УДК 004.773.5 DOI: 10.17587/it.26.177-184

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

Адаптивная бесшовная потоковая трансляция в реальном времени над протоколом HTTP методом опережающей загрузки

Предлагается технология организации адаптивной бесшовной подстройки качества изображения при изменении эффективной пропускной способности канала клиента для потоковой трансляции в реальном времени методом опережающей загрузки. Предложен способ измерения эффективной пропускной способности канала клиента не на самом клиенте, а на сервере в процессе проведения трансляции. Обсуждаются различные технические вопросы, связанные с реализацией адаптивности, и определены условия применимости адаптивной подстройки качества изображения для рассматриваемого подхода. Описана работающая диспетчерская система, реализующая технологию адаптивной бесшовной трансляции в реальном времени методом опережающей загрузки.

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

Введение

В настоящее время организация живой потоковой трансляции медиаинформации через всемирную сеть осуществляется с помощью различных технологий. Наиболее известные и используемые из них — Apple HLS [1], Adobe HDS [2], Microsoft Smooth Streaming [3] и MPEG DASH [4], которые в дальнейшем для краткости будем называть "манифестными" (manifest-технологиями). В этих технологиях клиент является активным участником процесса, управляющим процессом передачи фрагментов от сервера. Клиент время от времени получает от сервера "свежий" файл-манифест, который постоянно обновляется и отражает последнюю информацию о наличии на сервере доступных фрагментов потока для загрузки. Основываясь на метаинформации в файле-манифесте, клиент выполняет загрузку подходящих фрагментов потока для обеспечения непрерывного воспроизведения.

В работах [5, 6] задача организации живой потоковой трансляции решается с помощью технологии, базирующейся на методе опережающей загрузки (progressive download) потока. На первый взгляд эта технология существенно отличается от "манифестных", поскольку клиент является пассивным участником процесса. Клиент по своей инициативе делает только один единственный запрос на старт загрузки потока при подключении к серверу. После этого процессом передачи фрагментов от сервера управляет сервер.

Несмотря на указанное отличие, в обоих случаях проблемы передачи потока через реальную физическую среду у этих двух подходов одни и те же. Независимо от того, кто именно (клиент или сервер) управляет потоком, потоковую трансляцию реального времени можно рассматривать как процесс передачи от сервера к клиенту последовательности небольших фрагментов, снабженных меткой времени. Этот процесс зависит от текущего качества канала, которое может меняться со временем весьма непредсказуемо. При ухудшении качества канала скорость передачи будет падать, фрагменты начнут накапливаться в канале передачи данных и уже не смогут успевать поступать клиенту вовремя. Если не предпринять никаких действий по управлению потоком, то в итоге приемный буфер клиента может опорожниться, и воспроизведение живой презентации прервется.

Рассмотрим описанную выше ситуацию немного подробнее. С точки зрения процесса передачи потока от сервера к клиенту важнейшим параметром потока является его битрейт,

который далее для удобства будет измеряться в байтах потока, передаваемых в единицу времени. Битрейт потока является свойством потока и потому не зависит от качества канала. Битрейт потока меняется со временем, так как зависит от параметров кодирования потока — степени сжатия изображения, размера кадра, скорости изменения изображения, числа кадров в секунду, соотношения между ключевыми и разностными кадрами. Но как бы ни менялся со временем битрейт передаваемого через канал потока, он всегда должен быть меньше пропускной способности этого канала, т. е. меньше битрейта гипотетического потока, который бы заполнил собой весь канал передачи данных. В противном случае (что может случиться при ухудшении качества сети) канал не будет успевать пропускать через себя поток: непереданные вовремя данные будут накапливаться в буфере сервера, а в приемном буфере клиента объем данных начнет уменьшаться. Воспроизведение потока клиентом начнет опережать поступление потока и рано или поздно остановится.

Решение этой проблемы состоит в своевременном адаптивном изменении битрейта потока таким образом, чтобы фрагменты потока поступали клиенту "без опозданий" и, тем самым, воспроизведение презентации не останавливалось. Если качество канала упало, то надо уменьшить и битрейт передаваемого клиенту потока. И наоборот, если качество канала восстановилось, то можно будет вернуть потоку его прежний битрейт.

Технологии Apple HLS, Adobe HDS, Microsoft Smooth Streaming и MPEG DASH являются адаптивными. Сервер предоставляет клиенту на выбор несколько вариантов одного и того же фрагмента, но с разными битрейтами (т. е. качествами). Клиент, ориентируясь по ходу получения потока, загружает фрагменты того или иного битрейта, причем переключение между фрагментами делается по границам кадров (бесшовно — seamlessly), что обеспечивает плавность воспроизведения. Более детальное описание адаптивности для этих технологий приведено в п. 1.

Как было отмечено выше, в технологии опережающей загрузки потока [5, 6] клиент пассивно принимает передаваемый ему поток презентации и не может менять битрейт приходящего потока. Поэтому при падении пропускной способности канала клиента воспроизведение презентации на клиенте попросту остановится. Все, что клиент может сделать по своей инициативе — это прекратить воспроизведение текущей трансляции и начать новую, подключившись к потоку другого битрейта. Такую подстройку трудно назвать "адаптивной", и тем более она не будет бесшовной, поскольку произойдет неизбежная остановка воспроизведения презентации клиентом на время загрузки фрагментов, необходимых для начала воспроизведения трансляции другого качества. Каким же образом можно решить задачу организации адаптивной бесшовной трансляции для технологии опережающей загрузки потока, в которой клиент является пассивным участником?

В настоящей работе предлагается решение этой задачи. Адаптивность и бесшовность достигается тем, что процессом изменения качества видеоизображения управляет не клиент, а сервер. Сервер непрерывно вычисляет текущую ("эффективную") пропускную способность канала клиента и автоматически меняет качество передаваемого клиенту потока. Клиент при этом просто воспроизводит тот поток, который поступает ему от сервера. Такой подход диктует следующие условия на метод кодирования потока:

- метаданные (заголовок потока) должны передаваться перед медиаданными;
- изменение битрейта потока путем изменения качества изображения должно осуществляться *без изменения метаданных* (т. е. заголовка потока). Метаданные потока отправляются клиенту только один раз при его подключении к серверу, после чего качество создаваемых кадров меняется по ходу проведения трансляции;
- контейнер потока должен быть потоковым (однопроходным), т. е. декодирование медиаданных должно основываться на информации, содержащейся в самих текущих медиаданных, и не требовать предыдущих медиаданных.

Примером такого контейнера является контейнер Ogg [7] с видеокодеком Theora [8] и аудиокодеком Vorbis [9].

1. Адаптивная трансляция в manifest-технологиях

Схема работы механизма адаптивности для manifest-технологий следующая. Имея несколько *взаимозаменяемых* потоков разных битрейтов для одной и той же презентации, можно по ходу проведения трансляции подменять фрагменты презентации фрагментами с той же самой видео- и аудиоинформацией, но с другим битрейтом. Сервер создает несколько потоков разного качества (битрейта) для одной

и той же презентации. Клиент загружает поток презентации маленькими порциями — фрагментами длительностью воспроизведения в несколько секунд. Каждый фрагмент имеет уникальный URL. Клиент через периодически скачиваемый с сервера файл-манифест знает список всех имеющихся на сервере потоков, их битрейты, а также список доступных для скачивания фрагментов (т. е. их URL) каждого из этих потоков. Тем самым, клиенту всегда известны ссылки на свежие фрагменты одной и той же живой презентации, но с разным битрейтом. Основываясь на скорости загрузки фрагментов, клиент делает вывод о пропускной способности канала и может запросить следующий фрагмент с тем битрейтом (качеством), которое наиболее соответствует текущему состоянию канала передачи данных. Для того чтобы переключение потоков было бесшовным, начало каждого фрагмента, на котором проводится переключение потока, обязательно должно совпадать с началом кадра.

2. Адаптивная трансляция в технологии опережающей загрузки. Постановка задачи и общая схема ее решения

В случае технологии живой трансляции методом опережающей загрузки передача потока презентации от сервера к клиенту проводится также фрагментами, однако реализация этой передачи иная, чем описанная в предыдущем параграфе. Клиент не получает от сервера манифест и, соответственно, ничего не знает о существующих вариантах качеств потоков. Более того, он не запрашивает каждый раз фрагменты потока, а пассивно принимает их от сервера. При таком подходе организация бесшовной адаптации возможна, если решение о переключении отсылаемых клиенту фрагментов будет принимать не клиент, а сервер.

В настоящей работе предлагается следующая схема организации адаптивности. Источник презентации посылает серверу несколько потоков разного битрейта (рассмотрим случай двух). Сервер должен постоянно определять пропускную способность канала и посылать клиенту фрагменты того потока, битрейт которого более всего соответствует текущей пропускной способности канала. В случае необходимости смены битрейта сервер должен подменять фрагменты так, чтобы переключение делалось именно на границе кадра, а не на его середине, чем будет обеспечена бесшовность переключения.

На рис. 1 (см. вторую сторону обложки) проиллюстрирован случай, когда сервер вначале посылал клиенту поток высокого качества, затем некоторое время посылал поток низкого качества, после чего снова вернул потоку высокое качество. При этом последовательность кадров не изменилась: $n-1$, n, $n+1$, ..., $m-1$, m.

В предложенной выше схеме остался нераскрытым следующий вопрос — каким образом сервер сможет узнать текущую пропускную способность канала, чтобы осуществить переключение качества в нужную сторону?

Эффективная пропускная способность канала

Пропускной способностью канала называется наибольшая возможная скорость передачи данных. Таким образом, пропускная способность — это характеристика канала, имеющая отношение к его конструктивным свойствам и не имеющая отношение к загрузке канала. Пропускную способность канала можно измерить, проведя достаточно большое число измерений скорости передачи пакетов определенной длины (скажем, 512 байт) и найдя наибольшее значение из них. Если канал сильно загружен, то скорость доставки фрагмента, равная пропускной способности канала, достигается лишь для очень малого числа пакетов, тогда как для остальных пакетов скорость доставки окажется значительно ниже. В этом случае для пользователя гораздо большее значение имеет *эффективная пропускная способность* (ЭПС) канала — фактическая пропускная способность, которая может значительно меняться от степени загрузки канала в каждый момент времени. Эта величина представляет собой не характеристику канала, а переменную во времени характеристику загрузки канала. Измерить эффективную пропускную способность можно путем измерения скорости передачи данных при посылке максимально возможного объема данных за заданный интервал времени $\Delta t_{\text{ЭПС}}$. Вследствие неравномерности загрузки канала при разных измерениях мы будем получать разные значения ЭПС.

На рис. 2 (см. вторую сторону обложки) приведен пример типичной картины изменения ЭПС во времени. Мы видим, что временной тренд ЭПС в целом "поджат" к верхнему порогу $9 \cdot 10^6$ байт/с, что соответствует теоретической пропускной способности использовавшегося канала в 100 Мбит/с. Тренд несимметричен по оси ординат и содержит нерегулярные провалы в направлении меньших значений пропускной способности. Их значение и частота должны возрастать с ухудшением работы канала связи, например, при увеличении общей нагрузки на канал от разных пользователей.

Адаптивность и ЭПС

Суть использования адаптивности заключается в том, чтобы битрейт передаваемого клиенту потока был всегда ниже ЭПС. В этом случае скорость воспроизведения потока на клиенте всегда будет оставаться меньше, чем скорость передачи фрагментов, вследствие чего приемный буфер клиента будет всегда достаточно заполнен. В изображенном на рис. 2 случае в течение первых 20 мин трансляции оптимальный битрейт передаваемого клиенту потока не должен был превышать $6 \cdot 10^6$ байт/с (зеленая линия). В момент времени "B" (1260 с) ЭПС вдруг "просела" и далее в течение более чем трех минут регулярно оказывалась ниже $6 \cdot 10^6$ байт/с. Если не менять битрейт потока, то приемный буфер клиента рано или поздно опустошится, вследствие чего возникнет задержка воспроизведения видеоинформации. Поэтому в момент времени "B" сервер должен был переключиться на поток с меньшим битрейтом $4 \cdot 10^6$ байт/с (красная линия).

В момент времени 1400 с восстановилась прежняя ЭПС. Сервер, конечно, может и не предпринимать никаких действий и продолжать по-прежнему передавать поток с низким битрейтом. Но все же неплохо было бы воспользоваться улучшением ЭПС канала и переключить поток обратно на высокое качество $6 \cdot 10^6$ байт/с (зеленая линия).

Изображенная на рис. 2 ситуация длительного (более секунды) изменения ЭПС возникает не всегда. Весьма частыми событиями являются однократные понижения ЭПС — "просадки". Если длительность однократной "просадки" такова, что это не приведет к опустошению видеобуфера клиента, то такое событие не вызовет задержки отображения видеоинформации. Однако каждая такая "просадка" будет приводить к уменьшению накопленных данных в приемном буфере клиента. Например, в течение времени от "A" до "B" сервер мог бы передавать клиенту поток с битрейтом $7 \cdot 10^6$ байт/с. При этом мы наблюдаем четыре однократные "просадки" ЭПС ниже уровня $7 \cdot 10^6$ байт/с. При каждой такой "просадке" приемный видеобуфер клиента уменьшается на длительность воспроизведения:

$$\Delta t_{\text{прием.буфер}} = \Delta t_{\text{просадки}}(1 - Bwe/Br), \qquad (1)$$

где Bwe — ЭПС в момент "просадки"; Br — битрейт передаваемого клиенту потока; $\Delta t_{\text{просадки}}$ — общая длительность "просадки".

Если "просадка" недолгая, то приемный видеобуфер клиента не успевает опустошиться, и через некоторое время он снова будет заполнен, так как в итоге все задержанные фрагменты потока будут переданы клиенту "в срок". Определить, является очередное уменьшение ЭПС кратковременным или нет, сервер конечно не может и *обязан* переключиться на менее качественный поток. Если "просадка" ЭПС недолгая, то после нее сервер тут же вернется к передаче клиенту предыдущего потока с высоким битрейтом. Произойдет однократное кратковременное адаптивное переключение качества потока "высокий битрейт → низкий битрейт → высокий битрейт", которое, по сути, будет паразитным. Частота таких однократных паразитных переключений качества увеличивается с возрастанием частоты "просадок" и уменьшается с возрастанием длительности времени измерения ЭПС $\Delta t_{\text{ЭПС}}$. Все, что мы можем сделать для уменьшения числа таких однократных переключений качества — это выбрать некий *оптимальный* интервал $\Delta t_{\text{ЭПС}}$ измерения ЭПС.

Совмещение измерения ЭПС с процессом трансляции потока

В процессе трансляции потока от сервера к клиенту проводить какие-то специальные измерения ЭПС канала "сервер—клиент" избыточно, поскольку это можно делать с помощью самого же отправляемого потока. Это позволяет совместить передачу потока с постоянным измерением ЭПС этого же потока "в режиме online". Для этого сервер должен посылать данные клиенту не непрерывно по мере их поступления от источника презентации, а накапливать данные в буфере FIFO и отсылать блоками так, чтобы *длительность отсылки блока* клиенту равнялась заданному значению $\Delta t_{\text{ЭПС}}$. В настоящей работе в качестве оптимального было принято значение $\Delta t_{\text{ЭПС}} = 1000$ мс.

Как было отмечено в предыдущем параграфе, величина $\Delta t_{\text{ЭПС}}$ не должна быть чересчур малой (менее секунды), так как это может привести к частым паразитным переключениям битрейта потока. Вместе с тем она и не должна быть слишком большой (более десяти секунд), так как тогда измеренная ЭПС будет представлять усредненное значение ЭПС канала связи за эти десять секунд и не будет отражать его текущего состояния. Кроме того, значение задержки воспроизведения презентации на клиенте прямо связано со временем накопления блока. Длительность накопления $\Delta t_{\text{накопл}}$ блока, который потом будет отправлен клиенту за время $\Delta t_{\text{ЭПС}}$, равна

$$\Delta t_{\text{накопл}} = \Delta t_{\text{ЭПС}} \cdot Bwe/Br. \qquad (2)$$

Если битрейт потока (*Br*) в несколько десятков раз меньше ЭПС канала клиента (*Bwe*), то выбрав $\Delta t_{ЭПС}$ = 1000 мс, получим накопления блока в несколько десятков секунд и соответственно такую же задержку воспроизведения живой презентации на клиенте.

Строго говоря, необходимость введения блочной передачи потока требуется только для адаптивного переключения на повышенный битрейт. Если же ЭПС канала упала, то это можно понять уже по нескольким первым фрагментам отправляемого блока.

Бесшовное переключение качества

Требование бесшовности накладывает условие: переключение на поток другого качества (битрейта) не должно выполняться в произвольный момент времени, а только в момент начала нового кадра. Таким образом, блоки в FIFO должны содержать целое число кадров и быть взаимозаменяемыми соответствующими блоками FIFO потока другого битрейта. На рис. 3 (см. вторую сторону обложки) представлен пример переключения отправляемому клиенту потока, иллюстрирующий ситуацию, показанную на рис. 2. Отдельные прямоугольники изображают кадры, которые двумя потоками с разным качеством поступают от источника презентации на сервер (ретранслятор), размеры прямоугольников отражают размеры кадров: кадры большего качества имеют большие размеры. Кадры поступают в два буфера FIFO соответственно и накапливаются там блоками. Каждый блок изображен горизонтальной полоской, состоящей из нескольких кадров. Блок потока высокого качества изображен состоящим из трех кадров, а блок потока низкого качества состоит из 10 кадров. Поскольку время отсылки каждого блока, независимо от его качества, одинаковое (1000 мс), то размеры блоков разного качества изображены также одинаковыми.

В процессе отсылки блока сервер может обнаружить падение ЭПС, при этом он должен переключиться на поток с меньшим качеством. Он делает это не сразу, а сначала дожидается отсылки текущего кадра клиенту. Это проиллюстрировано укороченными блоками, но все равно состоящими из целого числа кадров.

От начала трансляции "A" до момента времени "B" сервер отправлял клиенту поток высокого качества. Он успел передать несколько блоков, после чего в момент времени "B" произошла "просадка" ЭПС. В процессе отправки очередного блока сервер обнаружил "просадку", дождался завершения отправки текущего кадра и переключил отправку на поток низкого качества. В момент времени "C" ЭПС канала восстановилась до прежнего значения, сервер дождался завершения отправки текущего кадра и переключил отправку на поток высокого качества.

Сервер должен структурировать все поступающие ему медиапотоки на отдельные кадры и отслеживать последовательность номеров кадров в каждом потоке. Если сервер выявил необходимость переключения на поток другого битрейта, то прежде всего он должен дождаться завершения отправки клиенту данных текущего кадра *N*. Затем найти кадр $N + 1$ в буфере FIFO потока необходимого битрейта, дождаться накопления в нем блока (начиная с кадра $N + 1$) такого размера, чтобы его пересылка клиенту составляла величину 1000 мс и только потом отправить этот блок.

Область применимости и эффективность предлагаемого подхода

Рассмотрим случай, когда ЭПС клиентского канала превышает битрейт потока наибольшего качества. По сути, в этом случае адаптивность уже не требуется, и необходимости в блочной передаче нет. Тем не менее сервер все равно должен постоянно контролировать ЭПС, чтобы не пропустить ситуацию с ее ухудшением. Для этого сервер должен накапливать блок в течение времени, которое может оказаться весьма значительным (равным десяткам секунд). Например, в соответствии с формулой (2) для типичной ЭПС = 10^7 байт/с и потока с битрейтом $Br = 2 \cdot 10^5$ байт/с (копия экрана 1280 × 1024, 30 кадров/с, кодек Theora) серверу придется каждый раз ожидать 50 с, чтобы накопить очередной блок, который он будет потом отправлять клиенту в течение 1 с. Такого же порядка (50 с) будет и задержка воспроизведения презентации на клиенте. И чем больше ЭПС будет превышать битрейт потока, тем дольше сервер будет накапливать блоки. В принципе, если трансляция не налагает требований реального времени, такая задержка не будет представлять серьезную проблему для клиента. Тем не менее, чтобы не снижать эффективность использования блоков, требуется ограничить сверху размер блока. В настоящей работе максимальный размер блока был ограничен значением 1 Мбайт.

3. Практическая реализация изложенной схемы бесшовной адаптивной подстройки качества видеоизображения

Идеи, изложенные выше, были реализованы практически путем расширения разработан-

ной ранее системы трансляций презентаций в реальном времени (диспетчерской системы) [10], структурная схема которой изображена на рис. 4 (см. третью сторону обложки).

Диспетчерская система состоит из трех базовых частей:

1. Кодировщик медиапотока, который принимает медиаданные с источника презентации (персональный компьютер, ip/web камера, микрофон), кодирует их в поток Ogg с кодеками Theora / Vorbis и отправляет кодированные медиаданные на Ретранслятор.

2. Ретранслятор, который принимает медиаданные от нескольких Кодировщиков и ретранслирует эти потоки клиентам, а статусную информацию обо всех медиапотоках помещает в базу данных. Один и тот же медиапоток может отправляться нескольким клиентам одновременно.

3. Набор HTML-страниц на web-сервере для выбора и просмотра клиентом презентаций. Страницы формируются автоматически, основываясь на информации из базы данных.

Расширение диспетчерской системы на адаптивность потребовало ряда изменений в структуре диспетчерской системы, рассмотренной в работе [10], которые базировались на уже функционирующей системе промежуточных FIFO-буферов. Первое изменение состояло во введении структурирования потоков от Кодировщика к Ретранслятору, которые в системе [10] просто передавались Ретранслятору "как есть" по мере кодирования презентации. В адаптивной версии диспетчерской системы к потоку был добавлен специальный пакет — маркер начала кадра, который передается Ретранслятору Кодировщиком перед каждым кадром. Ретранслятор должен сам надежно выявлять наличие маркера начала кадра в потоке данных.

Второе изменение состояло в том, что Кодировщик медиапотока стал кодировать презентацию не в один, а в три параллельных потока — высокого, среднего и низкого качества. Эти три потока отправляются на Ретранслятор по трем различным TCP-портам. Ретранслятор хранит медиаданные каждого потока в отдельном кольцевом буфере FIFO. Каждый буфер FIFO теперь разбивается на блоки, в каждом из которых содержится по нескольку целых кадров, их число определяется изложенным выше алгоритмом. Ретранслятор пересылает клиенту медиаданные блоками по мере их появления в буфере FIFO.

Третье изменение касалось алгоритма работы Ретранслятора. Теперь Ретранслятор постоянно следит за двумя величинами:

- скоростью поступления потока в FIFO с Кодировщика (*Bwe*);
- скоростью отправки блока данных из FIFO Клиенту (*Br*).

Ретранслятор переключает качество отправляемого Клиенту потока по условию повышения и понижения *Br* относительно *Bwe*.

4. Проверка и анализ реализации бесшовной адаптивной подстройки качества

Тестирование адаптивной версии системы трансляций проводили путем искусственного изменения пропускной способности канала клиента с помощью коммутатора второго уровня с фиксированными значениями пропускной способности (512, 1024, 2048, ..., Unlimit kbit/sec). Использовали три градации качества видеопотока, получаемых разной степенью сжатия при Theora-кодировании, причем разрешение экрана для всех качеств было одинаковым 640 × 480@20 fps. На рис. 5 (см. третью сторону обложки) представлен типичный результат изменения ЭПС канала клиента (тренд изображен синим цветом) относительно битрейтов трех принимаемых потоков от Кодировщика. Высокий, средний и низкий битрейты изображены фиолетовым, зеленым и красным цветами соответственно. При этом в процессе изменения ЭПС канала клиента Ретранслятор совершил три переключения качества — от высокого к среднему, от среднего к низкому и от низкого к среднему:

- в момент времени 180 с ЭПС упала до ~350 000 байт/с, что уже сопоставимо с битрейтом H-потока, поэтому сервер переключил поток клиента на среднее качество (H → M);
- в момент времени 210 с ЭПС упала до ~60 000 байт/с, и сервер переключил поток клиента на плохое качество (M → L);
- в момент времени 810 с ЭПС возросла до ~350 000 байт/с, и для клиента появилась возможность воспроизводить поток среднего качества. Сервер переключил качество на среднее (L → M).

В моменты переключения качества передаваемого клиенту потока воспроизведение видео- и аудиоинформации на клиенте продолжает идти плавно, без потерь видеокадров и аудиосэмплов. При резком падении ЭПС канала клиента остановок воспроизведения не наблюдалось, так как при падении приемный буфер клиента не успевал опустошаться из-за своевременного изменения качества потока.

Дополнительно проводилось сравнение процесса адаптивного переключения качества системы трансляций [10] с работой сервиса YouTube. Сравнение велось по двум критериям — битрейту установившегося потока и задержке между реальным событием и его воспроизведением в браузере клиента. В системе трансляций использовались три уровня качества потока одинакового разрешения 1280 × 720@30 fps и с разной степенью сжатия Theora-кодирования. Сервис YouTube предоставляет пять градаций качества, отличающихся разрешениями экрана 144p (254 × 144@30 fps), 240p (426 × 240@30 fps), 360p (640 × 360@30 fps), 480p (854 × 480@30 fps), 720p (1280 × 720@30 fps).

При искусственном установлении пропускной способности канала на фиксированные уровни 512, 1024 и 2048 Кбит/с в обоих случаях происходило адаптивное переключение трансляции на поток подходящего качества со сравнимыми битрейтами (см. таблицу).

Адаптивные битрейты установившихся потоков сравниваемых технологий для заданных значений пропускной способности

Система конференций, битрейт потока	YouTube, битрейт потока	Пропускная способность канала
2000 Кбит/с (высокое качество)	2000 Кбит/с (720p)	2048 Кбит/с
1000 Кбит/с (среднее качество)	640 Кбит/с (480p)	1024 Кбит/с
370 Кбит/с (низкое качество)	240 Кбит/с (240p)	512 Кбит/с

Минимальная задержка между реальным событием и его воспроизведением для системы трансляций [10] составила 10...13 с. Соответствующий параметр плеера YouTube "Live Latency" давал значение в районе 30 с, что более чем вдвое превысило полученный выше результат. Впрочем, задержку "Live Latency" можно искусственно уменьшить. Идея состоит в том, что частью этой задержки является буферизация (около 20 с), соответственно время декодирования составляет 10 с. Можно ускорить воспроизведение до такой степени, когда буферизация составит 5 с, после чего вернуть нормальную скорость воспроизведения. Тогда общая задержка YouTube будет равной 15 с, однако воспроизведение при этом становится неустойчивым.

Заключение

В настоящее время разработан и активно используется ряд технологий адаптивной трансляции в реальном времени [1—4]. Эти технологии объединяет тот факт, что клиент сам определяет эффективную пропускную способность сети и организует адаптивность путем загрузки фрагментов презентации того качества, которое является оптимальным в данный момент.

Для рассмотренной в работе технологии трансляции методом опережающей загрузки такой способ организации адаптивности невозможен, так как клиент является пассивным участником процесса трансляции. В настоящей работе предложено решение этой задачи. В основе решения лежит идея о том, что инициатором адаптивной подстройки является не клиент, а сервер. В работе обсуждено поведение ЭПС канала клиента и предложена идея совмещения измерения ЭПС с отправкой клиенту потока. Для этого сервер посылает клиенту медиаданные не равномерно, а группирует их в блоки. Во время отсылки байтов блока сервер определяет ЭПС сети клиента. Сервер переключает передаваемый клиенту поток на битрейт, который наиболее подходит для ЭПС канала клиента. Для обеспечения бесшовности переключения потока на другое качество сервер всегда заканчивает отсылку кадра текущего потока и начинает отсылку блоков потока другого качества строго на границе кадра. Область применимости предложенного решения адаптивной трансляции определяется условием на соотношение между ЭПС канала клиента и битрейтом потока: ЭПС канала клиента не должна превышать битрейт потока самого высокого качества.

Предложенный в работе способ организации адаптивности был реализован на практике путем расширения разработанной ранее системы трансляций [10]. Ретранслятор реагирует на все изменения эффективной пропускной способности канала клиента и своевременно переключает передаваемый ему поток на другое качество. При уменьшении ЭПС канала происходит переключение потока на меньший битрейт, буфер клиента не опустошается, а последовательность кадров не прерывается. В итоге клиент воспроизводит трансляцию без перерывов, аудио-, видеоискажений или задержек.

Сравнение работы системы трансляций для 1280 × 720@30 fps с работой видеосервиса YouTube по критерию "битрейт потока" дало сравнимые результаты, а для критерия "задержка между реальным событием и его воспроизведением" система трансляций дала более хороший результат.

В сравнении с обычными методами переключения качества потока, когда решение

о переключении принимает клиент, в настоящей работе предложена схема адаптивности, в которой решение о переключении принимает сервер. Это является главной особенностью используемого в работе подхода к организации адаптивности, и она может представлять собой определенное преимущество по сравнению с другими технологиями для тех случаев, когда клиент не может использовать специальные плагины или расширения для воспроизведения потока в условиях изменяющейся пропускной способности канала. От клиента в этом случае лишь требуется поддержка опережающей загрузки потока в рамках спецификации HTML5.

Список литературы

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

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

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

4. **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.

5. **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.

6. **Лобов И. В., Готман В. Г.** Использование контейнера Ogg для организации потоковой трансляции в реальном времени над протоколом HTTP методом опережающей загрузки // Информационные технологии. 2018. № 2. С. 87—96.

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

8. **Theora Specification.** Xiph.Org Foundation, 2011, 196 p.

9. **Vorbis I specification.** Xiph.Org Foundation, 2015, 74 p.

10. **Лобов И. В., Готман В. Г.** Система трансляций презентаций в реальном времени над протоколом HTTP // Известия Института инженерной физики (ИИФ). 2018. № 3. С. 60—66.

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",
Moscow, Russian Federation

Adaptive Bitrate Seamless Live Streaming over HTTP by Progressive Download Method

The paper proposes a technology for organizing the adaptive bitrate seamless live adjustment of image quality while changing the client channel effective in live streaming system based on the proggresive download. A server hosted method of measuring client channel bandwidth while live streaming has been proposed. Diverse technical issues related to the implementation of adaptive bitrate seamless streaming were discussed. Applicability conditions for the adaptive image quality adjustment for the technology under consideration were determined. A fair-functioning dispatch system that implements the adaptive bitrate seamless live streaming by proggresive download method has been described.

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

DOI: 10.17587/it.26.177-184

References

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

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

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

4. **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.

5. **Lobov I., Gotman V.** Real time media streaming over HTTP using progressive download method, *Tehnologii i Sredstva Svyazi*, 2016, no. 5, pp. 36—40 (in Russian).

6. **Lobov I., Gotman V.** The Utilization of Ogg Multimedia Container Format for Live Streaming over HTTP Using Progressive Download Method, *Informatsionnye Tekhnologii*, 2018, no. 2, pp. 87—96 (in Russian).

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

8. **Theora Specification.** Xiph.Org Foundation, 2011, 196 p.

9. **Vorbis I specification.** Xiph.Org Foundation, 2015, 74 p.

10. **Lobov I., Gotman V.** Live streaming system over http, Izvestiya Instituta Inzhenernoj Fiziki (IIF), 2018, no. 3, pp. 60—66 (in Russian).

Рисунки к статье И. В. Лобова, В. Г. Готмана

«АДАПТИВНАЯ БЕСШОВНАЯ ПОТОКОВАЯ ТРАНСЛЯЦИЯ В РЕАЛЬНОМ ВРЕМЕНИ НАД ПРОТОКОЛОМ HTTP МЕТОДОМ ОПЕРЕЖАЮЩЕЙ ЗАГРУЗКИ»

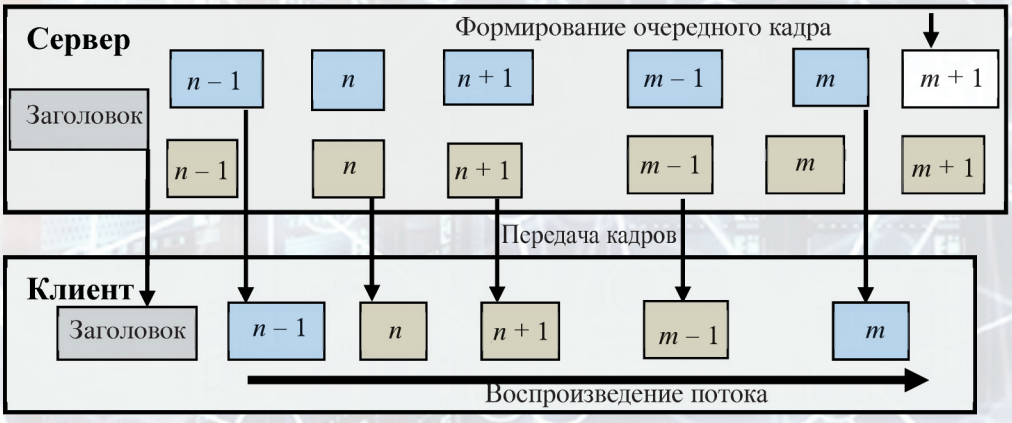

Рис. 1. Схема формирования клиенту видеопотока переменного битрейта

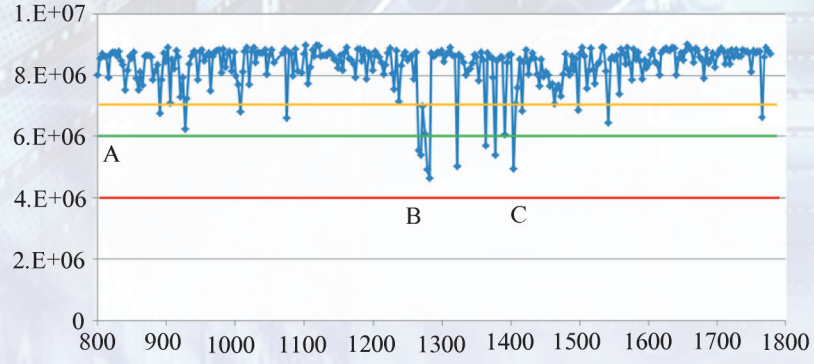

Рис. 2. Типичная картина временного тренда эффективной пропускной способности канала передачи данных (байт/с), $\Delta t_{эпс}$ = 500 мс, измерения проводились каждые 3,5 с

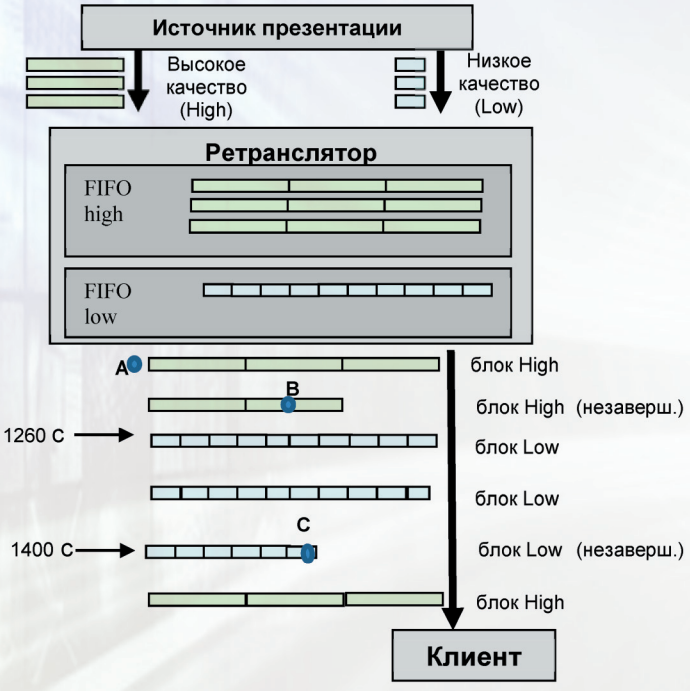

Рис. 3. Схема переключения качества потока, иллюстрирующая ситуацию рис. 2

Рисунки к статье И. В. Лобова, В. Г. Готмана

«АДАПТИВНАЯ БЕСШОВНАЯ ПОТОКОВАЯ ТРАНСЛЯЦИЯ В РЕАЛЬНОМ ВРЕМЕНИ НАД ПРОТОКОЛОМ HTTP МЕТОДОМ ОПЕРЕЖАЮЩЕЙ ЗАГРУЗКИ»

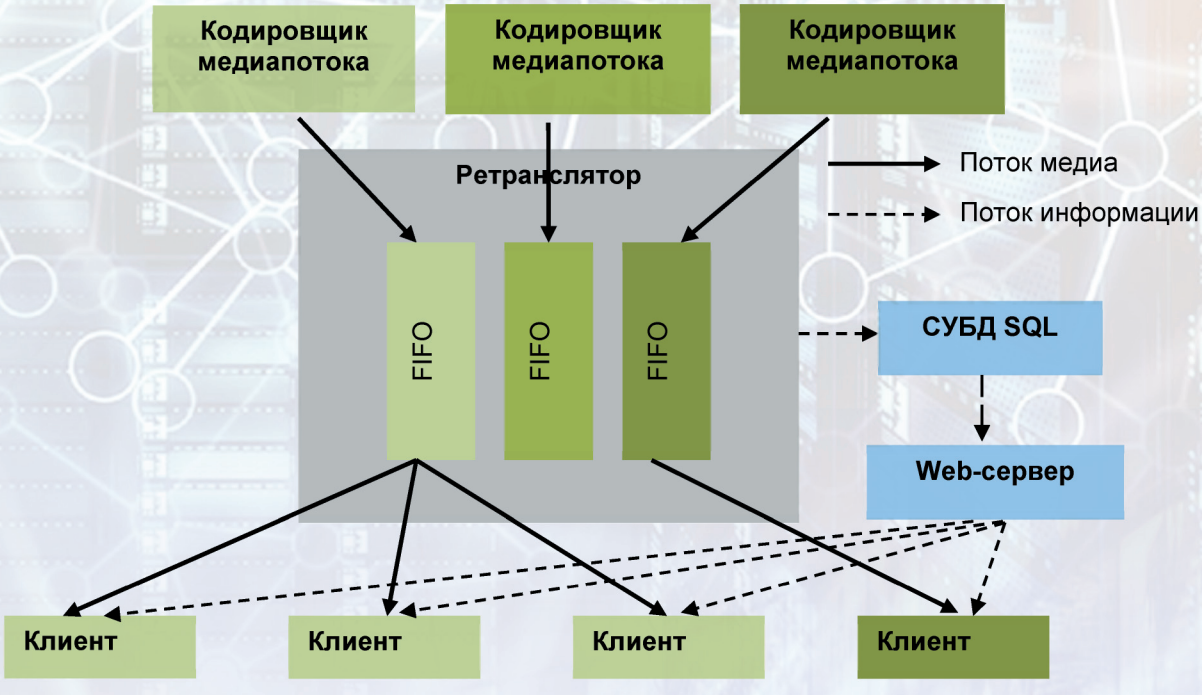

Рис. 4. Схема взаимодействия компонентов диспетчерской системы [10]

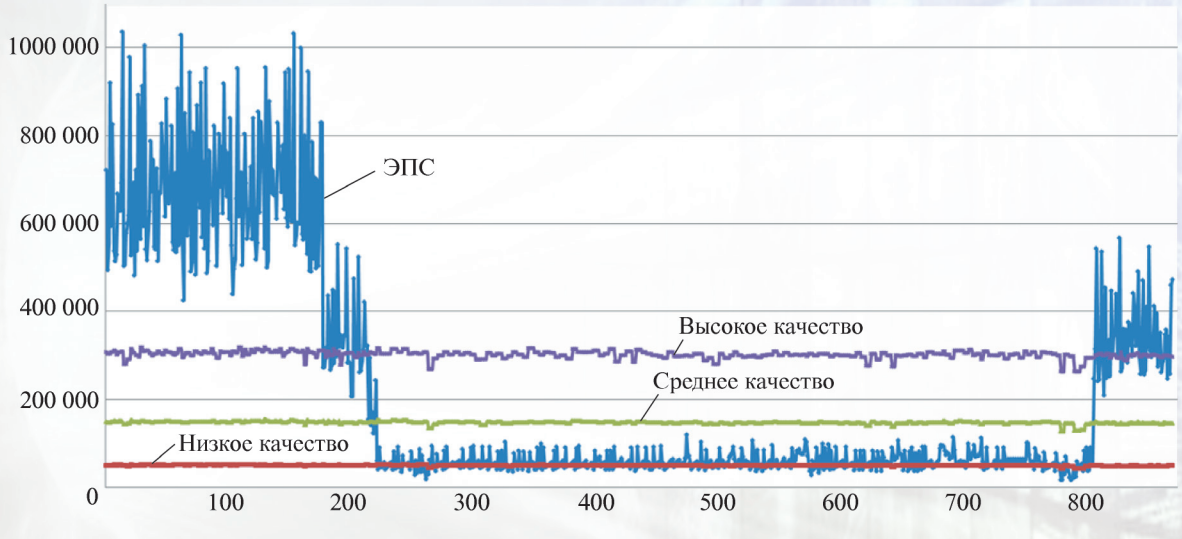

Рис. 5. Пример временного тренда эффективной пропускной способности клиента (байт/с) при тестировании системы

Рисунки к статье И. В. Лобова, В. Г. Готмана

«АДАПТИВНАЯ БЕСШОВНАЯ ПОТОКОВАЯ ТРАНСЛЯЦИЯ В РЕАЛЬНОМ ВРЕМЕНИ НАД ПРОТОКОЛОМ HTTP МЕТОДОМ ОПЕРЕЖАЮЩЕЙ ЗАГРУЗКИ»

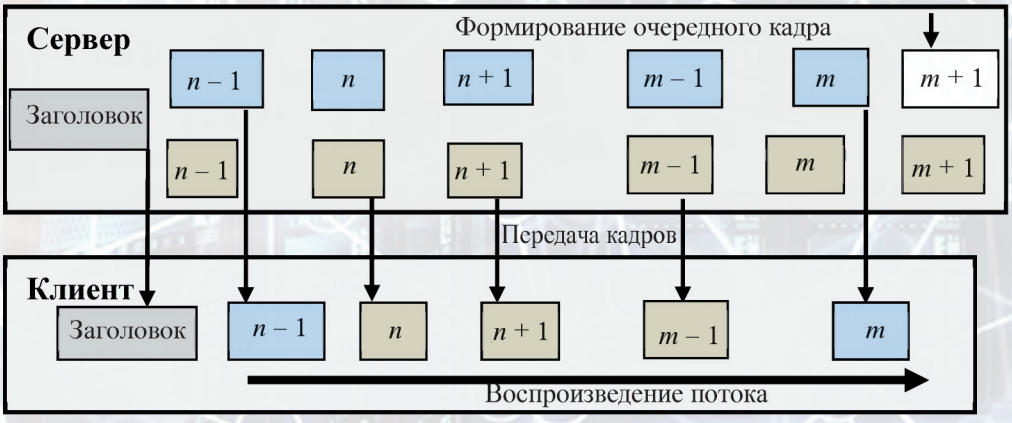

Рис. 1. Схема формирования клиенту видеопотока переменного битрейта

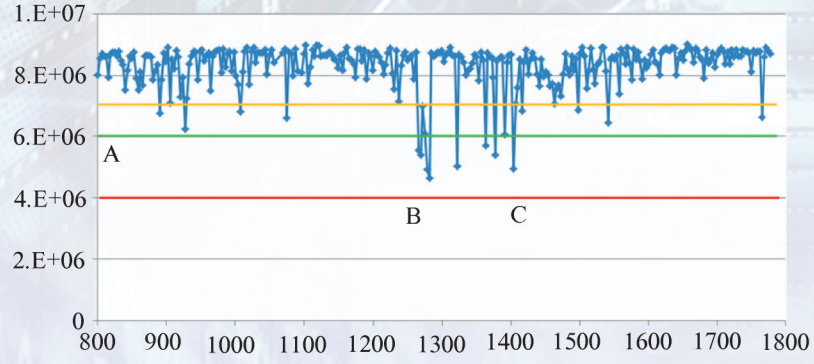

Рис. 2. Типичная картина временного тренда эффективной пропускной способности канала передачи данных (байт/с), $\Delta t_{эпс}$ = 500 мс, измерения проводились каждые 3,5 с

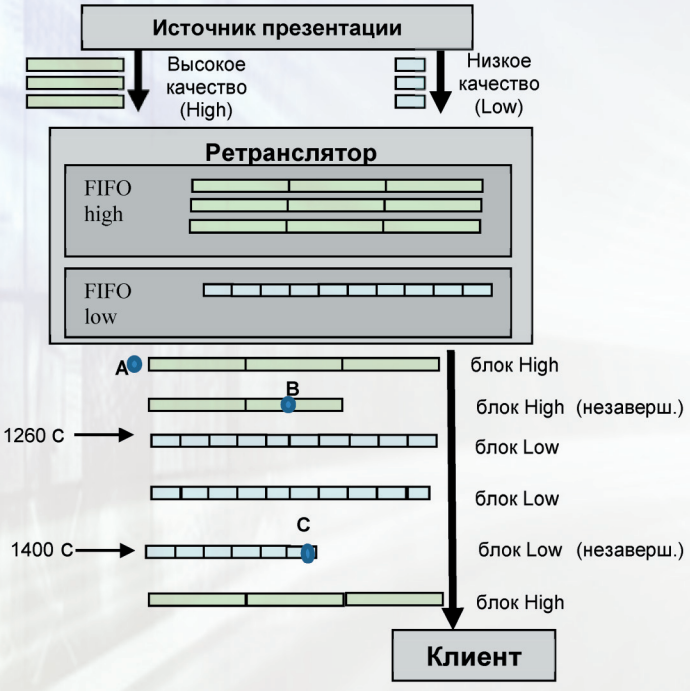

Рис. 3. Схема переключения качества потока, иллюстрирующая ситуацию рис. 2

Рисунки к статье И. В. Лобова, В. Г. Готмана

«АДАПТИВНАЯ БЕСШОВНАЯ ПОТОКОВАЯ ТРАНСЛЯЦИЯ В РЕАЛЬНОМ ВРЕМЕНИ НАД ПРОТОКОЛОМ HTTP МЕТОДОМ ОПЕРЕЖАЮЩЕЙ ЗАГРУЗКИ»

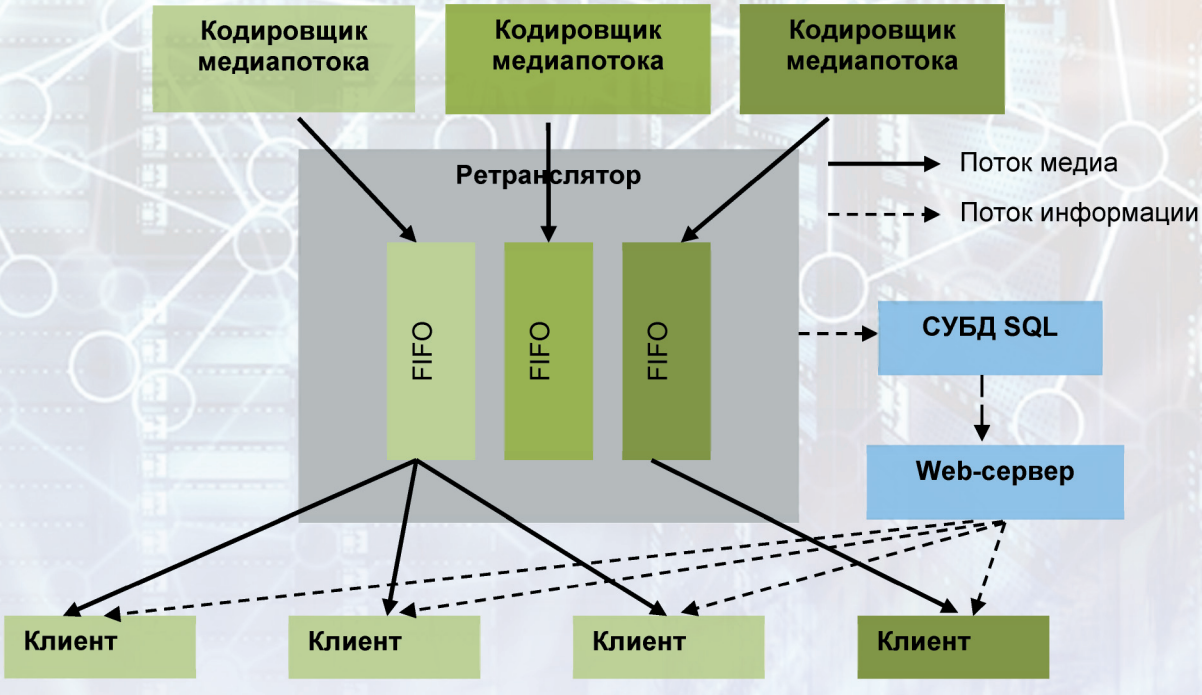

Рис. 4. Схема взаимодействия компонентов диспетчерской системы [10]

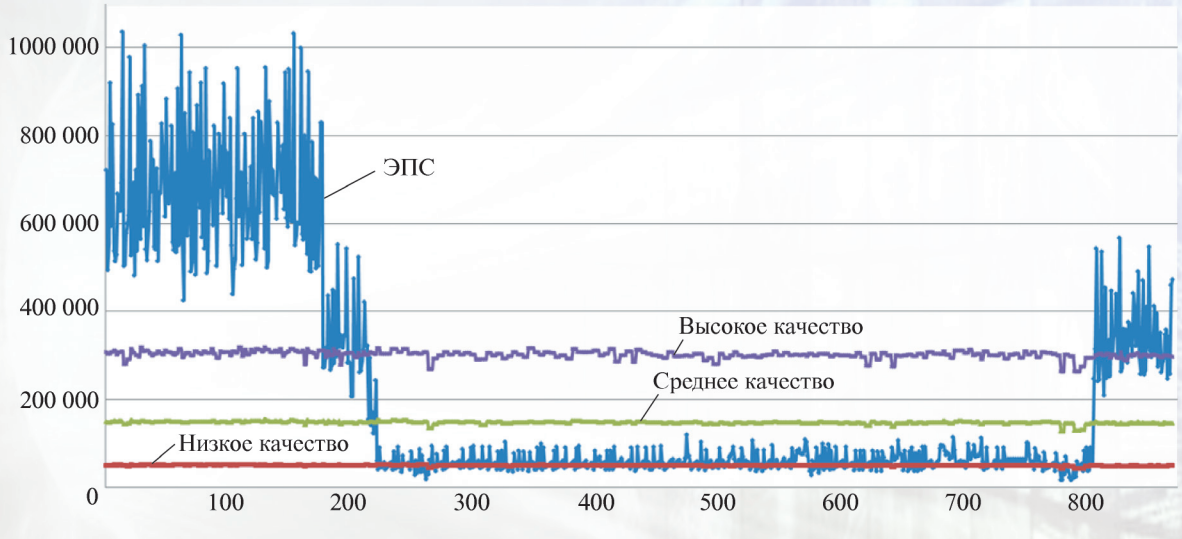

Рис. 5. Пример временного тренда эффективной пропускной способности клиента (байт/с) при тестировании системы