Трансляция мультимедиа в реальном времени над протоколом HTTP методом опережающей загрузки

Real time media streaming over HTTP using progressive download method

В работе обсуждается идея применения метода опережающей загрузки для организации трансляции медиапотока в реальном времени над протоколом HTTP. Клиент производит непрерывную загрузку и воспроизведение потока как единого медиафайла, создаваемого источником трансляции по мере ее прохождения. Обсуждена возможная реализация прототипа системы трансляции медиа в реальном времени на основе использования метода опережающей загрузки.

An idea of progressive download adaptation for real time media streaming over HTTP was discussed. The client is to download a continuous stream as a single media file created while the translation passes. On the basis of this method a possible implementation of real time media streaming has been proposed.

Игорь
Лобов

Старший научный сотрудник Института физики высоких энергий, НИЦ "Курчатовский институт", к.ф.-м. н.

Igor
LobovSenior researcher of the Institute for High Energy Physics, National Research Center "Kurchatov Institute"
PhD. in Physical and Mathematical Sciences

Владислав
Готман

Младший научный сотрудник Института физики высоких энергий, НИЦ "Курчатовский институт"

Vladislav
GotmanJunior researcher of the Institute for High Energy Physics, National Research Center "Kurchatov Institute"

Ключевые слова:

медиасервер, трансляция, поток, опережающая загрузка, формат Ogg

Keywords:

media server, streaming, progressive download, Ogg format

Введение

В последнее время наблюдается бурное развитие технологий потокового вещания в реальном времени над протоколом HTTP [1]. В качестве массового и удобного инструмента просмотра трансляций широко используются Web-браузеры. Существует несколько ведущих технологий организации живого вещания, основанных на протоколе HTTP. Наиболее популярными из них являются Apple HTTP Live Streaming (HLS) [2], Adobe HTTP Dynamic Streaming (HDS) [3], Microsoft Smooth Streaming (SS) [4], MPEG-DASH [5]. Отличительной особенностью этих технологий является то, что клиент получает трансляцию путем загрузки с сервера небольших фрагментов длительностью несколько секунд. Для того чтобы клиент смог воспроизводить фрагменты в нужное время и в правильной последовательности, сервер подготавливает для клиента специальный файл – плей-лист ("манифест"), в котором содержится схема правильного воспроизведения фрагментов, а также содержатся параметры, необходимые для начала воспроизведения. По мере прохождения трансляции клиент должен периодически обновлять содержимое плей-листа и производить загрузку фрагментов и их воспроизведение в соответствии с тем, как это предписано в манифесте.

Возможны разные схемы организации фрагментов на сервере. В одном случае фрагменты создаются по ходу трансляции в виде небольших файлов и прямо выкладываются на HTTP-сервере. В другом случае сервер в ответ на запросы клиента создает фрагменты "на лету" и затем отсылает их.

В качестве формата фрагментов все перечисленные выше ведущие технологии в той или иной мере используют стандарты MPEG-2 TS [6] или MPEG-4 Part 12 [7], видеопоток обычно сжимается кодеком H.264 [8], аудиопоток сжимается кодеком AAC [9].

Воспроизведение трансляции в браузере клиента для каждой из приведенных выше четырех технологий не может производиться с помощью "чистого" тега <video>, а требует плагинов или надстроек:

- Apple HLS использует Media Source Extensions [10];
- Adobe HDS использует Flash Player [11];
- Microsoft SS использует плагин Sil-verlight [12];
- MPEG-DASH использует Media Source Extensions.

В настоящей работе предлагается более простая схема организации живого вещания над HTTP, в которой на стороне клиента применяется тег <video> в "чистом" виде, т.е. без использования дополнительных расширений, таких как Media Source Extensions, Flash Player и т.п. Суть предлагаемой технологии заключается в том, что сервер формирует "на лету" медиапоток в виде единого потокового медиафайла и отсылает его клиенту по мере

создания каждого очередного фрагмента этого файла. Клиент производит прием и воспроизведение этого файла стандартным методом опережающей загрузки (Progressive Download).

Применительно к медиафайлу определительное слово "потоковый" в данном случае является решающим. Как показано ниже, это дает возможность подключения клиента в произвольный момент к уже ведущейся трансляции. В качестве формата потокового медиафайла предлагается контейнер Ogg [13] с видеокодеком Theora [14] и аудиокодеком Vorbis [15]. Помимо того, что формат Ogg является форматом медиапотока, форматы Ogg, Theora и Vorbis не имеют каких-либо лицензионных или патентных ограничений. На стороне клиента эти форматы напрямую поддерживается тегом <video> языка HTML5 [16].

Описание метода опережающей загрузки для воспроизведения медиафайла

Метод опережающей загрузки – это программная реализация возможности начала воспроизведения медиафайла без необходимости ожидания завершения его полного скачивания. Метод опережающей загрузки применяется повсеместно для организации просмотра клиентом медиафайлов, расположенных на сервере. В современных форматах медиафайлов вся информация, необходимая клиенту для того, чтобы начать воспроизведение (размер кадра в пикселях, скорость воспроизведения и т.п.), содержится в самом начале файла – в его заголовке. Поэтому после того, как клиент получил заголовок медиафайла и его первый фрагмент, способный к воспроизведению, клиент уже может начать воспроизведение медиафайла еще до завершения полной загрузки. Параллельно с воспроизведением медиафайла клиент производит его дозагрузку. Если при этом скорость загрузки будет не меньше скорости воспроизведения, то медиафайл будет воспроизводиться гладко, без рывков. Далее процесс воспроизведения медиафайла методом опережающей загрузки рассмотрен более подробно.

Клиент производит скачивание и воспроизведение медиафайла путем обращения к тегу <video>, в котором указывается URL медиафайла и атрибуты тега, которые передаются воспроизводящему медиафайл плееру. Выполнение тега <video> происходит следующим образом: браузер клиента делает запрос GET, который обязательно включает в себя стартовую строку с возможными параметрами выполнения запроса и заголовками, сервер отсылает клиенту ответ "HTTP/1.1 200 OK", а также может в зависимости от предпочтений клиента изменять для него технические параметры трансляции. После этого браузер клиента начинает фоновую загрузку медиафайла с его одновременным воспроизведением.

В дальнейшем будем представлять загружаемый файл в виде расположенных последовательно один за другим фрагментов. Под фрагментом будем понимать единичный элемент медиафайла, который может быть воспроизведен без знания других частей файла (кроме заголовка файла). Отсюда следует, что фрагмент должен содержать целое число видеокадров и целое число аудиосемплов. Минимальным является фрагмент с одним видеокадром (и с соответствующим числом аудиосемплов).

Загрузка медиафайла состоит в передаче с сервера на клиент всего содержимого файла, которая производится небольшими порциями. Для возможности воспроизведения медиафайла клиентом сразу после загрузки первого фрагмента клиент вначале должен получить заголовок медиафайла, в котором содержатся параметры, необходимые для

начала воспроизведения (частота кадров, частота семплирования, разрешение экрана и т.п.).

Получив заголовок, клиент сможет начать воспроизведение медиафайла сразу после того, как в буфере окажется первый фрагмент файла. Если каждый раз к моменту завершения воспроизведения клиентом очередного фрагмента в буфере оказывается следующий за ним фрагмент, клиент будет воспроизводить медиафайл плавно. Иными словами, в каждый момент времени в буфере должно содержаться достаточное количество фрагментов для обеспечения непрерывности воспроизведения. Временная длительность фрагмента задает минимальную задержку между временем началом загрузки медиафайла и временем начала его воспроизведения.

На рис. 1 представлена иллюстрация процесса проигрывания медиафайла с опережающей загрузкой. Для упрощения картины изображен гипотетический случай, когда фрагмент содержит три видеокадра, а на один видеокадр приходится два аудиосемпла (в реальности на один видеокадр приходится гораздо больше аудиосемплов). На рис. 1 изображена ситуация, когда клиент уже загрузил с сервера фрагменты медиафайла с первого по n-1. В текущий момент клиент производит загрузку и последующую декапсуляцию фрагмента n. Одновременно с этим клиент воспроизводит медиа, содержащееся во фрагменте n-2.

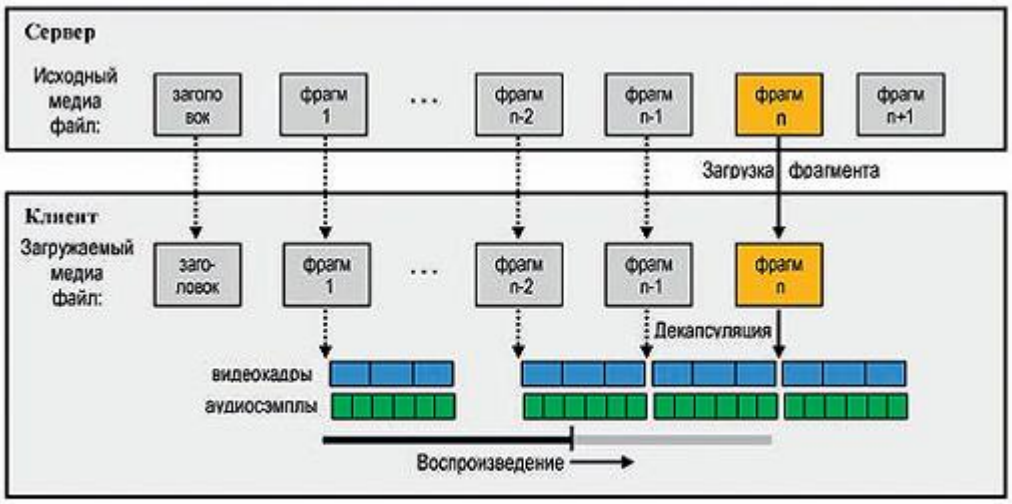

Рис. 1. Схема воспроизведения удаленного медиафайла методом опережающей загрузки

В нижней части рис. 1 изображен индикатор выполнения (Progress Bar), наглядно отображающий процесс загрузки и воспроизведения медиафайла. Вся горизонтальная линия индикатора выполнения схематично изображает временную длительность воспроизведения всех загруженных на текущий момент фрагментов медиафайла, тогда как левый отрезок этой линии черного цвета отображает временную длительность уже воспроизведенной части медиафайла. Короткий вертикальный отрезок показывает момент времени текущего воспроизведения, когда воспроизводятся видеокадры и аудиосемплы, содержащиеся во фрагменте n-2.

Применение метода опережающей загрузки для организации трансляции медиапотока в реальном времени

Изложенная в предыдущей части статьи схема широко используется для воспроизведения уже полностью сформированного медиафайла, однако ничто не мешает ее применить для живой трансляции презентации, если передаваемый клиенту медиапоток рассматривать как файл, создаваемый сервером по ходу трансляции "на лету". С точки зрения клиента будет происходить обычное воспроизведение медиафайла с опережающей загрузкой. Сервер производит имитацию отправки медиафайла клиенту, хотя никакого файла не существует, а каждый раз создается очередной фрагмент, который тут же отсылается клиенту. Сервер ведет запись живого вещания в виртуальный медиафайл. Этот файл назван виртуальным, поскольку он создается в оперативной памяти сервера, а не на диске. В реальности в оперативной памяти находится только буфер, в котором хранятся два фрагмента – предыдущий для отсылки клиенту и текущий для его формирования. По мере приема видеокадров и аудиосемплов сервер формирует текущий фрагмент. Сразу после записи в буфер только что сформированного текущего фрагмента сервер делает его предыдущим и начинает отправлять клиенту.

На рис. 2 приведена иллюстрация изложенной схемы организации живой трансляции с применением опережающей загрузки.

На рис. 2 изображена ситуация, когда сервер закончил прием очередного элемента медиапотока (трех видеокадров и шести аудиосемплов), необходимого для формирования фрагмента медиафайла, сформировал фрагмент n и отослал его клиенту. Одновременно с этим сервер принимает от видеокамеры следующие видеокадры и аудиосемплы, он уже принял два видеокадра и четыре аудиосемпла. Клиент загрузил фрагмент n и восстановил из него первоначальный элемент медиапотока (три видеокадра и шесть аудиосемплов). Одновременно с этим клиент воспроизводит медиа, содержавшееся в ранее принятом фрагменте n-1.

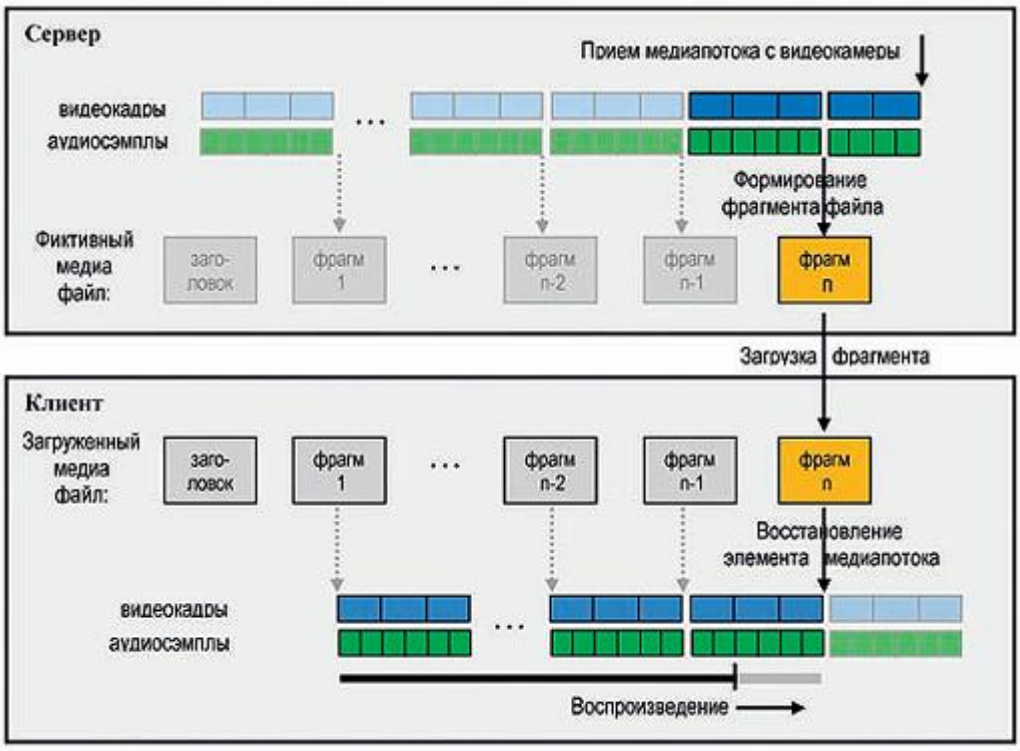

Рис. 2. Схема организации живой трансляции методом опережающей загрузки

На рис. 1 и 2 предполагается, что процессы преобразования "элемент медиапотока → фрагмент" и "фрагмент → элемент медиапотока", а также передачи фрагмента от сервера

к клиенту происходят мгновенно. Из рис. 2 видно, что если фрагмент содержит один кадр, то при частоте в N кадров в секунду минимальная задержка между временем приема медиа с видеокамеры и временем отображения медиа у клиента будет составлять 1/N секунд, т.е. менее одной секунды.

Из изложенной схемы также следует, что клиент может подключиться к трансляции презентации в любой момент времени. Вначале он получит заголовок файла, а затем начнет принимать фрагменты, начиная с момента подключения к трансляции. Таким образом, имеется возможность организовать параллельную трансляцию презентации одновременно нескольким клиентам, подключающимся к трансляции по ходу ее проведения. На рис. 3 показана схема подключения двух клиентов к трансляции презентации в разные моменты времени.

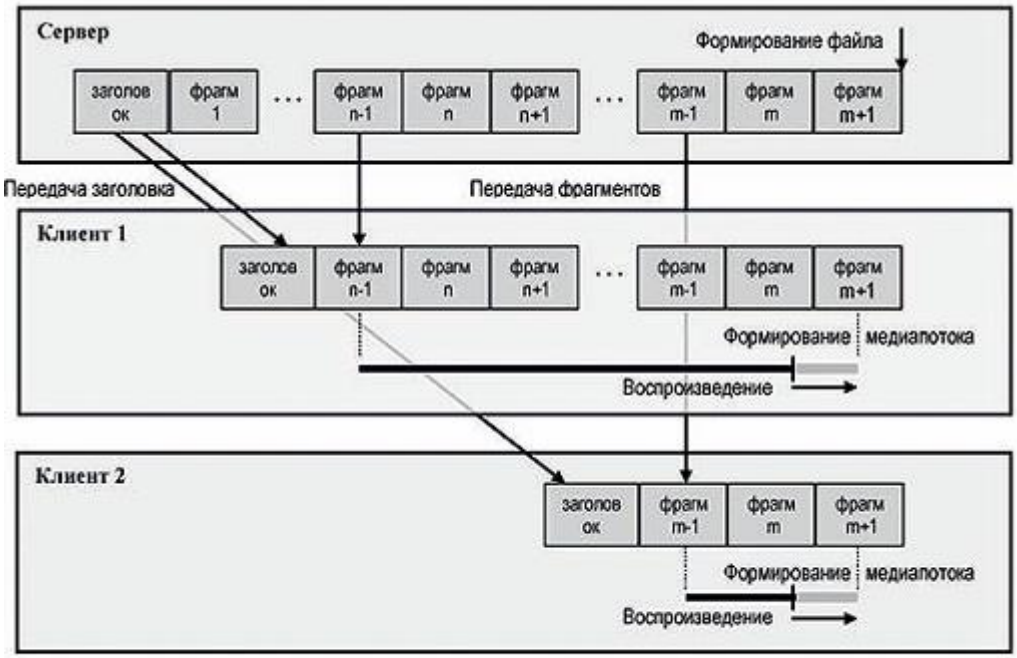

Рис. 3. Схема одновременного параллельного приема потоков двумя клиентами, подключающимися к трансляции в разные моменты времени

Первый клиент подключился к серверу в тот момент, когда сервер только что создал фрагмент n-1. Сервер вначале передает ему фрагмент с заголовком потока, а затем начинает передавать медиаданные, начиная с фрагмента n-1. Второй клиент подключился к серверу сразу после создания фрагмента m-1. Сервер вначале передает ему фрагмент с заголовком потока, а затем медиаданные, начиная с фрагмента m-1.

Возможная реализация системы живой трансляции медиа над HTTP методом опережающей загрузки

Резюмируя все сказанное выше, можно выделить следующие свойства медиафайла, делающие возможным его использование в качестве формата потока живой трансляции методом опережающей загрузки:

- наличие заголовка с описанием свойств, необходимых клиенту для декодирования и воспроизведения фрагментов;
- разбивка файла на фрагменты, позволяющие их воспроизведение независимо друг от друга;

- наличие у каждого фрагмента метки времени начала воспроизведения этого фрагмента относительно первого фрагмента.

Всем этим свойствам удовлетворяет формат Ogg, который и будет подразумеваться в дальнейшем. Контейнер Ogg может содержать несколько потоков, в нашем случае их два – поток видео и поток аудио. Поток формата Ogg состоит из так называемых страниц, которые играют роль фрагментов. Каждая страница, помимо медиаданных, содержит служебную информацию о принадлежности этой страницы к определенному потоку, ее порядковый номер в потоке, информацию о моменте времени, с которого нужно воспроизводить медиаданные этой страницы. Первые две страницы потока под номерами 0 и 1 являются заголовком потока Ogg и содержат служебную информацию обо всем потоке – разрешение экрана, битрейт и т.д.

Рассмотрим возможную схему организации живой трансляции с использованием контейнера Ogg. Для подключения к трансляции клиент использует тег <video>. При выполнении этого тега браузер делает запрос GET на адрес сервера:

GET / HTTP/1.1
Host: IP-address:port
Connection: keep-alive

Получив запрос от клиента, сервер в ответ должен отослать следующую строку:

HTTP/1.1 200 OK
Connection: keep-alive
Cache-control: no-cache, no-store
Content-type: video/ogg

Затем сервер создает заголовок контейнера Ogg и отсылает его клиенту. Клиент воспринимает прием заголовка как сигнал начала воспроизведения. На основании данных принятого заголовка он инициализирует видеоплеер и переходит к приему, декодированию и воспроизведению страниц Ogg.

В процессе загрузки и проигрывания видеопотока браузер клиента может сохранять загружаемый файл в своем буфере (кеше). В этом случае достаточно длинная видеотрансляция может привести к неприятным последствиям нехватки дискового пространства. К счастью, сервер может контролировать буферизацию. При подключении к серверу браузер клиента в ответ на запрос GET получает информацию – как именно надо производить буферизацию медиафайла:

- буферизация только метаданных;
- буферизация медиаданных и метаданных;
- без буферизации.

Лучшей стратегией воспроизведения трансляции методом опережающей загрузки является выбор опции "без буферизации".

Предлагается следующая реализация прототипа системы трансляции медиа в реальном времени, использующая метод опережающей загрузки. Прототип состоит из следующих компонентов (см. рис. 4):

1. Потокового сервера, функционирующего на выделенном компьютере. В его функции входит:
    - прием данных от источников трансляции (модуль "Приемник");
    - перекодирование принимаемых данных "на лету" в поток Ogg (модуль "Перекодировщик");
    - отправка потока Ogg клиентам (модуль "Коммуникатор").
2. HTTP-сервера, содержащего необходимый набор HTML-страниц для клиента.
3. Базы данных SQL и программы управления этой базой данных, в задачу которых входит организация согласованной работы всех компонентов системы трансляции.

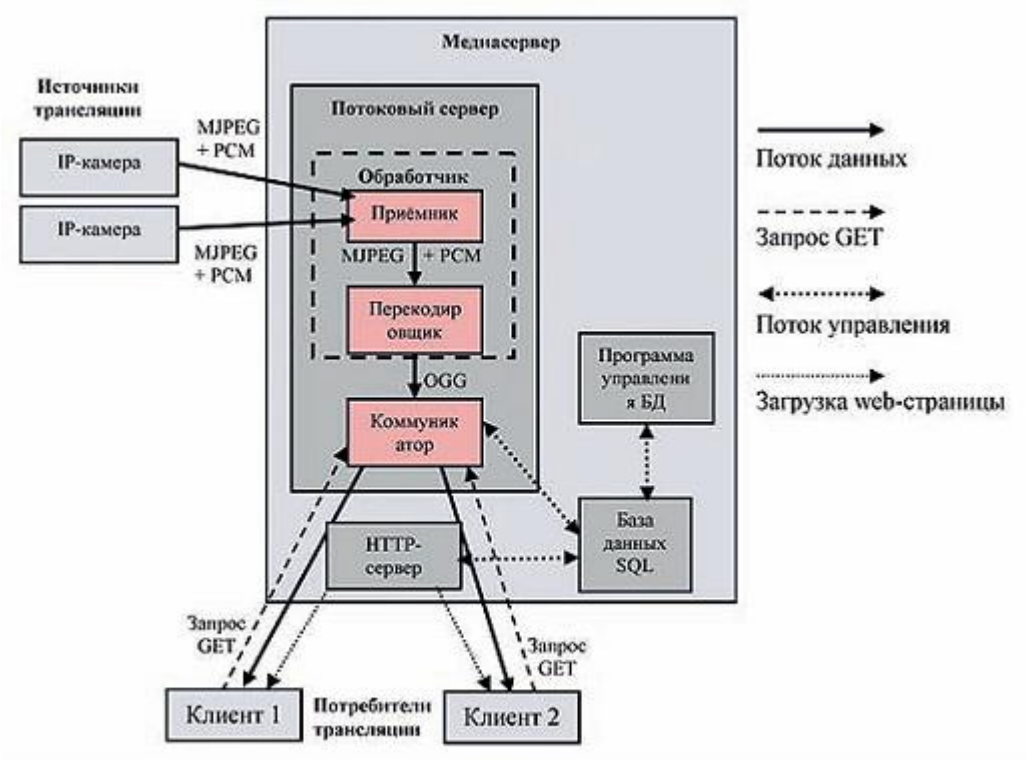

Рис. 4. Схема прототипа системы трансляций

В качестве источника трансляции видео и аудио предполагается использовать IP-камеры, передающие на потоковый сервер данные видео в формате Motion JPEG [17] и аудио в формате PCM [18], инкапсулированные в пакеты формата RTP [19]. После приема данных с камер потоковый сервер должен перекодировать их в формат Ogg.

При подключении к одной трансляции сразу нескольких клиентов параметры выполнения запроса GET различных клиентов могут отличаться. Коммуникатор должен уметь подстраивать отправляемый клиенту поток медиаданных формата Ogg под его требования.

В процессе трансляции клиент постоянно загружает фрагменты медиафайла. Время, за которое клиент загружает фрагмент, должно быть меньше времени его воспроизведения. В противном случае воспроизведение трансляции будет сопровождаться рывками – остановками воспроизведения для дозагрузок. Время загрузки фрагмента зависит от пропускной способности канала связи и качества изображения и звука в медиафайле. Чем лучше качество изображения, тем больше размер каждого кадра и, следовательно, всего медиафайла. Обеспечить плавность воспроизведения трансляции можно двумя способами – либо увеличить пропускную способность канала связи, либо ухудшить качество изображения и звука, уменьшив тем самым размер загружаемых фрагментов.

Заключение

Использование метода опережающей загрузки медиапотока, рассматриваемого как файл, позволяет организовать трансляцию медиа в реальном времени. В качестве контейнера медиапотока предлагается использовать формат Ogg, который по своей сути является потоковым и позволяет нескольким клиентам подключаться к уже ведущейся трансляции в произвольный момент времени.

В сравнении с основными технологиями потокового вещания (Apple HLS, HDS, Microsoft SS, MPEG-DASH), предлагаемая реализация системы трансляции методом опережающей загрузки обладает определенными достоинствами. Серверу нет необходимости создавать манифесты и производить предварительную разбивку медиафайлов трансляции на фрагменты. Медиафайл живой трансляции формата Ogg передается как есть. Для воспроизведения трансляции на стороне клиента используется тег <video> без привлечения сторонних плагинов и дополнительных расширений браузера. Отсутствие ограничения на размер фрагмента страницы Ogg позволяет уменьшить задержку вещания до приемлемой величины. Ожидаемая задержка времени воспроизведения трансляции на клиенте составляет менее одной секунды. Отсутствие патентных или лицензионных ограничений на формат Ogg позволяет использовать предлагаемую технологию в узких сегментах (на предприятии, производстве и т.д.)n.

Литература

1. Hypertext Transfer Protocol – HTTP/1.1, Request for Comments: 2616, Network Working Group, 06.1999.
2. HTTP Live Streaming draft-pantos-http-live-streaming-18, R. Pantos, Ed., W. May, Apple Inc, 19.11.2015.
3. HTTP Dynamic Streaming Specification Version 3.0 FINAL, Adobe Systems Incorporated.
4. [MS-SSTR] – v20150630, Smooth Streaming Protocol, Microsoft Corporation, 30.06.2015.
5. INTERNATIONAL STANDARD ISO/IEC 23009-1, Second edition 2014-05-15, Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats.
6. H.222.0 TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS (10/14) Infrastructure of audiovisual services – Transmission multiplexing and synchronization.
7. INTERNATIONAL STANDARD ISO/IEC 14496-12, Second edition 2005-04-01, Information technology – Coding of audio-visual objects – Part 12: ISO base media file format.
8. Draft ITU – T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC).
9. ISO/IEC 13818-7:1997 Information technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced Audio Coding (AAC).
10. Media Source Extensions, W3C Candidate Recommendation, 03 May 2016, Matthew Wolenetz, Google Inc., Jerry Smith, Microsoft Corporation.
11. Adobe Flash Player. [online]. Доступ через: http://www.adobe.com/ru/products/flashplayer.html.
12. Get Silverlight 5 [online]. Доступ через: https://www.microsoft.com/silverlight/.
13. The Ogg Encapsulation Format Version 0, Request for Comments: 3533, Network Working Group S. Pfeiffer, 05.2003.
14. Theora Specification, Xiph.Org Foundation, 03.16.2011.
15. Vorbis I specification, Xiph.Org Foundation, 27.02.2015.
16. HTML Living Standard, 4.02.2016.
17. RTP Payload Format for JPEG-compressed Video, Request for Comments: 2435, Network Working Group, 10.1998.

18. The Audio/L16 MIME content type, Request for Comments: 2586, Network Working Group, 05.1999.
19. RTP: A Transport Protocol for Real-Time Applications, Request for Comments: 3550, Network Working Group, 07.2003.

Опубликовано: Журнал "Технологии и средства связи" #5, 2016