Buffer Optimization in Realtime Media Servers Using Jitter-constrained Periodic Streams

Claude-Joachim Hamann, Andreas Märcz, Klaus Meyer-Wegener
Technische Universität Dresden, Fakultät Informatik, 01062 Dresden, Germany
[ch4|am9|kmw]@inf.tu-dresden.de

Abstract

Media servers that provide data independence need converters to adapt the data to the client's specific representation. For timed media objects, the conversion must eventually be performed in realtime. This requires a careful planning of resources. In addition to processors, SCSI devices, busses etc. buffers have to be allocated. They are needed to cope with the jitter in processing times of subsequent converters. This papers presents a model that describes the buffer accesses in only a few parameters and still allows to derive the minimum buffer size.

1. Introduction

Media servers today offer either data independence or realtime support, but not both [4]. *Data independence* means that applications can access data without knowing the storage format. For media data, this additionally means that some parameters like resolution, presentation of elements (e.g. frames), and rate can be different in the playout. The benefit of this indirection is that heterogeneous sets of clients can reference a single copy of the media data object. *Realtime* on the other hand means that the system guarantees a certain behavior. For a media server, a predefined quality of service is to be expected, namely a frame rate and a resolution in the case of video. This requires resource reservation and runtime support like scheduling.

In the AMOS project at GMD IPSI in Darmstadt, Germany, an object-oriented database system has been enhanced to support realtime [8]. However, while internal processes have been optimized to a large extent, no guarantees can be given. Therefore a media server with data independence and realtime playout functions based on the planning and scheduling of processing times and buffer sizes is still not at hand.

The goal of the project Memo.real at the Dresden University of Technology is to design a media server that combines data independence with realtime support. Since data independence can only be achieved with the help of *converters* (or mappings), these converters must be executed in realtime. Given that their temporal behavior is known, an important task (next to scheduling) is to connect them through buffers. The idea of this work is that the streams flowing through the converters and the buffers can be modeled as so-called *jitter-constrained periodic event streams* [6]. While this modeling allows proper scheduling of the converters by a realtime operating system like DROPS [7], it also helps to calculate the minimum buffer sizes. The latter aspect is described in this paper.

2. System Architecture

Media servers offer a set of operations to access the media data objects. This set can be rather large, and it varies from system to system. Standardization has begun in the context of SQL:1999 under the name of SQL/MM [9], but it is not finished yet. In any case, there will only be a few operations that are relevant for realtime support, namely playout and recording. For this so-called streaming, proprietary protocols and thus specific clients must be used in most cases. A media server should instead be open for a large variety of different clients.

Streaming operations that produce a variety of formats and support many protocols need converters in the server. The term “converter” can in fact mean many things. Converters can code or decode media data, they can scale or filter objects, they can change the color space, and so on. In order to generate the output format requested by the client from the storage format, it can be necessary to use a *chain* (or more general, a directed acyclic graph) of converters. This idea is not new; it has been published by Dingeldein [3] and Candan [2], and it has been used in Microsoft's DirectShow [13]. A more recent approach is described by Marder [12]. They all, however, restrict the discussion to the function and do not consider execution time.

The converters in such a chain will have different processing times for the data elements (*quanta* [4]) that stream through them. Hence, buffers are needed in between. The starting point of the chain is usually the file system of the operating system. Even if the operating system supports realtime, it will deliver the quanta (i.e. blocks) with some jitter. So an additional buffer is needed at the beginning of the chain. At the end, the network accepts packets for delivery to the client. Again a buffer is needed, since the network will not accept packets at any time. The situation is sketched in Fig. 1. Assuming that the processing time of the converters in the chain is known, the question now is how large the buffers must be.

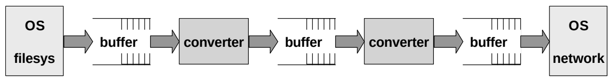

Figure 1. Chain of converters.

The converters read quanta as input and produce other quanta as output. Typical examples of such quanta are the frames of a video. To organize the chain of converters, some information must be available about the size and the timing of the quanta. It is not sufficient to know the mean value of both, that is, average frame size and period. The *worst case* of both, however, can be used in planning, as it is done in realtime systems. But this usually leads to very bad resource utilization. In particular, the allocated buffers are much too large. This kind of planning could also lead to a rejection of the whole chain, if the resources needed for the worst case are not available.

On the other extreme, the media object stored can be analysed and statistical information can be recorded for each quantum. This will be called a "*trace*" in the following. Of course, it is rather expensive.

In between, there is the model of a *jitter-constrained periodic event stream* [6]. It provides more information than mean or worst-case, but much less than a trace. Regarding *time*, the events are the reads and writes of the converters. In principle, these are periodic processes. Due to variations in processing time, there may be a jitter. Events may be too early or too late. This jitter, however, is constrained, so it must not pass beyond a boundary. The model is introduced in chapter 3. Please note that it can also be used for "infinite" streams (e.g. live video), while traces can not.

The idea of period and constrained jitter can in fact be extended to the *size* of the quanta; for details see chapter 3 below. This extends the model with a second stream. The two streams can be introduced into the planning process stepwise as depicted in Fig. 2. Starting from a worst-case analysis for both at the top, either processing time or quantum size can be described by a stream. This is useful if time, or size, resp., does *not* vary. For instance, an uncompressed video can have fixed frame size, and a converter can be programmed to read from a buffer exactly at the beginning of each period. Even if the worst case is used to cope an unknown jitter, a single stream is still better than the worst-case analysis for both time and size. At the bottom, there is the model that uses both streams. This is most complicated, but it should yield the best result among the four models of Fig. 2. Of course, it cannot be as good as a trace.

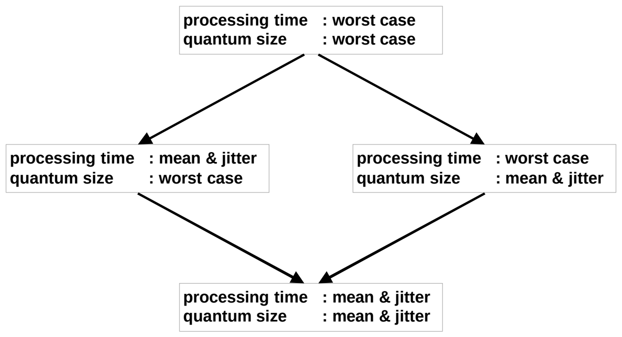
processing time : worst case
quantum size : worst case
processing time : mean & jitter
quantum size : worst case
processing time : worst case
quantum size : mean & jitter
processing time : mean & jitter
quantum size : mean & jitter

Figure 2. Planning models for converters

3. Buffer sizes

To actually perform the planning, the streams must be discussed in more detail. The situation shown in Fig. 1 can be regarded as a sequence of converters with buffers between them, each of which represents a typical producer-consumer problem: A producer P starts at some point in time t_0 and periodically, at intervals of length T, generates quanta Q_i of varying size. It needs varying time to do so (see Fig. 3).

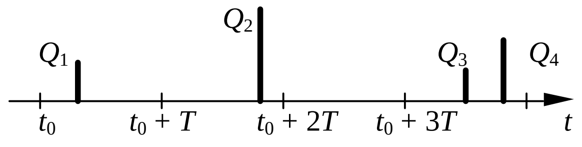

Figure 3. Generation of quanta

The quanta are stored in a buffer B. Consumer C reads quanta from the buffer periodically, too, either in the same size as stored or in a different size. The time of removal (i.e. the time when the consumer releases the buffer allocated to the consumed data) may vary, too. The consumer acts as a producer for the next link in the chain. In any case, the consumer must not starve, i.e. at least those quanta must be available in B that the consumer needs in the next period. This makes it necessary to identify two values: the lead time t_{lead} of the producer and the minimum size B_{min} of the buffer that can hold all data. Ideally, a trace is available to calculate the values. It contains the complete information of all quantum sizes plus corresponding production and processing times. However, this information is not available in all cases- obtaining it might be too expensive.

On the other hand, planning on the basis of worst-case assumptions alone (that is, maximum quantum size and maximum production time) will lead to bad resource utilization. To cope with this situation, the model of jitter-constrained periodic streams has been introduced in [6]. The variation in time of events in a periodic sequence is constrained: Events are supposed to happen with a constant distance T in time, but are allowed to vary in fixed boundaries, namely not earlier than t before and not later than t' after the beginning of each period. Also, the minimum distance D in time between two subsequent events is given. This allows to calculate a number of important characteristics for the streams, for example the maximum burst size, minimum and maximum burst distance, and minimum buffer size to avoid loss of data [6]. In fact, these characteristics do not depend on t and t' individually, but only on the sum of the two. Finally, to combine two of these streams, a starting point is needed as an additional parameter. This leads to the following definition (see also Fig. 4).

Definition. Given T, D, t, t_0 with

$T > 0$	average event distance (length of period),
$0 \le D \le T$	minimum distance,
$t \ge 0$	maximum lateness (deviation from beginning of period),
$t_0 \in \mathbb{R}$	starting point.

Then a jitter-constrained periodic stream is a sequence $(a_i)_{i=0,1,2,...}$ with

$$a_i \in [iT, iT + t] \quad \text{and} \quad a_{i+1} - a_i \ge D \quad \forall i \in \mathbb{N}.$$

As a simplification, such a stream will be identified by the tuple (T, D, t, t_0) in the following.

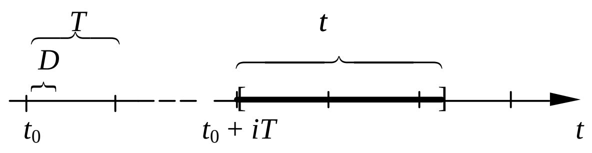

Figure 4. Jitter-constrained periodic stream

Experience indicates that the jitter of size and the jitter of processing time are independent. Now, the idea that goes beyond other approaches (e.g. [1], [15]) and the worst-case analysis is to model the sequence of quantum sizes also by such a stream (S, M, s, s_0) with

$S > 0$	average quantum size,
$0 \le M \le S$	minimum quantum size,
$s \ge 0$	maximum deviation from accumulated quantum size,
$s_0 \in \mathbb{Z}$	initial value.

The following example illustrates the idea (see Tab. 1). A process P generates at time t_i 8 quanta of size s_i, $i = 0,...,7$.

t_i/s	1,6	2,4	2,8	3,2	6	6,4	6,8	8,5
s_i	1	2	0	1	4	4	3	1

Table 1. Trace $s_i = P(t_i)$ to generate 8 quanta

Two jitter-constraint periodic streams are assigned to this process: a "time stream" (see Fig. 5a)

$$P_T = (T = 1\text{s},\ D = 0{,}4s,\ t = 2s,\ t_0 = 0s)$$

and a “size stream” (see Fig. 5b)

$$P_S = (S = 2,\ M = 0,\ s = 5,\ s_0 = -1).$$

(The minimum values D and M are irrelevant in the following).

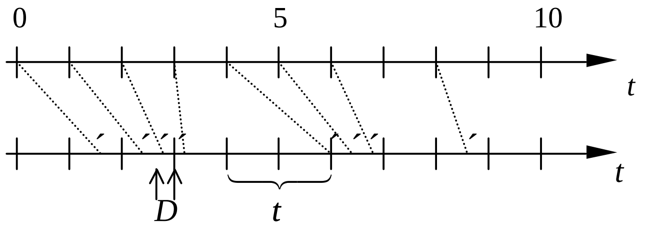

Figure 5a. Time stream of producer *P*

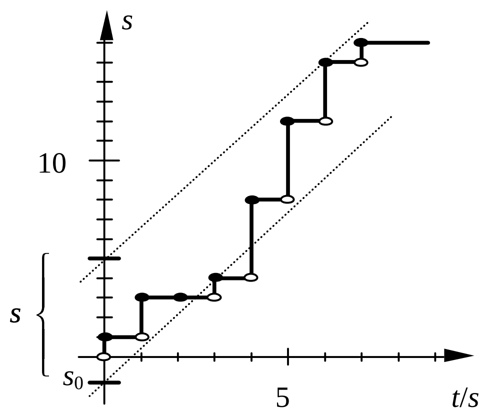

Figure 5b. Size stream of producer *P*

To calculate the lead time t_{lead} and the needed buffer B_{min} , it is assumed that the consumer C takes from the buffer the ith quanta with size unchanged at exactly the beginning of the ith period, and releases the buffer immediately. Using the traces $P(t)$ and $C(t)$ of producer P and consumer C, the lead time t_{lead} equals the largest difference between $P(t)$ and $C(t)$ in direction of the abscissa (t–axis, see Fig. 6). Then B_{min} is the largest difference between $P(t - t_{lead})$ and $C(t)$ (for more details see [5]).

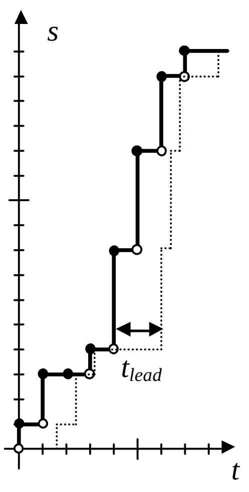

Figure 6. Calculating t_{lead}

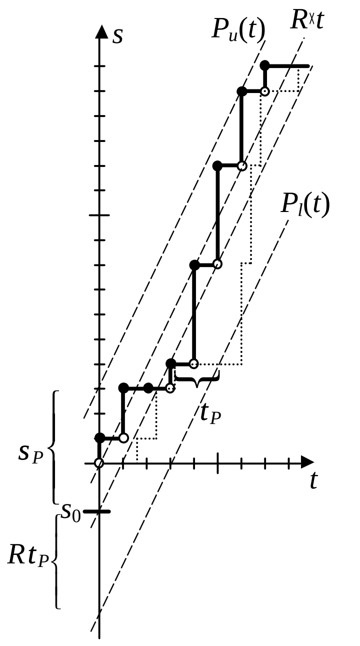

Figure 7. Jitter-constrained stream instead of trace

Assuming now that P and C are jitter-constrained periodic streams just given by

$$P_T = (T_P,\ D_P,\ t_P,\ t_{0,P}),$$
$$P_S = (S_P,\ M_P,\ s_P,\ s_{0,P}),$$
$$C_T = (T_C,\ D_C,\ t_C,\ t_{0,C}),$$
$$C_S = (S_C,\ M_C,\ s_C,\ s_{0,C}).$$

The only restriction is that

$$R = \frac{S_P}{T_P} = \frac{S_C}{T_C} = \text{const.},$$

i.e. both processes work with the same average data rate. (This condition must hold for all pairs of buffer processes in a converter chain like the one in Fig. 1). Then the traces $P(t)$ and $C(t)$ must be substituted by linear functions $P_u(t)$, $P_l(t)$, $C_u(t)$, and $C_l(t)$, resp., resulting from the parameters of P_T , ... , C_S and bounding the traces upwards and downwards. It holds for the upper bounds

$$P_u(t) = Rt + s_P + s_{0,P}$$
$$C_u(t) = Rt + s_C + s_{0,C}$$

and for the lower bounds

$$P_l(t) = Rt + s_{0,P} - Rt_P,$$
$$C_l(t) = Rt + s_{0,C} - Rt_C.$$

Now t_{lead} is given by (see Fig. 7)

$$P_l(t - t_{lead}) = C_u(t).$$

The calculation of B_{min} is done in the same manner as described above. It follows :

$$t_{lead} = \frac{s_C}{R} + t_P \ ,$$

$$B_{\min} = \lceil (t_C + t_P) \cdot R + s_P + s_C - s_0 \rceil.$$

A concluding example illustrates the approach. A converter produces a stream of 400 quanta in total with a period of length $T_P = 1s$. The time needed to produce a quantum varies from $D_P = 0.341s$ to $1.012s$ with a mean of $0.751s$. The accumulated beginnings of periods are exceeded by at most a t_P of $15.501s$. The average size of quanta is 3362.4 Bytes. Also, s_P = 289,391 Bytes and $s_{0,P}$ = -37,478 Bytes (see Fig. 7). The consumer C takes the quanta from the buffer with size unchanged (that is, $P_S = C_S$) and releases the buffer space after $1s$ at the latest. Then the producer is to start $t_{lead} = 16.501s$ before the consumer wants to start, and a buffer size B_{min} of 382,373 Bytes guarantees a data transfer without loss of data.

4. Conclusions and Outlook

The buffer sizes in chains of converters can now be derived from a small set of parameters that characterize a stream of reads or writes. These parameters are period, jitter (lateness), and starting point for time, and average size, maximum deviation from accumulated size, and prefetch for size. This leads to better buffer allocation than in worst-case analysis.

In the particular situation that the consumer always reads quanta in exactly the same size as the producer has written them, further optimization is possible. The model of jitter-constrained streams can also be used to plan the resources needed by the converters. [11]

Once media servers in isolation have been enhanced with data independence, they can be used in cooperation. Projects like Imos [14] and MacWrap [10] demonstrate how media servers can be integrated in database management systems.

5. References

[1] Anderson, D. J., “Metascheduling for Continuous Media“, *ACM TOCS*, vol. 11, no. 3, Aug. 1993, pp. 226-252.

[2] Candan, K.S., V.S. Subrahmanian, and P.V. Rangan, “Towards a Theory of Collaborative Multimedia”, in: *IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS) 1996*.

[3] Dingeldein, D., “Multimedia interactions and how they can be realized”, in: A.A. Rodriguez and J. Maitan (eds.), *Proc. SPIE 2417* (1995), Photonics West Symposium, Multimedia Computing and Networking (San José, Febr. 6-10, 1995), pp. 46-53.

[4] Gemmel, D.J., H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe, “Multimedia Storage Servers: A Tutorial”, *IEEE Computer,* vol. 28, no. 5, May 1995, pp. 40-49.

[5] Hamann, C.-J., and L. Reuther, „Pufferdimensionierung für schwankungsbeschränkte Ströme in DROPS“, in: *Proc. MMB'99*, 10. GI/ITG-Fachtagung Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen (Trier, Germany, Sept. 22-24, 1999), VDE-Verlag.

[6] Hamann, C.-J., “On the Quantative Specification of Jitter Constrained Periodic Streams”, in: *Proc. MASCOTS'97,* 5th Int. Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (Haifa, Israel, Jan. 1997).

[7] Härtig, H., R. Baumgartl, M. Borriss, C.-J. Hamann, M. Hohmuth, F. Mehnert, L. Reuther, S. Schönberg, and J. Wolter, “DROPS - OS Support for Distributed Multimedia Applications”, in: *Proc. 8th ACM SIGOPS European Workshop* (Sintra, Portugal, Sept. 7-10, 1998).

[8] Hollfelder, S., F. Schmidt, M. Hemmje, and K. Aberer, “Transparent Integration of Continuous Media Support into a Multimedia DBMS”, in: *Proc. Int. Workshop on Issues and Applications of Database Technology (IADT'98).*

[9] *Information Technology – Database Languages – SQL Multimedia and Application Packages – Part 1: Framework,* ISO/IEC CD 13249-1.

[10] Lindner, W., and A. Heuer, „Das MacWrap Projekt – Nutzung eines objekt-relationalen DBMS für ein offenes Multimedia-Datenbanksystem“, in: *Proc. 12. GI-Workshop Grundlagen von Datenbanken* (Plön, 13.-16. Juni 2000).

[11] Märcz, A., „Entwurf eines Modells zur Konverterbeschreibung“, Diplomarbeit, Dresden University of Technology, Department of Computer Science, 2000.

[12] Marder, U., “VirtualMedia: Making Multimedia Databases Fit for World-wide Access**”**, extended abstract, in: *Proc. EDBT 2000 Ph. D. Workshop* (Konstanz, Germany, March 31 - April 1, 2000), pp. 47-50.

[13] Microsoft, “Streaming Media Technology”, White Paper, 1998.

[14] Süß, H., “A Flexible Architecture for the Integration of Media Servers and Databases”, in: *Advances in Multimedia Information Systems: Proc. 4th Int. Workshop MIS'98,* Springer, 1998, LNCS vol. 1508, pp. 174-184.

[15] Want, R., A. Hopper, V. Falcão, and J. Gibbons, “The Active Batch Location System”, *ACM TOIS*, vol. 10, no. 1, Jan. 1992, pp. 91-102.