ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 67 за 2011 г.

Бухштаб Ю.А., Воробьев А.А., Евтеева Н.Н.

Методы виртуального редактирования в распределенной среде видео и аудио потоков, представленных в различных форматах

Рекомендуемая форма библиографической ссылки: Бухштаб Ю.А., Воробьев А.А., Евтеева Н.Н. Методы виртуального редактирования в распределенной среде видео и аудио потоков, представленных в различных форматах // Препринты ИПМ им. М.В.Келдыша. 2011. № 67. 12 с. URL: http://library.keldysh.ru/preprint.asp?id=2011-67

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Российской Академии наук

Ю.А. Бухштаб, А.А. Воробьев, Н.Н. Евтеева

Методы виртуального редактирования в распределенной среде видео и аудио потоков, представленных в различных форматах

Москва - 2011

Ю.А. Бухштаб, А.А. Воробьев, Н.Н. Евтеева. Методы виртуального редактирования в распределенной среде видео и аудио потоков, представленных в различных форматах. Препринт Института прикладной математики им. М.В. Келдыша РАН, 2011, 12 страниц, библиография: 4 наименования.

В работе рассматриваются вопросы организации программных средств, обеспечивающих возможность создания и использования в средах разных браузеров единого виртуального видеоконтента из потоков, представленных в различных форматах и транслируемых с распределенных серверов.

Y.A. Bukhstab, A.A. Vorobjov, N.N. Evteeva. Methods of virtual editing in a distributed environment of video and audio streams, presented in various formats. Preprint, Inst. Appl. Mathem., Russian Academy of Sciences, 2011, 12 Pages, 4 References.

This paper discusses the organization of software tools that provide the ability to create and use in environments of different browsers, unified virtual video content of the streams provided in various formats and transmitted from the distributed servers.

1. Введение.

Ранее авторами настоящей статьи была разработана технология, поддерживающая виртуальное управление мультимедийной потоковой информацией (видео и аудио), на базе этой технологии была осуществлена реализация действующего программного комплекса и проведена его опытная эксплуатация [1,2,3]. Эта реализация обеспечивает эффективное удаленное редактирование потоковой информации, которая транслируется в форматах, используемых в среде Real Media.

Однако, в последние два года область исследований, связанных с потоковыми технологиями, переживала период бурного развития. В результате сегодня в основном используются такие средства как FLASH, VLC и, в самое последнее время, HTML5. Соответственно, доставка видео и аудио информации осуществляется с использованием различных форматов. В связи с этим авторами была поставлена задача создать программные средства, способные функционировать в средах разных браузеров и обеспечивающие создание единого виртуального видеоконтента из потоков, не только транслируемых с территориально разнесенных серверов, но и представленных в различных форматах. Также предполагается разработать версию этих программных средств, предназначенную для использования на мобильных устройствах. При этом планируется использовать возможности средств глобальной навигации.

Разрабатываемый программный продукт может применяться как один из программных компонентов систем обеспечения принятия управляющих решений. В сфере образования этот инструментальный программный продукт позволит создавать и использовать учебные on-line курсы и образовательные энциклопедии. Еще одной областью возможных приложений рассматриваемых программных средств является информационное обеспечение средств массовой информации. Наконец, эти средства могут быть использованы в социальных сетях и коммерческих приложениях.

2. Современные форматы видео и аудио файлов.

Сейчас для трансляции мультимедийных потоков обычно используются следующие форматы видео: FLV (с кодеком H.264), MP4/MPEG (с кодеком H.264), WEBM (с кодеком VP8). FLV, используемый в среде FLASH, более распространен. Но с появлением браузеров с поддержкой HTML5 форматы MPEG и WEBM также набирают популярность. Среди аудио форматов, наиболее популярен MP3, но так же начинает набирать популярность OGG (с кодеком Vorbis).

Формат FLV используется только в FLASH, который может встраиваться в WEB страницы как плагин. Форматы MPEG, WEBM, MP3,

OGG поддерживаются непосредственно браузерами в HTML5. Анализ возможностей браузеров показывает такие результаты:

HTML5 поддерживают браузеры FireFox 4, Chrome 10, Opera 11, IE 9 (Vista и Windows 7), Safari 5.5. То есть сейчас уже есть реальная возможность использовать браузер с поддержкой видео HTML5, установив любой из этих браузеров. Различается поддержка форматов для видео HTML5 в этих браузерах. Chrome поддерживает все форматы (MPEG, WEBM, MP3, OGG). FireFox и Opera поддерживают только WEBM и OGG, и пока не предполагают обеспечить поддержку других форматов. IE 9 имеет встроенную поддержку только MPEG и MP3, но дает возможность использовать WEBM и OGG при установке соответствующих кодеков.

Таким образом, видно, что большинство используемых браузеров поддерживают или могут поддерживать форматы WEBM и OGG. По-видимому, на них можно ориентироваться как на основной формат для распространения видео в будущем, особенно при использовании мобильных устройств. Однако, нельзя не учитывать достаточно большое количество существующих ресурсов в форматах MPEG и, особенно, FLV для видео и MP3 для аудио. Поэтому хотелось бы, что бы разрабатываемые программные средства могли поддерживать различные форматы.

Для поддержки форматов, у которых нет встроенной в браузер поддержки, предполагается использовать VLC плагин. VLC – свободно распространяемый видео-плеер, способный воспроизводить файлы различных форматов и получать данные по сети используя различные протоколы (RTSP, UDP, HTTP). При установки плеера на компьютер, можно установить плагины, способные встраиваться в WEB страницы (также как происходит в среде FLASH). Имеются плагины для браузеров FireFox (также работает в Chrome и Opera) и для IE. Работой встроенного на страницу плагина можно управлять из Javascript.

3. Воспроизведение потоков, транслируемых по протоколу HTTP.

Для воспроизведения видео и аудио данных, получаемых через Интернет, обычно используется потоковая передача (стриминг), в отличие от полной загрузки данных. Это особенно актуально, если речь идет о воспроизведении достаточно объемной информации. При этом воспроизводятся данные, получаемые с сервера по мере их поступления, и предварительная загрузка всего контента не требуется. Для передачи информации используются различные протоколы передачи данных (HTTP, RTSP, RTP и др.), основанные на транспортных протоколах TCP или UDP. Транспортный протокол TCP используется для надежной передачи данных между клиентом и сервером. Он регулирует порядок передачи запросов и ответов, реакцию на сбои и ошибки при передаче, обеспечивает повторную отправку потерянных или испорченных пакетов данных. Протокол UDP более

простой и быстрый. Он просто обеспечивает отправку пакетов сервером, не заботясь о том, получает ли клиент эти данные, успевает ли клиент обрабатывать данные.

Сами передаваемые данные оформляются в соответствии с протоколами более высокого уровня, которые определяют как оформляются запросы и ответы, какие данные передаются (текстовые или двоичные).

Для передачи потоковых видеоданных часто используются протоколы RTSP и RTP. Протокол RTSP использует транспортный протокол TCP, разработан для передачи команд и информационных данных между медиа сервером и клиентом-плеером. Используя этот протокол, клиент может передавать серверу команды такие, как: начать передачу данных из указанного файла, перейти на нужный тайм-код в файле, пауза, получение информации о медиа-файле (продолжительность, размер экрана). По этому протоколу клиент и сервер могут согласовывать скорость передачи данных. Сами данные передаются по протоколу RTP, обычно используя в качестве транспортного протокола - UDP.

Однако, эти протоколы требуют наличия специализированных серверов для передачи мультимедийных данных. Такие сервера обычно работают и с содержимым медиафайлов. Так, по команде перехода на требуемый тайм-код, сервер сам должен найти в файле требуемые данные и начать их передачу. Это ограничивает возможность работы серверов с данными различных форматов. Кроме того, провайдеры часто блокируют входные данные по протоколу UDP межсетевыми экранами и файерволами в целях безопасности и уменьшения трафика. Поэтому стали пытаться приспособить наиболее часто используемый в Интернете протокол HTTP для передачи медиаданных. Протокол HTTP довольно медленный, поскольку помимо данных он требует передачи большого количества служебной информации. Раньше, на медленных сетях, это было существенным ограничением, но с ростом скорости передачи данных это перестало создавать большие проблемы. Сейчас протокол HTTP все чаще используется для передачи потоковых мультимедийных данных (иногда такой способ передачи называют псевдостримингом).

В принципе для осуществления потоковой передачи мультимедийных данных при взаимодействии плеера-клиента и сервера требуются две основные возможности:

- возможность воспроизведения данных по мере поступления, не дожидаясь полной загрузки файла (прогрессивная загрузка);

- возможность быстро перейти на нужное место по тайм-коду, не дожидаясь загрузки файла до этого места (быстрый поиск). Помимо непосредственного перехода по тайм-коду, данная возможность необходима для воспроизведения фрагментов файлов.

Рассмотрим возможности различных плееров (встроенные в браузеры с HTML 5, FLASH, и VLC) для воспроизведения мультимедийных файлов различных форматов (WEBM, JGG, FLV, MPEG4/H.264, MP3) с использованием потоковой передачи по протоколу HTTP.

С первым требованием все обстоит хорошо. Все плееры обеспечивают устойчивое воспроизведение файлов указанных форматов по мере их передачи, требуя только лишь предварительной загрузки небольшого количества данных в буфер (0,1-2 сек). На современных сетях загрузка данных обычно опережает воспроизведение.

Гораздо больше проблем возникает с быстрым поиском. Что бы быстро перейти на нужное место, плеер должен запросить у сервера только ту часть файла, с которой начинаются данные, необходимые для воспроизведения с нужного места. HTTP сервера могут обрабатывать запросы на передачу части файла, но для этого в запросе необходимо указать номер байта от начала файла, с которого требуется передать информацию, и количество передаваемых байт (ранжированный запрос). Сам сервер не оперирует тайм-кодами и не знает структуру передаваемого файла. Поэтому плеер должен сам, используя формат файла и тайм-код, определять номер нужного байта, чтобы сделать запрос серверу. Для определения положения байта, соответствующего тайм-коду в файле, должна быть соответствующая таблица, в которой для каждого ключевого кадра указан байт, с которого начинается информация (таблица ключевых кадров).

Таблица ключевых кадров всегда содержится в файлах формата WEBM и MPEG4, поэтому быстрый поиск в плеерах, встроенных в браузеры с поддержкой HTML5, для этих форматов осуществляется эффективно.

В формате аудио-файлов MP3 таблицы тайм-кодов не предусмотрено (о таблице ключевых кадров говорить нельзя, т.к. в аудио нет понятия ключевого кадра). Быстрый поиск для этого формата невозможен. Быстрый переход на требуемый тайм-код возможен только в пределах уже загруженной части файла.

В видео и аудио файлах формата OGG также нет таблиц тайм-кодов. Однако эксперименты показали, что плееры HTML5 в браузерах FireFox, Chrome и Opera, а также плеер VLC довольно быстро могут перейти на нужный тайм-код. Повидимому, в этом случае используется алгоритм подобный двоичному поиску. Зная общий размер файла и общую продолжительность содержимого, плеер примерно вычисляет место, где должны располагаться данные для требуемого тайм-кода и делает запрос на сервер на считывание содержимого, начиная с этого места, и проверяет находятся ли в этом месте требуемые пакеты. В случае неудачи делается перерасчет. Так, за несколько считываний коротких фрагментов находится нужная позиция в файле. В целом для больших файлов считывание нескольких коротких фрагментов оказывается гораздо быстрее, чем считывание всего файла до требуемой позиции.

Гораздо сложнее обстоит дело при использовании FLASH плеера. Различные FLASH плееры являются программами, написанными на языке ActiveScript и использующими стандартные видео и аудио элементы системы FLASH. Сами по себе эти элементы не предоставляют никаких возможностей для быстрого поиска при передаче данных по HTTP, но в ряде случаев их можно реализовать программно. Для этого необходимо, чтобы файл содержал таблицу ключевых кадров. С помощью программы на ActiveScript можно сначала считать эту таблицу, которая содержится в метаданных файла. Затем при переходе на требуемый тайм-код можно сделать запрос на сервер, что бы считывать данные с нужного места в файле. Из рассматриваемых форматов файлов таблицы ключевых кадров есть только в MPEG4 и FLV. Стандартными средствами кодирования в формате FLV таблица ключевых кадров не создается, но она может быть создана и вставлена в файл с помощью сторонних программ.

Однако, есть еще одна трудность. FLASH плеер не будет воспроизводить фрагмент файла, переданный начиная с произвольного ключевого кадра. Требуется, что бы ему предшествовал некоторый заголовок. Обычный HTTP сервер при ранжированном запросе никаких дополнительных заголовков создавать не будет, поэтому необходима дополнительная программа на сервере, которая обработает такой запрос, и передаст плееру нужный заголовок и требуемую часть файла. Чтобы обратиться к такой программе, необходимо в запросе задавать не имя требуемого файла, а имя программы, а имя файла передавать ей как параметр. Можно также настроить сервер так, что бы все запросы к FLV или MPEG4 файлам обрабатывались с помощью такой программы. Кроме того, как параметры необходимо передавать и указатели на требуемый фрагмент файла.

Заголовок, требуемый для FLV файлов, очень простой, и представляет из себя фиксированную последовательность из нескольких байт (она немного отличается для файлов, содержащих только видео, только аудио или видео и аудио информацию). Видимо поэтому серверных CGI программ для передачи FLV файлов создано очень много. Кроме того, существует специальный HTTP сервер Lighttp, в который подобная программа встроена. К сожалению, в различных системах используются различные параметры. Так, для обозначения начала запрашиваемого фрагмента в различных популярных программах используются следующие имена параметров:

Lighttp – "start"
Bitgravity – "apstart"
Edgecast – "ec_seek"
Limelight – "fs"

Многие FLASH плееры создаются в расчете на использование вместе с какой-то одной CGI программой и используют необходимый параметр. Некоторые FLASH плееры имеют возможность настройки, позволяющей

указать какие параметры необходимо использовать. Но, даже зная URL файла, не всегда можно узнать, какая именно программа установлена на сервере. Возможно имеет смысл, что бы FLASH плеер использовал в запросе все наиболее часто встречающиеся параметры, в расчете на то, что каждая CGI программа будет брать только требуемые ей и игнорируя остальные.

Аналогичная программа существует и для MPEG4(H.264) файлов. Структура файлов MPEG4 гораздо более сложная, чем у FLV файлов. При передачи фрагмента файла с нужного байта недостаточно просто передать фиксированный заголовок. Указанная программа выполнена в качестве модуля к HTTP серверу. Программа оформлена в качестве динамической библиотеки к HTTP серверам, работающим на операционных системах Linux. Эта программа обрабатывает запросы на передачу сервером MPEG4 файлов, и в случае запроса на передачу части файла, она фактически перестраивает файл, выделяя из него пакеты, относящиеся к требуемому фрагменту. При этом заново создаются все заголовки, таблицы и другая служебная информация MPEG4 файла. Фактически создается новый MPEG4 файл, который и передается плееру.

Это показывает, что при использований FLASH плеера для потоковой передачи с использованием HTTP нет достаточного универсализма. В общем случае невозможно обеспечить нужные параметры трансляции с произвольного сервера, на котором не установлены требуемые программы. Кроме того, невозможен быстрый поиск в звуковых, так как MP3 файлы не предусматривают таблиц тайм-кодов. А для FLV файлов таблица ключевых кадров возможна только для видео содержимого.

4. Моделирование программ, обеспечивающих модификацию и объединение потоков, представленных в различных форматах.

В рамках работ по созданию программных средств, обеспечивающих в среде различных браузеров, модификацию и объединение видео и аудио фрагментов, представленных в различных форматах и транслируемых с распределенных серверов, был создан и протестирован ряд программных моделей. Так, были проведены тесты по возможностям воспроизведения последовательностей фрагментов видео и аудио различных форматов в HTML5 и VLC и управлению, соответственно, видеоэлементом и плагином из Javascript. Для тестов использовались браузеры FireFox 4, Opera 11, Chrome 10.

Тесты с форматом WEBM (и MPEG для Chrome) показали очень хорошие результаты. Видео и аудио воспроизводятся в отличном качестве, последовательности фрагментов воспроизводятся практически без видимых задержек на границах. Видеоэлементом HTML5 легко управлять из JavaScript. Для этого используются следующие методы и свойства элемента:

vid.src="<url>" - задание адреса входного потока

vid.play() vid.pause() - задание режимов воспроизведения/паузы
vid.currentTime=<time> - задание таймкода для быстрого перемещения

vid.volume – управление громкостью
vid.muted – включение/выключение звука

vid.duration – определение общей продолжительности клипа
vid.networkState – определение состояния соединения и степени готовности элемента к воспроизведению
vid.paused, vid.seeking, vid.ended определение, что элемент находится в соответствующем состояние.
vid.error – определение, что произошла ошибка.

В разрабатываемом стандарте HTML5 [4] описаны и другие свойства и методы видеоэлемента, но они пока реализованы не во всех браузерах (например, степень заполнения буфера можно узнать только в FireFox, причем не в полном соответствие с описанным стандартом).

Для воспроизведения последовательности фрагментов использовался следующий метод. Для каждого фрагмента динамически создается видеоэлемент (в терминах HTML и DOM). Для этого используется функция JavaScript vid=createElement("video"). Пока этот элемент не подключен к DOM (объектная модель документа браузера, в которой перечислены все имеющиеся на WEB странице элементы, их свойства и связи между ними), он невидим. Но при этом им можно управлять, он способен считывать данные из заданного потока и позиционироваться на необходимый тайм-код. Все созданные элементы могут функционировать параллельно и независимо друг от друга. Для визуализации нужного элемента он должен быть подключен к DOM функцией appendChild(vid). При достижении конца фрагмента, элемент может быть удален с помощью функции removeChild(vid) (это не полное удаление элемента, а только удаление из DOM, сам элемент продолжает существовать и может быть снова подключен и использован в случае необходимости), и подключается элемент со следующим фрагментом в последовательности.

Функционирование видеоэлемента и управление им в различных браузерах практически одинаково. Экспериментально выявлены незначительные отличия при смене значений степени готовности элемента (свойство vid.networkState) в различных браузерах или реакции на вызов метода play(), когда элемент находится в состоянии seeking. Эти отличия нужно учитывать при программировании.

Эксперименты с VLC пока не дали таких позитивных результатов. Сам по себе, плагин VLC функционирует довольно хорошо. Он легко встраивается в WEB страницы и воспроизводит видеофрагменты различных форматов в высоком качестве. Но он не способен функционировать как

видеоэлемент без подключения к DOM. Это означает, что плагин нужно сначала сделать видимым, и только потом устанавливать соединение с потоком и позиционировать на нужный тайм-код. Это занимает определенное время, поэтому переход с одного фрагмента на другой бывает довольно заметным. Немного выручает то, что плагин можно сделать невидимым на какое то время, задав ему нулевые размеры (ширину и высоту). Использование нескольких плагинов, встроенных в страницу, иногда вызывает неустойчивую работу, а в Chrome вообще невозможно запустить более одного плагина VLC.

Управление VLC плагином из JavaScript, в принципе, мало отличается от управления видео-элементом HTML5. Есть похожие функции для воспроизведения/паузы, поиск (переход к тайм-коду), определение текущего состояния. Однако при их использовании возникают сложности, связанные с тем, что приходится обращаться к некоторым внутренним объектам плагина, которые не создаются одновременно с плагином, а только спустя какое-то время. Поэтому приходится проверять, что этот внутренний объект уже создан. Например, не получается запустить плагин сразу в режиме без звука:

```
vlc=document.createElement("embed"); // создание элемента и задание его свойств
vlc.type="application/x-vlc-plugin";
……..
vlc.audio.mute=true // отключение звука.
```

Внутренний объект audio еще может не существовать, и это вызовет ошибку JavaScript. Приходится опрашивать плагин с каким-то временным интервалом, чтобы убедиться, что этот объект существует.

Кроме того, возникают проблемы при управлении звуком, при одновременном функционировании 2-х и более плагинов. Если один из них перевести в режим без звука, то полностью пропадает возможность управлять звуком в других (не поддерживается изменение громкости звука и переключение режимов - со звуком/без звука). Это проявляется в браузерах FireFox и Opera.

Таким образом плагин VLC может быть ограниченно использован при создании плеера. На него можно переключаться для воспроизведения видео или аудио формата, который не поддерживается в браузере (например, MPEG или MP3 в Opera или FireFox). При этом не рекомендуется воспроизводить одновременно два неподдерживаемых формата (на видео MPEG накладывать звук MP3). Также следует учитывать временную задержку при переключении на такой фрагмент при воспроизведении последовательности фрагментов.

Значительные проблемы возникают при использовании FLASH и формата FLV.

Первое. При использовании формата FLV, чаще всего невозможно использовать быстрый поиск (быстрый переход на нужный тайм-код). При воспроизведении формата FLV используется прогрессивная загрузка – файл воспроизводится по мере загрузки, но при переходе на какой-то тайм-код, необходимо ждать, пока файл не загрузится до этого места. Переход ближе к концу длинного клипа может занять очень много времени. Проблема частично решается установкой определенных программ на сервер (которые способны начать передачу файла с нужного места), но в этом случае мы ограничены только воспроизведением фрагментов в формате FLV только с такого сервера (или нескольких серверов).

Второе. При создании данного плеера предполагается, что он может встраиваться на различные страницы. При управлении плеером используется JavaScript, и плеер управляется единообразно, показывая в нужные моменты видео то в элементе HTML5, то в окне VLC плагина, то в окне элемента FLASH. Элемент FLASH сам по себе представляет программу, написанную на языка ActiveScript (язык программирования FLASH). Эта программа должна загружаться на страницу с какого-то сервера и взаимодействовать со внешним окружением (программами на JavaScript на WEB странице). Это возможно, но с существенным ограничением: WEB страница и FLASH элемент должны быть загружены с одного сервера (с одного домена). В противном случае система безопасности браузера не позволит взаимодействовать программам на JavaScript и ActiveScript.

Таким образом использование формата FLV и FLASH возможно при выполнении этих условий: фрагменты в формате FLV могут использоваться только с серверов, на которых установлены программы, обеспечивающие быстрый поиск. Страницы, на которых используется видеоплеер, могут размещаться только на том же домене, где будет размещен созданный для плеера FLASH элемент.

Также были проведены эксперименты по синхронизации видео и аудио потоков. При хорошей скорости соединения все работает эффективно. Проблемы возникают, когда производится озвучивание видео звуком, транслируемым с другого удаленного сервера, и скорость соответствующих видео и аудио потоков сильно отличается друг от друга. Для предотвращения возможной рассинхронизации видео, с наложенным на него аудио был опробован алгоритм, обеспечивающий приостановку более быстрого потока (виде или аудио), пока медленный не достигнет нужного тайм-кода. Этот алгоритм используется в тех случаях, когда опережение по тайм-коду одного из потоков становится больше некоторой пороговой величины.

Литература

1. Ю.А. Бухштаб, А.А. Воробьев, Н.Н. Евтеева. Реализация редактора видео и аудио потоков, транслируемых с удаленных серверов. – Препринт ИПМ им. М.В.Келдыша РАН, №131, 2005. – 10 с.

2. Ю.А. Бухштаб, А.А. Воробьев, Н.Н. Евтеева. Программный комплекс для обработки потоковых данных, транслируемых с распределенных серверов. - Труды Всероссийской научной конференции «Научный сервис в сети Интернет», 2007. – с. 349-352.

3. Ю.А. Бухштаб, А.А. Воробьев, Н.Н. Евтеева. Некоторые подходы к организации виртуального управления потоковыми данными. - Препринт ИПМ им. М.В.Келдыша РАН, №72, 2009. – 10 с.

4. HTML5 - A vocabulary and assotiated APIs for HTML and XHTML. W3C Working Draft 25 May 2011.
http://www.w3.org/TR/html5/