogg Encapsulation Format Version 0. Request for Comments: 3533. 2003. 15 p.

3. **Fielding R., Gettys J.** Hypertext Transfer Protocol — HTTP/1.1. Request for Comments: 2616. 1999. 114 p.

4. **Pantos R., May W.** HTTP Live Streaming. draft-pantos-http-live-streaming-18. Apple Inc. 2015. 49 p.

5. **HTTP Dynamic Streaming Specification Version 3.0 FINAL.** Adobe Systems Incorporated. 2013. 31 p.

6. **Smooth Streaming Protocol.** [MS-SSTR] — v20160714. Microsoft Corporation. 2016. 64 p.

7. **ISO/IEC 23009-1.** Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats. Second edition. 2014. 144 p.

8. **HTML Living Standard.** The Web Hypertext Application Technology Working Group. 2017. 1177 p.

9. **ITU-T H.222.0 (02/00).** Information technology — Generic coding of moving pictures and associated audio information: systems. 2000. 153 p.

10. **Wiegand T., Sullivan G., Luthra A.** Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC). 2002. 253 p.

11. **ISO/IEC 13818-7:1997.** Information technology — Generic coding of moving pictures and associated audio information — Part 7: Advanced Audio Coding (AAC). 1997.

12. **Bocharov J., Burns Q., Folta F., Hughes K., Murching A., Olson L., Schnell P., Simmons J.** Portable encoding of audio-video objects The Protected Interoperable File Format (PIFF). Microsoft Corporation. 2010. 28 p.

13. **ISO/IEC 14496-12:2015.** Information technology — Coding of audio-visual objects — Part 12: ISO base media file format. Edition: 5. 2015. 233 p.

14. **Extensible Markup Language (XML) 1.0.** W3C. Edition: 5. 2008. 32 p.

15. **Adobe HTTP Dynamic Streaming, User Guide.** Concentric Cloud Solutions An Xo Company. 2012. 37 p.

16. **Azure Media Services overview and common scenarios.** Microsoft Corporation 2017. 986 p.

17. **ISO/IEC 13818-1.** Information technology — Generic coding of moving pictures and associated audio information: Systems. Edition: 2. 2000. 154 p.

18. **Hughes K.** DASH Live Streaming with Azure Media Service. Azure Media Services. 2014.

19. **Theora Specification.** Xiph.Org Foundation. 2011. 196 p.

20. **Vorbis I specification.** Xiph.Org Foundation. 2015. 74 p.

21. **Lobov I., Gotman V.** Media server for video and audio exchange between the U-70 accelerator complex control rooms // Proceedings of RuPAC2014. RUPAC. Obninsk. 2014. P. 368—388.

I. V. Lobov, Senior Researcher, e-mail: lobov@ihep.ru

V. G. Gotman, Junior Researcher, e-mail: vladislav.gotman@ihep.ru

Institute for High Energy Physics, National Research Center "Kurchatov Institute", MO, Protvino

The Utilization of Ogg Multimedia Container Format for Live Streaming over HTTP Using Progressive Download Method

The general scheme of the media live streaming over HTTP has been proposed. It was shown that the general scheme has three common basic features: generalized fragments, metadata and fragment timestamp. The generalized fragment is the media container unit being transmitted over network. The fragment should contain encoded part of the media stream. The metadata stores common information (codec type, frame rate, image dimensions, bit rate, sample rate) which necessary for the client to decode and play the media stream back. The timestamp is the fragment property necessary to correct playback and synchronize for several media tracks — particularly for video and audio streams. The analysis of existing basic technologies (Apple HLS, Adobe HDS, Microsoft Smooth Streaming, MPEG DASH) of live streaming over HTTP was made. It was revealed that all these technologies meet this scheme. All of them have their own concept for the small unit of the coded media stream which matches to the proposed generalized fragment with one significant addition — the fragment has independent download possibility. The metadata and timestamps are stored into the manifest file. The technology of the media container Ogg utilization for live streaming over HTTP by progressive download method was proposed. This technology is quite different from the existing basics technologies. Yet it meets the general scheme. The comparison of the proposed technology with existing basic technologies has been made.

Keywords: *live streaming, progressive download, Ogg format, Apple HLS, Adobe HDS, Microsoft Smooth Streaming, MPEG DASH*

References

1. **Lobov I. V., Gotman V. G.** Translyatsiya multimedia v realnom vremeni nad protokolom HTTP metodom operezhayuschey zagruzki (Real time media streaming over HTTP using progressive download method). *Tehnologii i sredstva svyazi*. 2016. no 5. pp. 36—40.

2. **Pfeiffer S.** The Ogg Encapsulation Format Version 0. Request for Comments: 3533. 2003. 15 p.

3. **Fielding R., Gettys J.** Hypertext Transfer Protocol — HTTP/1.1. Request for Comments: 2616. 1999. 114 p.

4. **Pantos R., May W.** HTTP Live Streaming. draft-pantos-http-live-streaming-18. Apple Inc. 2015. 49 p.

5. **HTTP** Dynamic Streaming Specification Version 3.0 FINAL. Adobe Systems Incorporated. 2013. 31 p.

6. **Smooth** Streaming Protocol. [MS-SSTR] — v20160714. Microsoft Corporation. 2016. 64 p.

7. **ISO/IEC 23009-1**. Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats. Second edition. 2014. 144 p.

8. **HTML** Living Standard. The Web Hypertext Application Technology Working Group. 2017. 1177 p.

9. **ITU-T H.222.0 (02/00).** Information technology — Generic coding of moving pictures and associated audio information: systems. 2000. 153 p.

10. **Wiegand T., Sullivan G., Luthra A.** Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC). 2002. 253 p.

11. **ISO/IEC 13818-7: 1997.** Information technology — Generic coding of moving pictures and associated audio information — Part 7: Advanced Audio Coding (AAC). 1997.

12. **Bocharov J., Burns Q., Folta F., Hughes K., Murching A., Olson L., Schnell P., Simmons J.** Portable encoding of audio-video objects The Protected Interoperable File Format (PIFF). Microsoft Corporation. 2010. 28 p.

13. **ISO/IEC 14496-12: 2015.** Information technology — Coding of audio-visual objects — Part 12: ISO base media file format. Edition: 5. 2015. 233 p.

14. **Extensible Markup Language (XML) 1.0.** W3C. Edition: 5. 2008. 32 p.

15. **Adobe HTTP Dynamic Streaming, User Guide.** Concentric Cloud Solutions An Xo Company. 2012. 37 p.

16. **Azure Media Services overview and common scenarios.** Microsoft Corporation 2017. 986 p.

17. **ISO/IEC 13818-1.** Information technology — Generic coding of moving pictures and associated audio information: Systems. Edition: 2. 2000. 154 p.

18. **Hughes K.** DASH Live Streaming with Azure Media Service. Azure Media Services. 2014.

19. **Theora Specification.** Xiph.Org Foundation. 2011. 196 p.

20. **Vorbis I specification.** Xiph.Org Foundation. 2015. 74 p.

21. **Lobov I., Gotman V.** Media server for video and audio exchange between the U-70 accelerator complex control rooms. *Proceedings of RuPAC2014. RUPAC. Obninsk*, 2014, pp. 368—388.