$$y'_{i+1}(x) = y'_i(x) + \tilde{\rho}_i(x) = y'_1(x) + \tilde{\rho}_1(x) + \tilde{\rho}_2(x) + \ldots + \tilde{\rho}_i(x),$$
$$y_{i+1}(x) = y_i(x) + \tilde{\varepsilon}_i^*(x) = y_1(x) + \tilde{\varepsilon}_1^*(x) + \tilde{\varepsilon}_2^*(x) + \ldots + \tilde{\varepsilon}_i^*(x), \quad i \geq 1.$$

Если степень m многочлена $p_m(\xi)$ невелика, то непосредственным перемножением скобок в правой части (14) этот многочлен приводится к виду, удобному для точного интегрирования. Чтобы эту процедуру автоматизировать, что особенно актуально в случае $m >> 1$, перепишем многочлен (14) в форме

$$p_m(\xi) = \sum_{k=0}^{m} (-1)^k \xi^{m-k} p_{m,k} . \qquad (15)$$

Так как по построению $p_m(\xi) = (\xi - \xi_m) p_{m-1}(\xi)$, $m \geq 1$, то для определения коэффициентов $p_{m,k}$ из (15) можно записать следующие рекурсивные формулы:

$$p_{m,k} = \begin{cases} 1, & k = 0, \\ p_{m-1,k} + \xi_m p_{m-1,k-1}, & k < m, \\ \xi_m p_{m-1,k-1}, & k = m. \end{cases}$$

Очевидно, что эти коэффициенты, как и сам многочлен $p_m(\xi)$, не связаны с исходной задачей, а зависят только от выбора точек $\xi_1, \xi_2, \ldots, \xi_m$.

В заключение заметим, что в общей схеме рассмотренных построений существенных изменений не произойдет, если в качестве стартового приближения вместо (3) избрать приближенное решение задачи (1), (2), полученное любым другим численным методом, при этом под производной приближенного решения следует понимать его локальную производную [3].

1. Бобков В.В., Бобкова Н.А. // Вестн. Белорус. ун-та. Сер. 1. 1994. № 2. С. 47.
2. Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи: Пер. с англ. М., 1990.
3. Бобков В.В. // Дифференц. уравнения. 1995. Т. 31. № 7. С. 1174.

Поступила в редакцию 21.06.2000

УДК 621.321.1:519.1

Н.И. ЛИСТОПАД, А.Г. КОПАЧЕВ

СИНХРОНИЗАЦИЯ ПОТОКОВ МУЛЬТИМЕДИА В РАСПРЕДЕЛЕННЫХ СРЕДАХ

In this paper we have presented a set of algorithms for achieving synchronization in a best-effort distributed system. These algorithms, based on numerical timestamps, take into account both the possible frame lost due to the device buffers overflow at the sender, and the display time of a video frame at the receiver.

We extended the solution to n continuous streams synchronization that have the same or different sources.

Одна из основных проблем при реализации систем мультимедиа – способ синхронизации потоков данных. В статье описывается новый метод синхронизации потоков в распределенных средах с использованием временных меток. Представлены алгоритмы, реализующие данный метод, которые основаны на числовых метках и учитывают как потери кадров из-за переполнения буфера устройства, так и время вывода кадра у получателя.

Вопросы синхронизации потоков данных рассмотрены в ряде литературных источников, однако наиболее подробно в работах [1–7]. При этом, как правило, анализируются только два потока (аудио и видео). На практике же, особенно при широком использовании мультимедиаприложений при проведении видеоконференций, возникают ситуации, когда потоки формируются различными n источниками в распределенной среде. Данная задача специалистами ранее не решалась.

Постановка задачи

Рассмотрим непрерывные аудио-, видеопотоки. Так как временные отношения между кадрами при формировании потоков могут искажаться, то вводится процедура синхронизации для оценки временных отношений между потоками [1]. Каждому кадру потока назначаются числовые временные метки. Если потоки сформированы одним источником, то процедура определения временных отношений между ними простая, так как у них один источник синхронизации (часы). Если потоки формируются разными источниками, то необходимо введение механизма централизованной синхронизации или эмуляции глобальных для этих потоков часов.

Таким образом, необходимо задать для каждого кадра корректный номер (временную метку) в соответствии с тем порядком, в котором он захвачен драйвером устройства, а не с которым он был доставлен в приложение, а также учесть наличие нескольких независимых источников потоков.

Модель схемы синхронизации

Рассмотрим модель на основе числовых меток (порядковые номера кадров).

Введем следующие обозначения:

1) α_i– порядковый номер i-го кадра, полученного программой;

2) α_{play}– порядковый номер кадра потока α, который воспроизводится в данный момент;

3) d_α – продолжительность кадра в потоке α;

4) n_α – количество буферов в очереди драйвера устройства потока α;

5) diff_α – временная разница между двумя последовательными операциями чтения из потока α;

6) lost_α – количество потерянных кадров потока α в промежутке между двумя последними операциями чтения из-за переполнения очереди драйвера устройств;

7) $t_{0\alpha}$ – время начала потока α;

8) $A_{\alpha\beta}$ – максимально допустимое время расхождения потоков α и β (асинхронность);

9) $t_{\alpha\beta}$ – толерантность (максимально допустимое расхождение потоков α и β, выраженное количеством кадров).

Порядковый номер кадра зависит от алгоритма, реализованного драйвером устройства в случае переполнения очереди. Тогда количество потерянных кадров выражается следующей формулой:

$$\mathrm{lost}_\alpha = \begin{cases} \dfrac{\left[\mathrm{diff}_\alpha - n_\alpha d_\alpha\right]}{d_\alpha}, & \text{если} \quad (\mathrm{diff}_\alpha - n_\alpha d_\alpha) \rangle 0. \\ 0 \text{ иначе} \end{cases} \qquad (1)$$

Алгоритм 1

(случай, когда драйвер устройства переписывает кадры поверх старых в циклическом режиме)

```
Пока процесс обрабатывается {
  ПолучитьКадр (fr);     /* Получить кадр из очереди */
  α_c = α_c + 1 + lost_α; /* Вычислить следующий порядковый номер */
  МаркироватьКадр(fr, α_c) /* Назначить вычисленный номер текущему кадру */
    Пока (очередь ≠ 0 ) {
       ПолучитьКадр (fr);
       α_c = α_c + 1;
       МаркироватьКадр(fr, α_c);}
}
```

Алгоритм 2

(случай, когда драйвер устройства не помещает данные в очередь после ее заполнения)

```
α_c = 1;
Пока процесс обрабатывается {
  ПолучитьКадр (fr);     /* Получить кадр из очереди */
  МаркироватьКадр(fr, α_c); /* Назначить вычисленный номер текущему кадру */
    Пока (очередь ≠ 0 ) {
      ПолучитьКадр (fr);
      α_c = α_c + 1;
      МаркироватьКадр(fr, α_c);
    }
    α_c = α_c + lost α; /* Вычислить порядковый номер следующего кадра */
}
```

Условия синхронизации двух потоков

Реализация любой схемы синхронизации налагает ряд ограничений на поток. Приведем примеры таких ограничений [2]:

1) кадры с одинаковыми номерами должны воспроизводиться одновременно;

2) разница во времени между основным и вспомогательным кадрами должна быть меньше допустимой разницы между потоками.

Рассмотрим два потока: один основной, другой вспомогательный. Пусть аудиопоток будет основным. Всякий раз, когда видеокадр должен отобразиться, мы вычисляем порядковый номер аудиокадра, который идеально подходит для воспроизведения. Если порядковый номер текущего аудиокадра совпадает с вычисленным, то кадр воспроизводится немедленно. Если номер текущего аудиокадра меньше вычисленного, то видеокадр ожидает. Если же номер аудиокадра больше вычисленного, то видеокадр отбрасывается.

Вычислим порядковые номера кадров α_i основного потока, которые должны воспроизводиться во время отображения кадра β_j вспомогательного потока. Следует отметить, что во время отображения одного кадра вспомогательного потока может быть отображено больше одного кадра основного потока, но при старте кадра β_j может отображаться только один кадр α_i. Тогда имеем:

$$\alpha_i = \left\lfloor \frac{t - t_{0_\alpha}}{d_\alpha} \right\rfloor, \qquad (2)$$

$$\beta_j = \left\lfloor \frac{t - t_{0_\beta}}{d_\beta} \right\rfloor. \quad (3)$$

Из (2) и (3) получаем:

$$\beta_j \le \frac{t - t_{0_\beta}}{d_\beta} < \beta_j + 1, \quad (4)$$

$$\frac{t - t_{0_\alpha}}{d_\alpha} - 1 < \alpha_i \le \frac{t - t_{0_\alpha}}{d_\alpha}. \quad (5)$$

Заменив t из (4) и подставив в (5), получаем:

$$\beta_j \frac{d_\beta}{d_\alpha} + \frac{t_{0_\beta} - t_{0_\alpha}}{d_\alpha} - 1 < \alpha_i < \beta_j \frac{d_\beta}{d_\alpha} + \frac{t_{0_\beta} - t_{0_\alpha}}{d_\beta} + \frac{d_\beta}{d_\alpha}. \quad (6)$$

Введем следующие подстановки:

$$D = \frac{d_\beta}{d_\alpha}, \quad T = \frac{t_{0_\beta} - t_{0_\alpha}}{d_\alpha}. \quad (7)$$

Так как α_i – целое значение, получаем следующую зависимость для кадров α_i основного потока, которые должны воспроизводиться в течение отображения кадра β_j вспомогательного потока:

$$\alpha_i \in \begin{cases} \lfloor \beta_j D + T,\ \beta_j D + T + D - 1 \rfloor, & \text{если } D, T \in Z \\ \lceil \beta_j D + T - 1 \rceil,\ \lfloor \beta_j D + T + D \rfloor, & \text{иначе.} \end{cases} \quad (8)$$

Выражение (8) позволяет получить порядковые номера кадров основного потока, которые должны были бы воспроизводиться во время отображения кадра β_j вспомогательного потока. Для нахождения номера кадра основного потока, во время которого кадр β_j должен стартовать, среди кадров, полученных в соответствии с (8), возьмем кадр с наименьшим порядковым номером. Чтобы обеспечить максимально допустимое расхождение двух потоков, вычислим толерантность потоков $t_{\alpha\beta}$:

$$t_{\alpha\beta} = \begin{cases} \frac{A_{\alpha\beta}}{d_\alpha} - 1, \text{если} \frac{A_{\alpha\beta}}{d_\alpha} \in Z, \\ \left\lfloor \frac{A_{\alpha\beta}}{d_\alpha} \right\rfloor, \text{иначе.} \end{cases} \quad (9)$$

Поэтому кадр β_j может стартовать, если порядковый номер текущего кадра основного потока α_{play} удовлетворяет следующему условию:

$$\alpha_i - t_{\alpha\beta} \le \alpha_{play} \le \alpha_i + t_{\alpha\beta}, \quad (10)$$

где α_i – значения, полученные из (8).

Алгоритм синхронизации одного аудио- и одного видеопотоков

Рассмотрим алгоритм, который учитывает изменение времени отображения видеокадра. Поток видео будет вспомогательным и будет синхронизироваться с аудиопотоком. Оба потока являются непрерывными.

Алгоритм 3

(случай синхронизации аудио-, видеопотоков)

$t_{\text{отображения}}$ = Начальное Значение; /* Инициализировать значение отображения кадра */

Пока (1) {

ПолучитьКадр(v); /* Получить видеокадр v из буфера приложения */

t_d=*РазархивироватьКадр*(v); /* Разархивировать кадр и измерить время */

α_{play}=*ПолучитьТекущийАудиокадр*() $+\left\lfloor \frac{t_{\text{отображения}}}{d_\alpha} \right\rfloor$; /* Вычислить номер аудиокадра, который будет стартовать после отображения видеокадра */

α_i=*ВычислитьНеобходимыйАудио*(v); /* Вычислить номер аудиокадра для отображения, если бы заданный видеокадр стартовал */

Если ($\alpha_i<\alpha_{\text{play}}-t_{\alpha v}$); /* Если видеокадр опоздал, то отбросить его */

Продолжить;

Если (($\alpha_i>\alpha_{\text{play}}+t_{\alpha v}$) /* Если видеокадр опережает, то ожидать*/

Ожидать(($\alpha_i-\alpha_{\text{play}}$)x$d_\alpha$);

t_p=*Отобразить*(v); /* Видеокадр вовремя отобразить и засечь время отображения */

$t_{\text{отображения}}$=0,25t_p+0,75 $t_{\text{отображения}}$; /* Вычислить новую оценку времени отображения */

$v=\left\lfloor \frac{t_d+t_p}{d_v} \right\rfloor$; /* Вычислить порядковый номер следующего видеокадра */

}

Синхронизация n непрерывных потоков

Ранее описанный алгоритм синхронизации обобщим на случай n потоков.

При решении задачи синхронизации n потоков необходимо:

1) найти допустимые значения асинхронности $A_{\alpha\beta}$ между двумя любыми видами потоков (между двумя аудиопотоками, между аудио- и видеопотоками, между двумя видеопотоками);

2) разработать алгоритм смешивания аудиопотоков;

3) разработать алгоритм синхронизации n потоков.

Опишем каждый из названных пунктов.

1) Эмпирически было получено, что время ожидания, на границе которого потоки считались синхронизированными, составляет примерно 0,5 с.

2) В случае двух или более аудиопотоков предлагается следующая стратегия. Предположим, что потоки имеют одинаковую длину кадров. Согласно отношениям (8) и (9), кадр подчиненного аудиопотока вычисляет порядковый номер кадра основного потока, который должен быть отображен в случае, если кадр подчиненного потока стартовал (α_i) бы при толерантности потоков $t_{\alpha\alpha}$. Если порядковый номер текущего аудиокадра основного потока меньше α_i, то подчиненный поток ожидает кадр основного потока, порядковый номер которого равен α_i. Если порядковый номер текущего кадра основного потока находится в интервале $[\alpha_i, \alpha_i+t_{\alpha\alpha}]$, то кадр подчиненного потока отображается вместе со следующим кадром основного потока. В противном случае кадр подчиненного потока отбрасывается.

3) Алгоритм синхронизации n непрерывных потоков можно описать следующим образом: аудиопотоки «смешиваются» по схеме, описанной в предыдущем пункте, а каждый видеопоток синхронизируется с результирующим аудиопотоком в соответствии с алгоритмом 3.

Синхронизация потоков в распределенной среде

В описанном алгоритме 3 синхронизации n непрерывных потоков один источник. Покажем, как данный алгоритм может быть обобщен для случая распределенной среды. Согласно условию синхронизации двух потоков приложение должно иметь представление о временной разнице между потоками. Так как потоки могут формироваться различными источниками, то необходимо введение временного отношения между потоками в распределенной среде.

Временной интервал делится на дискретные блоки, которые будем называть временными кадрами [3]. На каждой станции есть локальный счетчик, показания которого увеличиваются при старте каждого нового кадра. Одна станция делает групповую рассылку всем остальным станциям сообщения «СТАРТ». Станция при получении данного сообщения принимает время его получения за время «общего старта». Во избежание потерь в сети и задержек при передаче пошлем n таких сообщений через равные промежутки времени (σ). При получении i-го сообщения «СТАРТ» в момент времени T_i станция вычисляет время начала «общего старта» путем вычисления минимума (T_0, $T_1-\sigma$, $T_2-2\sigma$,... $T_i-i\sigma$). (В данном случае мы игнорируем задержку распространения сигнала. Такой подход допустим в локальных сетях. При проведении экспериментальных исследований задержка распространения сигнала между конечными устройствами не превышала 5 мс, что значительно меньше продолжительности аудио- и видеокадра.)

Определим через T_0 время старта. Каждая станция будет хранить «виртуальные часы», которые стартуют в момент T_0 и изменяются через каждые Δ реальных временных интервалов. Общее время можно рассматривать как поток кадров длиной Δ, который воспроизводится на каждой станции и стартует независимо. В этом случае используем термин: *поток синхронизации – время* Δ.

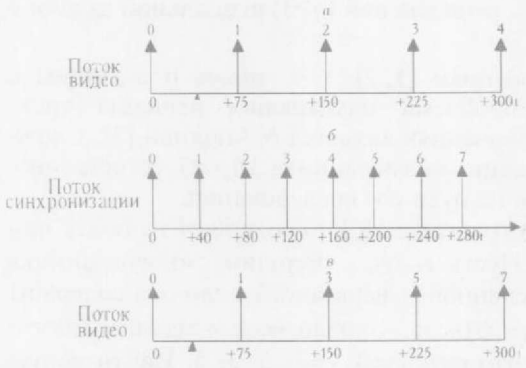
После старта потока синхронизации все события в системе связаны с ним. Для этого порядковые номера кадров во всех потоках выражаются через порядковые номера кадров потока синхронизации. С учетом этого необходимо внести изменения в процедуру назначения порядковых номеров.

После вычисления порядкового номера α_i по формуле 6 дополнительно вычисляется α_i:

$$\alpha_i = \left\lfloor \frac{\alpha_1, d_\alpha}{\Delta} \right\rfloor. \qquad (11)$$

Константа D из соотношения (7) меняется на $D = \frac{d_\Delta}{d_\Delta} = 1$.

На рис. *а* показаны порядковые номера кадров, назначенные потоку видео в соответствии с алгоритмом 1. Продолжительность кадра потока видео 75 мс. На рис. *б* показан поток синхронизации с продолжительностью кадра 40 мс, а на рис. *в* – порядковые номера потока видео, выраженные через порядковые номера потока синхронизации, описанного формулой (11). Первый кадр имеет номер 0, второй кадр – порядковый номер 1, поэтому ему назначается $\left\lfloor \frac{1x75}{40} \right\rfloor = 1$. Третий кадр имеет порядковый номер 2, ему назначается номер $\left\lfloor \frac{2x75}{40} \right\rfloor = 3$ и т. д.



Порядковые номера потока видео в соответствии с алгоритмом 1

При использовании потока синхронизации снимаются ограничения, накладываемые на продолжительность кадра основного потока, так как данная величина может изменяться приложением.

Таким образом, для синхронизации n потоков предлагается следующий алгоритм:

1) дать старт потоку синхронизации на каждой станции, используя предложенный механизм;

2) назначить порядковые номера кадров, используя отношение (11);

3) синхронизировать потоки в соответствии с алгоритмом 3.

Отличие предложенного алгоритма от известных состоит в том, что он позволяет синхронизировать непрерывные мультимедиапотоки в распределенной среде при наличии n независимых источников потоков и разной длине аудио-, видеокадров.

1. Stoica E., Abdel-Wahab H., Maly K. Synchronization of multimedia streams in distributed environment. Norfolk, 1997.

2. Cen S., Pu C., Staehli R. A distributed real-time MPEG video audio player. New Hapmshire, 1995.

3. Sun Microsystems // SunVideo 1.0 User's guide. Oct. 1993.

4. Chen H-Y., Wu J-L. // IEEE Journal of Selected Areas in Communications. 1996. Vol. 14. № 1. P. 238.

5. Stoica I., Zhang H. LIRA: A model for service differentiation in the internet. In proceedings of NOSSDAV'98. London, 1998.

6. Листопад Н.И., Копачев А.Г. // Вестн. связи. 1998. № 6. С. 23.

7. Quemada J. ISABEL: looking to the future // EXPERT Winter Workshop.Villars, 1999.

Поступила в редакцию 11.01 2000.

УДК 519.10

М.К. КРАВЦОВ, Е.В. ЛУКШИН

К ОЦЕНКЕ ЧИСЛА НЕЦЕЛОЧИСЛЕННЫХ ВЕРШИН МНОГОГРАННИКА МНОГОИНДЕКСНОЙ АКСИАЛЬНОЙ ЗАДАЧИ О НАЗНАЧЕНИЯХ

Explicit formula for determination of 7-non-integer vertices number of the polytope of three-axial assignment problem, i.e. the vertices with the number of fractional components being equal 7, is obtained.

Хорошо известно [1], что многогранник $M(2,n)=\{x=\|x_{ij}\|_n : x_{ij}\geq 0 \ \forall (i,j)\in N_n\times N_n, \sum_{i=1}^n x_{ij}=1 \ \forall j\in N_n, \ \sum_{j=1}^n x_{ij}=1 \ \forall i\in N_n\}$ задачи о назначениях является целочисленным. Здесь $N_n=\{1, 2, ..., n\}$. В отличие от $M(2,n)$ многогранник $M(p,n)=\{x=\left\|x_{i_1\dots i_p}\right\|_n : x_{i_1\dots i_p}\geq 0 \ \ \forall (i_1, ..., i_p)\in N_n^p, \ \sum_{i_1=1}^n \dots \sum_{i_{s-1}=1}^n \sum_{i_{s+1}=1}^n \dots \sum_{i_p=1}^n x_{i_1\dots i_p}=1 \ \ \forall i_s\in N_n,$