УДК 681.3.06

МЕТОДЫ ОПТИМИЗАЦИИ КАЧЕСТВА ПОТОКОВЫХ ВИДЕОИЗОБРАЖЕНИЙ ПРИ ПЕРЕДАЧЕ ПО СЕТИ ИНТЕРНЕТ

О.А. Оксюк, О.А. Оксюк

Физико-технический учебно-научный центр НАН Украины
03680, Киев-142, бульвар Вернадского, 36,
факс: +380(44) 424 8250, т.: + 380(44) 424 3025, oksyuk@netcracker.com

Работа посвящена вопросу оптимизации качества видеоизображения при передаче потокового видео по сети Интернет. Рассматривается случай гарантированной доставки пакетов. При этом учитываются такие характеристики сети как ограниченная и меняющаяся пропускная способность. Построена аналитическая модель, учитывающая дерево зависимостей между элементами данных видеопоследовательности. Рассмотрены подходы ее решения, основанные на составлении расписания передачи пакетов.

The article is dedicated to questions of optimizing video quality in case of streaming over Internet. The packet delivery is considered to be assured. Limited and time-varying bandwidth of the Internet is considered. Analytic model that takes into account tree of dependencies between data elements inside video is built. Approaches to its resolving that based on packets scheduling are considered.

Введение

Видеоинформация – важная составляющая данных уже более ста лет. С ростом мощности компьютерной техники в начале 80-х годов началось активное развитие цифрового видео. А в середине 90-х с ростом пропускной способности каналов сети Интернет начинает активно развиваться потоковое видео.

Потоковое видео имеет очевидные преимущества по сравнению с локально сохраненным цифровым видео, поскольку оно позволяет сразу или почти сразу начинать проигрывать видео, а не ждать пока оно загрузится. Кроме того, позволяет экономить дисковое пространство, поскольку видеоизображение не обязательно сохранять локально.

Но потоковое видео имеет и ряд проблем, поскольку информация передается по сети, что гораздо медленнее и менее надежно, чем считывание видео с локального устройства. При передаче видеоинформации по сети Интернет возникают следующие основные проблемы:

- низкая пропускная и иногда меняющаяся способность сети;
- потеря части информации;
- флуктуации времени задержки доставки пакетов в сети.

В работах [1–4] предлагаются различные методы решения этих проблем, но основной упор делается на решении проблемы потери части информации. Решение этой проблемы, как правило, основывается на получении нотификаций от клиента, что не всегда технически возможно.

В работе [5] эта же проблема решается другим способом. Там учитывается изменение качества изображения для каждой пары элементов данных.

Цель данной работы – решение проблемы низкой и меняющейся пропускной способности сети для передачи цифрового видео по сети Интернет. Решение, предлагаемое в данной работе, хорошо работает в условиях, в которых вторая и третья проблемы гораздо менее критичны, чем первая. При этом предполагается, что пропускная способность сети динамически меняется.

Будем различать два случая кодирования видеоизображения для передачи потокового видео:

- предварительное кодирование – видеоизображение хранится на сервере в файле и запрашивается разными клиентами в разные моменты времени. Примеры использования: видео по требованию, доставка потокового видео на персональный компьютер через Internet;
- кодирование в режиме реального времени – видеоизображение на сервере кодируется и сразу же или через небольшой промежуток времени передается. Примеры использования: видеофоны, видеоконференции, интерактивные видеоигры.

В данной работе предлагаются решения, применимые в обоих случаях.

Постановка задачи

Видеоизображение в медиасистемах с цифровой передачей состоит из элементов данных, которые передаются в виде пакетов данных. Будем считать, что в общем случае пакет может содержать несколько элементов данных, но каждый элемент данных целиком содержится в пакете. При этом с прикладной точки зрения видео элемент данных является неделимым, если потерялась часть элемента данных, то весь элемент не может быть использован. Обычно элементами данных являются видеокадры или части видеокадров, во втором случае видеокадр может состоять из нескольких элементов данных.

В современных алгоритмах видеокомпрессии, вне зависимости от того, что собой представляют элементы данных, между ними существует зависимость в виде ациклического графа (рис. 1). Каждый узел графа представляет собой элемент данных, а каждая ветка из элемента данных L' в элемент данных L представляет зависимость элемента L от элемента данных L'. Будем ее обозначать L' ← L, что означает, что элемент данных L не может быть декодирован на клиенте до тех пор, пока не декодирован элемент данных L'. Так, в случае MPEG-2 P кадры зависят от I или другого P кадра, а B кадры зависят от 2xP или I кадров. На рис. 1 I, B, и P кадры обозначены символами I, B и P соответственно.

Кроме того, в современных алгоритмах видеокомпрессии видеопоследовательность делится на блоки данных, которые, в свою очередь, делятся на элементы данных. При этом блоки данных независимы друг от друга, каждый блок данных имеет отдельное дерево зависимостей и может быть использован без декодирования предыдущих блоков данных. В рассматриваемой модели основная идея состоит в том, что если пропускная способность сети не позволяет проигрывать видеоизображение с необходимой скоростью, то передается не все видеоизображение, а только часть его элементов. Очевидно, что имеет смысл передавать элемент данных L только в том случае, если все элементы данных, от которых он зависит, уже передались или точно известно, что они будут переданы. При этом мы последовательно передаем блоки данных друг за другом, решая для каждого блока, какие элементы данных сможем передать, а какие нет.

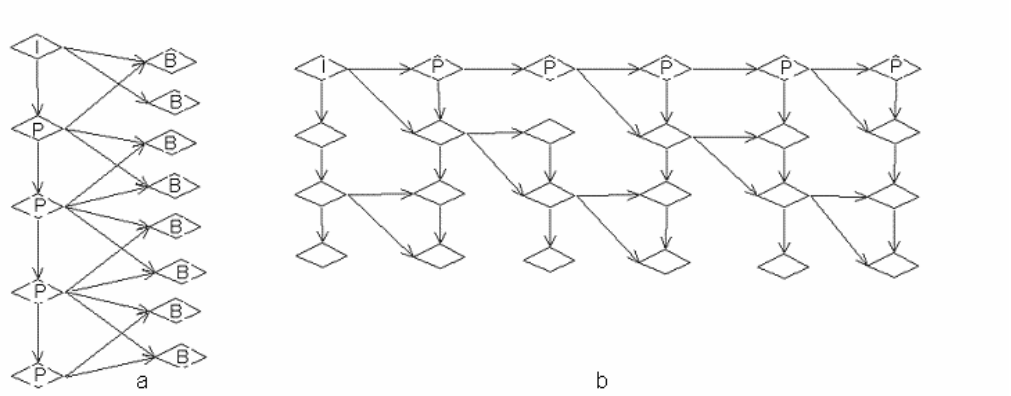

Рис. 1. Ациклический граф зависимостей между элементами данных видеоряда:
a – зависимость для видеоряда MPEG-2 IBBPBBPBBP; b – типичная зависимость для видеоряда формата MPEG-4 (прогрессивный)

Пусть M – множество элементов данных одного блока, Bi – размер элемента данных i, а Di – улучшение качества изображения при декодировании элемента данных i. Тогда в общем случае задача формулируется так. Нужно найти подмножество M' такое, чтобы видеоизображение имело наилучшее качество:

$$\sum_{i \in M'} Di \to \max . \tag{1}$$

Размер (в байтах) передаваемых данных ограничен:

$$\sum_{i \in M'} Bi \le B . \tag{2}$$

Все элементы, зависящие от элементов множества M' принадлежат M':

$$\forall i \in M', \forall j \in M : j \leftarrow i \Rightarrow j \in M . \tag{3}$$

В данной работе качество видеоизображения измеряется в децибелах, как это было предложено в [7]:

$$Q = 10.log_{10}\left(\frac{(2^r-1)^2}{\sigma^2}\right) \text{dB} ,$$

где $\sigma = \sum_{x=1}^{A} \sum_{y=1}^{B} (p(x,y) - p'(x,y))^2$.

Здесь r – количество бит в пикселе; A, B – размеры x, y видеоизображения в пикселях; p(x, y), p'(x, y) – значение пикселя с координатами (x, y) в воспроизводимом и оригинальном изображении.

Алгоритм адаптации к дереву зависимостей

При отсутствии дерева зависимостей между элементами данных легко можно показать, что оптимальным решением будет включение тех пакетов, для которых $\lambda i = Di / Bi \ge \lambda$.

В этом случае можно применить простой алгоритм, который находит оптимальное решение:

отсортировать все элементы данных по λ;

один за другим последовательно включать все элементы данных в порядке убывания пока их общий размер не превысит максимальный.

Основываясь на этом решении можно предложить решение в случае зависимостей между элементами:

1) находим решение M' по вышепредложенному алгоритму, как в случае, если бы элементы не зависели друг от друга;

2) для каждого элемента данных, не вошедшего в M’ находим Ki – число элементов в дереве вошедших в M’, которые от него зависят;

3) выбираем элемент из M/M’ с максимальным значением K;

4) если для этого элемента значение K равно нулю, то выходим из цикла – оптимальная выборка найдена;

5) добавляем Ki в M’;

6) если размер всех элементов из M’ превысил пороговое значение, то удаляем из M’ один или несколько элементов, от которых не зависит ни один из элементов M’ до тех пор, пока суммарный размер всех элементов из M’ не превысит пороговое значение. При этом выбираются элементы с минимальным значением λ;

7) пересчитываем значения K, отнимая единицу у тех элементов, от которых завесили элементы, удаленные на шаге 6;

8) переходим к шагу 3;

Этот алгоритм показан на рис. 2. В данном случае элемент со значением K = 9 включается во множество, а если размер всех элементов из M’ превысил пороговое значение, то один из элементов обозначенных черным кружочком удалится (если после этого размер всех элементов из M’ все еще превышает пороговое значение, то удаляется следующий и т.д.).

Данный алгоритм адаптации к дереву зависимостей не всегда приводит к оптимальному решению. Но он достаточно быстро выполняется, причем сложность зависит от того, насколько много останется элементов из M/M’ с ненулевым значением K.

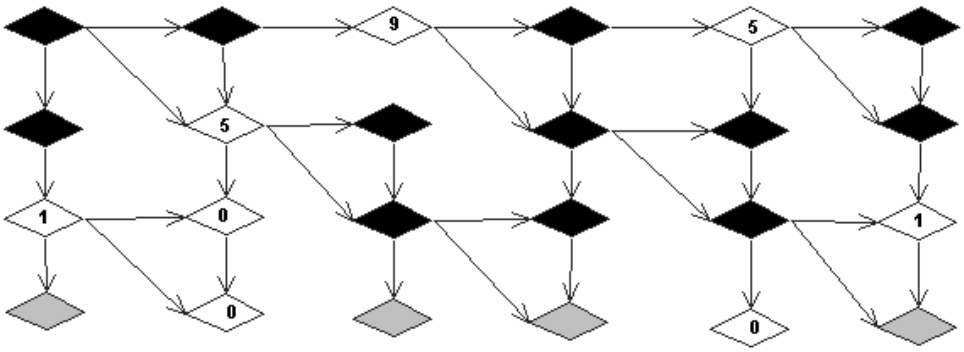

Рис. 2. Алгоритм адаптации к дереву зависимостей. Черным цветом обозначены элементы множества M’ от которых зависят другие элементы, серым – элементы множества M’ от которых не зависят другие элементы, числами обозначено число элементов в дереве вошедших в M’ которые зависят от элемента (K)

Алгоритм перебора с возвратом

Перебор всех вариантов, в отличие от алгоритма адаптации, всегда приводит к оптимальному решению, но требует большого времени выполнения. Поэтому логично было бы использовать алгоритм перебора с возвратом, который сначала включает в текущее решение только корень, а потом последовательно добавляет по одному элементу по следующему алгоритму:

1)обозначим множеством M’ текущее решение. На первом шаге оно пустое;

2)включаем в M’ корень дерева (если при этом нарушается условие (2), то выходим из цикла, при этом решение является пустым множеством);

3)находим множество N – таких узлов, что все узлы, от которых они зависят, уже включены;

4)отбрасываем из множества N такие узлы, которые при включении в M’ нарушают условие (2) для M’;

5)если после шага 3 или 4 множество N стало пустым, то выходим из цикла – решение, которое является множеством M, найдено;

6)для каждого элемента из N вычисляем значение оценочной функции;

7)включаем в M’ элемент N с максимальным значением оценочной функции;

8)переходим к шагу 3.

При этом оценочной функцией алгоритма перебора с возвратом с шагом T является $\sum Di/\sum Bi$ для всех возможных выборок из T элементов по вышеописанному алгоритму. Таким образом, в простейшем случае алгоритм перебора с возвратом с шагом 1 на первом шаге включает корень дерева, на втором один из узлов зависящий от корня и т. д., каждый раз включая во множество M’ новый узел с максимальным значением λ, такой, что все узлы, от которых он зависит, уже включены. Алгоритм перебора с возвратом с шагом T рекурсивно проходит по данному алгоритму с глубиной T, каждый раз включает такой узел дерева, который дает максимальное значение $\sum Di/\sum Bi$ для всевозможных выборок из T элементов удовлетворяющих условию 3.

При T, равном количеству элементов, алгоритм перебора с возвратом, по сути, сводится к перебору всех вариантов. При T меньше количества элементов, алгоритм перебора с возвратом так же не всегда находит оптимальное решение. Например, если встретится элемент с низким значением λ, от которого зависят много

элементов с высоким значением λ, то этот элемент не будет включен и таким образом все элементы, от которых он зависит, не будут включены.

Экспериментальные результаты

Алгоритмы, предложенные в данной работе, использовались для построения расписания передачи элементов данных в видеосервере, который передает пакеты данных по протоколу TCP/IP. Поскольку TCP/IP – протокол с гарантированной доставкой, то пакеты не теряются, и выполняется условие отсутствия потери информации. Для выполнения условия ограничения флуктуаций времени задержки, размер буфера на клиенте выбран достаточно большим, чтобы можно было передавать пакеты на 1-2 минуты вперед. Таким образом, флуктуации времени задержки не проявляются, за исключением случаев временной потери соединения. Такие случаи исключались из экспериментальных результатов и не влияли на ход эксперимента. Эксперимент проводился для фильма в формате MPEG-4 с частотой 30 элементов в секунду и числом элементов данных в блоке около 500, что соответствует около 17 сек. проигрывания видеоизображения.

На рис. 3 показан график зависимости качества получаемого видеоизображения от количества шагов в алгоритме перебора с возвратом.

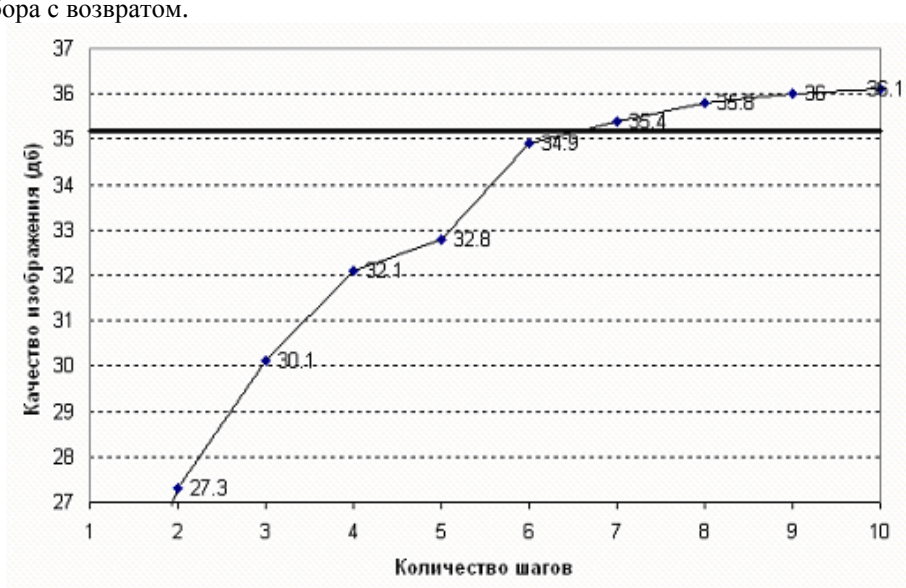

Рис. 3. Зависимость качества получаемого видеоизображения от алгоритма: жирной линией обозначен алгоритм адаптирования к дереву зависимостей, простой линией – алгоритм перебора с возвратом

На рис. 4 показаны графики зависимости быстродействия алгоритма от количества шагов в алгоритме перебора с возвратом. На графиках также показаны значения для алгоритма адаптации к дереву зависимостей. Как видно из графиков, алгоритм адаптации к дереву зависимостей дает такое же качество изображения, как и алгоритм перебора с возвратом с шагом 6 – 7, при этом по быстродействию он работает как алгоритм перебора с возвратом с шагом 3–4. Поэтому его целесообразно применять вместо алгоритма перебора с возвратом с шагом от 3–4 до 6–7.

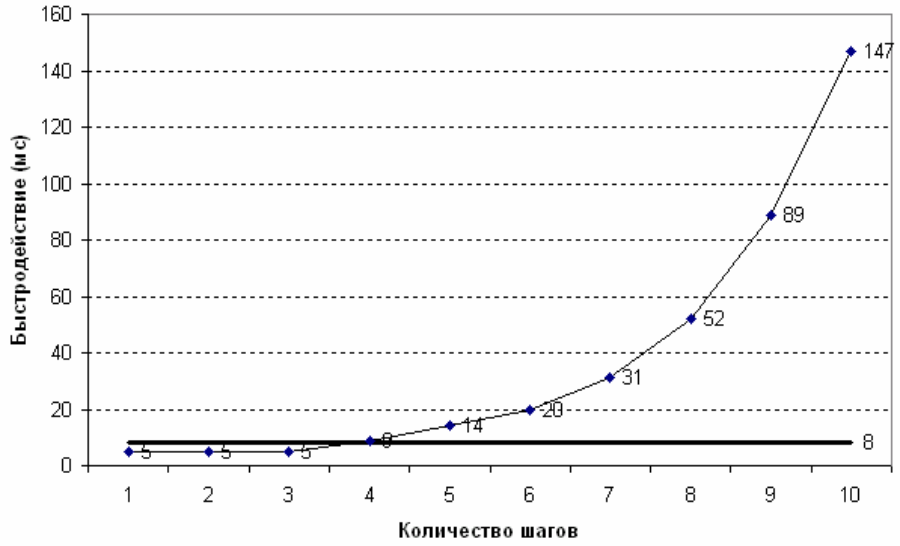

Рис. 4. Зависимость быстродействия от алгоритма: жирной линией обозначен алгоритм адаптирования к дереву зависимостей, простой линией – алгоритм перебора с возвратом

Заключение

В настоящей работе предложены алгоритмы составления расписания передачи потокового видео по сети интернет: алгоритм перебора с возвратом и алгоритм адаптации к дереву зависимостей. Показано, что эти алгоритмы дают существенный выигрыш в качестве декодируемого видеоизображения по сравнению с передачей элементов данных подряд друг за другом. Таким образом, в случае предварительного кодирования видеоизображения на сервере целесообразно сгенерировать хинт-трек для видеоизображения – метаданные, указывающие порядковый номер, под которым элемент данных включался в подмножество передаваемых элементов в алгоритме перебора с возвратом. При этом во время передачи блока данных сервер будет передавать все номера последовательно, начиная с 1, до тех пор, пока не закончится лимит времени передачи блока данных. В этом случае целесообразно использовать алгоритм перебора с возвратом с шагом 8–9. При этом время хинтирования (составления хинт-трека) для двухчасового видеофильма будет составлять от 2 до 48 часов.

1. *Chou P, Miao Z.* Rate-distortion optimized streaming of packetized media // IEEE Trans. on Multimedia. – 2001, № 2. – P. 3–9.
2. *Podolsky M., McCanne S., Vetterli M.* Soft ARQ for Layered Streaming Media // J. of VLSI Signal Processing Systems for Signal, Image and Video Technology, Special Issue on Multimedia Signal Processing, Kluwer Academic Publisher, April 2000. – P. 1–3.
3. *Chou P.A., Morh A.E., A.* Wang and S. Mehrotra, FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video // in Proc. Data Compression Conf., Snowbird, UT, Mar. 2000, IEEE Computer Society. – P. 3–6.
4. *Miao Z., Ortega A.* Optimal Scheduling for Streaming of Scalable Media // Proc. Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, November, 2000. – P. 1–5.
5. *Chakareski J., Apostolopoulos J., Wee S.* Wai-tian Tan, Girod B. R-D Hint Tracks for Low-Complexity R-D Optimized Video Streaming // IEEE International Conference on Multimedia and Expo (ICME) Taipei, Taiwan, 27-30 June, 2004,. – P. 3–4.
6. *Prashant J. Shenoy* Efficient Support for Interactive Operations in Multi-resolution Video Servers Distributed Multimedia Computing Laboratory Department of Computer Sciences, University of Texas at Austin Taylor Hall 2.124, Austin, Texas 78712-1188. – P. 16.