Real Time Clock Skew Estimation over Network Delays

Dominique Fober, Yann Orlarey, Stéphane Letz

GRAME
Laboratoire de recherche en informatique musicale
Technical report TR-050601

Real Time Clock Skew Estimation over Network Delays

Dominique Fober, Yann Orlarey, Stéphane Letz
{fober,orlarey,letz}grame.fr

June 2005

Grame
Centre National de Création Musicale
9, rue du Garet BP 1185
FR - 69202 Lyon Cedex 01

Abstract

Clock skew is one of the significant issues of real-time networking. We have previously proposed a new algorithm for clock skew detection that operates one a one-way measurement, simply based on time stamped packets. This algorithm was initially targeted to Internet transport context. It has been showed that it is also suitable to low latency transport conditions using an adequate parameters set. This report presents improvements to the algorithm as well as as well as parameters characterization, along with the corresponding performance measurements.

1 Introduction

Works concerning clocks synchronization are numerous: whether developed in the context of distributed systems or to provide a worldwide accurate common time, clock synchronization algorithms have reached a mature level of performance. However, there are many cases where these algorithms are not applicable: peer to peer networking for example cannot benefit of the distributed systems methods, one-way transmissions cannot support protocols like NTP (Network Time Protocol) [7]. There are also many applications that only needs to maintain a common time rate without concern about the clock offsets: Internet delays measurement for example have to separate the clock skew from actual transport delay; real-time multimedia streaming applications have to maintain a synchronized rate for data exchange.

However, there are a few works focusing on the clock skew estimation only. In network performance measurements domain, several algorithms have been developed to remove the clock skew from one-way delay measurements [9] [11]. The most accurate of them is the Moon linear programming algorithm [9]. Almost of these algorithms are not applicable in real-time: the skew computation becomes rapidly prohibitive when the time increases. Only the linear regression algorithm supports real-time with a constant cost but since the deferred time version is not robust in presence of outliers, it performs even worse in real-time.

In the multimedia domain, the problem is approached using techniques similar to NTP [2], or frequently ignored because it is avoided by talkspurt based systems, or naturally solved by adaptive playout mechanisms. This latter solution is however not satisfying since the skew compensation produces significant distortions.

Many techniques used to manage time are based on Phase Locked Loop like in NTP [8] or on Delay Locked Loop to filter time like in the jack audio system [1]. We have previously developed an algorithm named EPTMA (which stands for Exponential Peak Tolerant Midpoint Algorithm) [4], mostly inspired by distributed systems CSA (Clock Synchronization Algorithms) but that is not so distant to NTP.

EPTMA operates on a one-way packets transmission and produces a real-time estimation of the receiver clock deviation in regard of the sender clock. It only relies on time stamped messages and may therefore operate over existing protocols like RTP [12]. It has been first designed for the Internet but also performs very efficiently on a low latency network [6] using an adequate parameters set.

The idea behind the present work is to improve the algorithm by investigating in several directions: what are the best parameters suitable to a wide range of transport contexts? can adaptive parameters improve the skew estimation? can technics developed in [8] or [1] benefit to the algorithm? this report aims at providing replies to all these questions.

The rest of the paper is organized as follows. Section 2 introduces the skew estimation problem. Section 3 recalls the EPTMA functioning. Section 4 presents the methodology used to assess the algorithm performances. In section 5 we describe our strategy to improve the algorithm and we give the corresponding performances, along with their correlation to the parameters set. We finally conclude in section 6.

2 The skew estimation problem

The figure 1 presents typical latency variations measured during a network audio session. The latency variation is expressed in audio frames. 4 different stations were broadcasting 128 frames audio buffers collected at a 44.100 kHz sampling rate. The underlying network architecture was Ethernet 100 Mb. The constant increase slope drawn on the figure represents the clock skew; it may be viewed as the plot of the difference between the sender and the receiver clocks minus the clocks offset ie:

$\delta(t) = C_s(t) - C_r(t) - (C_s(0) - C_r(0))$ where C_s and C_r represents the sender and receiver clocks.
It has been obtained using the linear programming algorithm developed by Moon [9]. It exhibits a 36 frames skew over the 3 minutes measurement, which represents about a 1 per 220 000 time unit deviation. The problem of the clock skew estimation may be viewed as the problem of the constant slope evaluation.

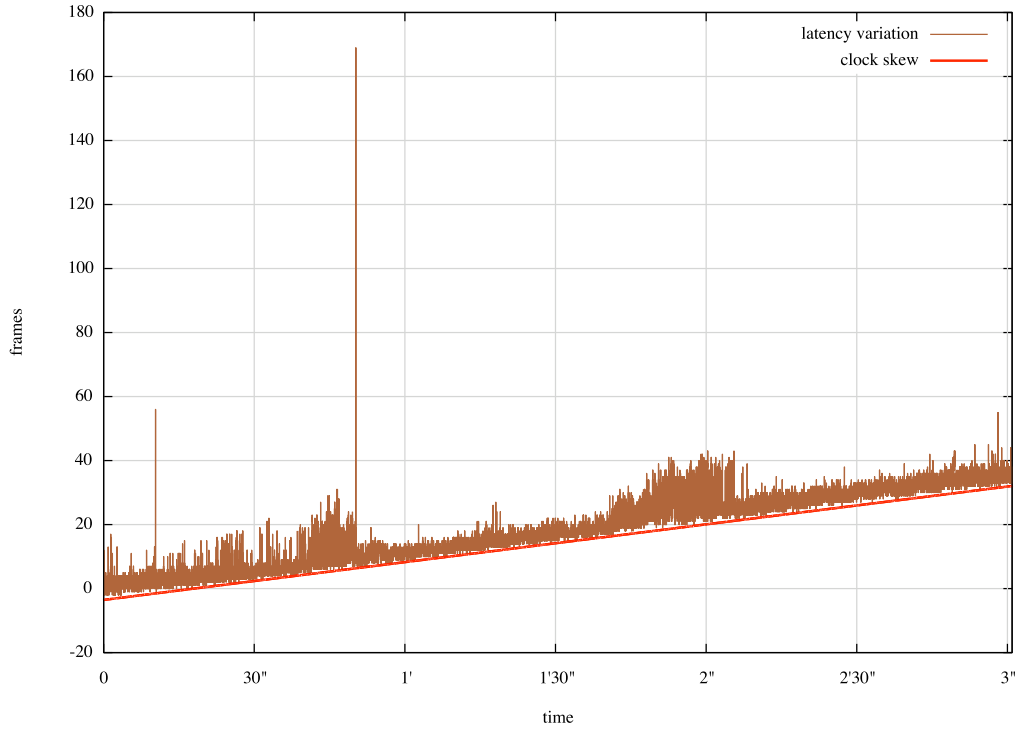

Figure 1: Typical mixed latency variation and clock skew.

Real-time skew estimation is a compromise between accuracy and inertia. Depending on the parameters used, the algorithm may adapt quickly to the deviation slope but also produce distorted values on transient transport changes, but it could also be less sensitive to local transport changes at the expense of additional time to detect the correct slope. The figure 2 clearly illustrates this compromise: the straight

line is the linear estimation previously shown by the figure 1, it represents the optimal detection slope. The first curve below is the output of a responsive estimation but that shows a significant distortion around time 2". The lower curve is a smoothed estimation that avoids almost of the distortion but takes about 20 seconds to reach the correct slope: during this time, the clocks are drifting without any possible correction, leading to potential problems with the audio rendering engine.

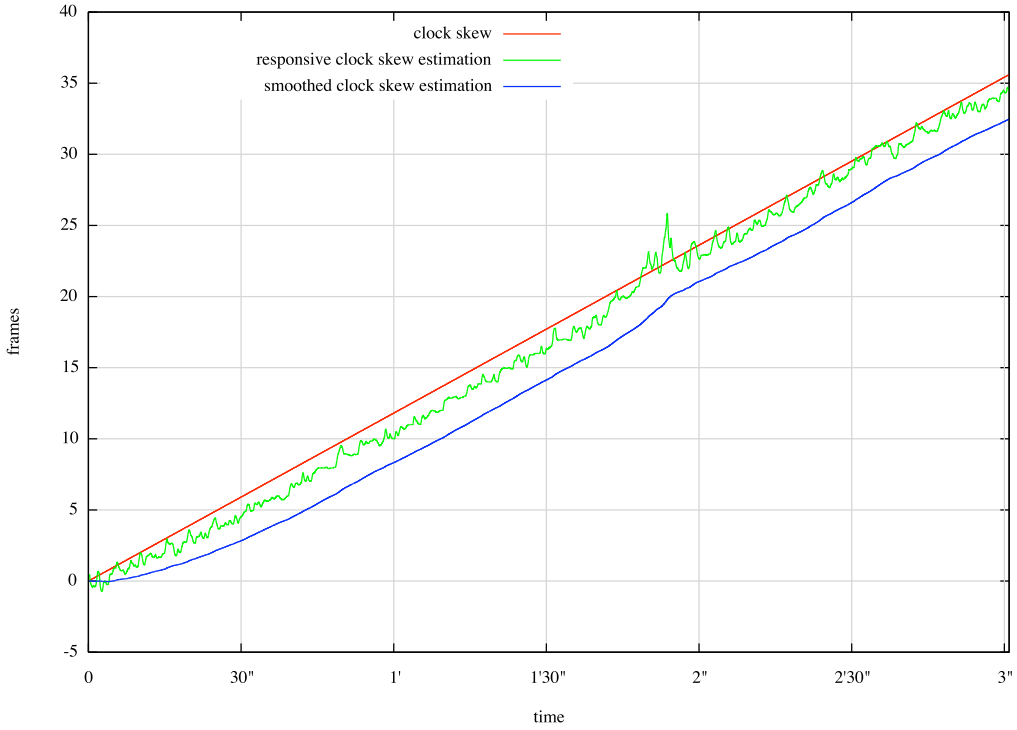

Figure 2: Clock skew estimation compromise.

3 EPTMA

The basic intuitive idea about EPTMA consists in selecting the mean values of the delay variations and to operate a smoothing of these values to obtain the clock skew estimation. The selection process makes use of a sliding window. Let w be the window size and k the number of values retained per window, at a time t, the algorithm computes a sorted delay variation vector as: $\Delta_t^w = [\delta_{t-w} \cdots \delta_i \cdots \delta_t]\,, \quad \delta_i \leqslant \delta_{i+1}$
The highest and lowest values are discarded and the mean delay variation d_m is then:

$$d_m = \frac{1}{k} \sum_{i=(w-k)/2}^{(w+k)/2} \delta_i \qquad 1 \leqslant k \leqslant w \tag{1}$$

At time t, the clock skew estimation s_t is finaly obtained by smoothing the mean delays using exponential averaging:

$$s_t = \alpha d_m^t + (1-\alpha) s_{t-1} \qquad 0 \leqslant \alpha \leqslant 1 \tag{2}$$

where α is the weighting factor.

During the initialization stage (i.e. while the temporal window w isn't complete), the algorithm computes the mean delay values using a parabolic value for k, dynamically computed from the actual available values. Let n be the values count, the parabolic retained values r_n is computed as:

$$r_n = k - \frac{k}{w^2}(n-w)^2 \qquad 0 < n < w \tag{3}$$

4 Methodology

An important point, before presenting the possible improvements of the algorithm, concerns the way to assess these improvements. In this intent, we introduce a method to characterize the skew accuracy and we describe the data set used for assessment.

4.1 Characterization of the skew estimation accuracy

The clock skew estimation is normalized in order to provide a common basis for characterization. Normalization consists in subtracting the optimal skew slope obtained as output of the linear programming algorithm, to the clock skew estimation. This is similar to evaluate clock deviations with perfectly frequency synchronized clocks: there is no skew and the optimal slope line is $f(x) = 0$. Accuracy of the estimation is then characterized by two figures that are directly related to the compromise mentioned in section 2:

- the responsiveness R: corresponds to the time necessary to reach the correct skew slope. The responsiveness is expressed in time unit deviations. It may be viewed as the deviation from the optimal detection slope during the initial stage of the transmission. It depends obviously on the initial transport context.
- the instability I: denotes the estimation curve stability after the initial stage. It is expressed in time units representing the range of the values after the initial stage. An optimal detection slope should converge to $f(x) = n$ where n is the above responsiveness.

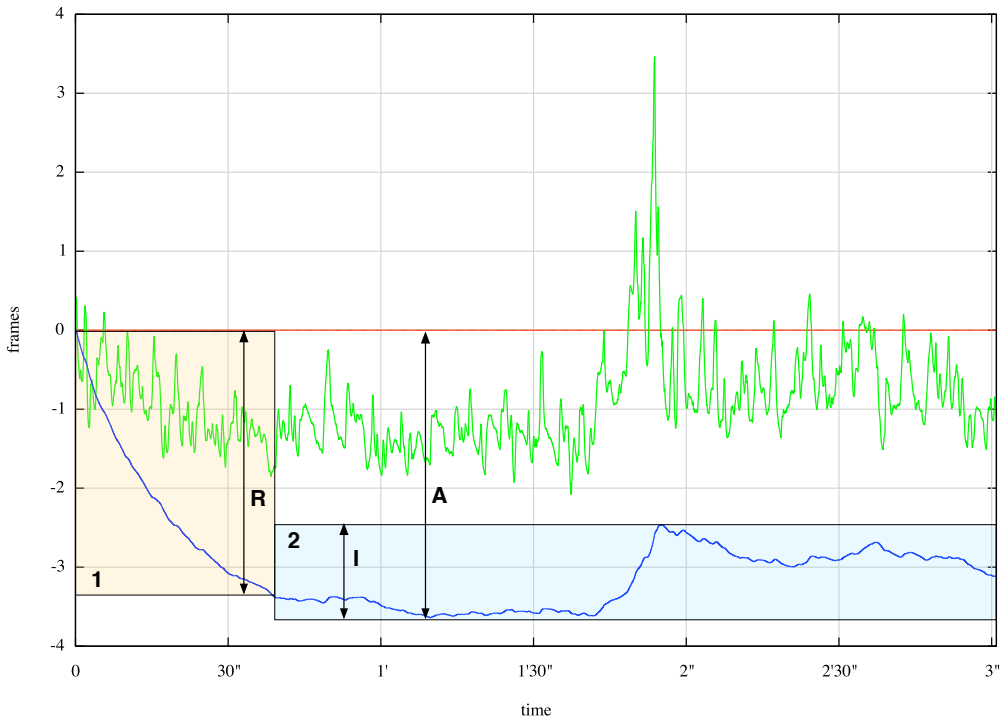

Figure 3: Normalized clock skew detections and corresponding stages.

The figure 3 shows normalized estimation curves (from figure 2) and illustrates their responsiveness and stability. Highlithed areas 1 and 2 correspond respectively to the responsiveness ($\sim$ 2.5 frames) and stability ($\sim$ 1 frame) of the smoothed estimation clock skew. The overall accuracy is about 3.5 frames. Concerning the upper curve, it may be considered as very responsive (no initial stage) but it behaves very badly from accuracy viewpoint.

4.2 Data set

We have selected a set of 8 transport delay variations obtained from real transmissions over several kind of network and using different payloads. To collect the results we used the following methods:

- the unix tool *ping*: it has the advantage to give results with a known clock skew i.e. none
- an audio streaming application supporting several concurrent audio streams
- a MIDI streaming application [5].

Transport characteristics are summarized in table 1. The whole set is presented in appendix.

method	network	connection speed	name
ping	general purpose Ethernet	10 Mb	local 1
-	isolated Ethernet	100 Mb	local 2
-	Internet via adsl	$\geqslant$ 2 Mb	adsl 1 & 2
audio streaming	isolated Ethernet	100 Mb	audio 1 & 2
MIDI streaming	Internet via modem	28.8 kbs	midi 1 & 2

Table 1: Measurements methods and context

4.3 Data set normalization

The data set normalization consists in removing the clock skew, which is computed using the linear programming algorithm. This algorithm gives high quality results over the whole data set. Of course, the measures obtained using *ping* were not normalized since not subject to clock deviation. Normalized data set are flat and the corresponding optimal clock skew estimation should converge to $f(x) = 0$.

Next, measurement of the accuracy is achieved thru the following steps:

- a simulated clock skew varying from -0.003 to +0.003 by 0.001 step is applied to each normalized data,
- for each simulated slope, the skew estimation is computed, which produces a *skew curve* as output,
- this resulting curve is finally normalized back and the accuracy figures are computed as described in section 4.1.

Each operation is repeated over the whole data set.

5 Improving the estimation accuracy

Several directions have been investigated with more or less success, to improve the results produced by the EPTMA algorithm: using the DLL presented in [1] or a second order filter as defined in [3] in place of the exponential averaging, changing the first stage selection policy and exploring the parameters space.

5.1 The selection process

5.1.1 Mid point versus low point averaging

The first operation of the algorithm consists in discarding the highest and lowest delays to produce a mean value from a subset of the medium delays. Results of the measurements show that using the lowest instead of the medium delays produces better estimations, especially in the context of transient latency increase. The equation 1 is thus replaced by the low point averaging d_l computed as:

$$d_l = \frac{1}{k} \sum_{i=1}^{k} \delta_i \qquad 1 \leqslant k \leqslant w \tag{4}$$

The figure 4 shows the increase of accuracy obtained using low point instead mid point averaging with a window size $w = 200$ and $k = 10$, on different delays data. The graphic plots the difference $A_m - A_l$ where A_m and A_l are respectively the mid point and the low point accuracies.

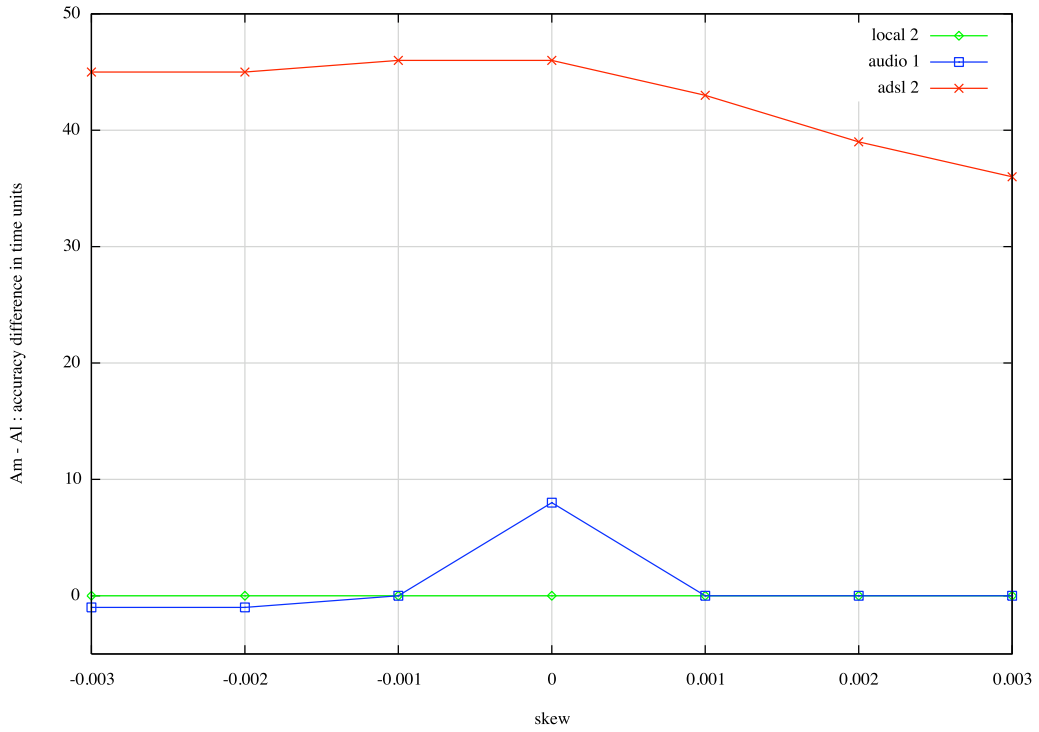

Figure 4: Mid point versus low point comparison: the graphic shows the accuracy difference between the mid and low point averaging.

Results over the whole data set shows that equations 1 and 4 produce equivalent estimations for low latency and stable transport contexts. But in case of delay increase, the benefit of the low point averaging may become very significant.

The intuitive interpretation of this behavior is the following: let's consider that there are *true* and *false* delays. *true* delays are those that represent the minimum (or optimum) transport time, *false* delays are those that include additional latency (mainly due to queuing). The first idea with the selection process comes from the distributed systems that discard *false* information using a strategy proposed by Lamport [10]. But there was a misconception concerning the discarded values: low delays are obviously denoting an information more close to the *true* delay than the medium ones.

5.1.2 Window size and retained values

The window size and the number of retained values have been explored with w varying from 10 to 300 by 10 increment and k from 1 to 30. From the results obtained, it is obvious that a single value retained per window is most of the time the most efficient solution. Therefore equation 4 can be simplified as:

$$d_l = min(\Delta_t^w) \tag{5}$$

Concerning the window size, the larger the window is, the better are the results. In particular, a large window allows to cross a transient high latency period without distortion, provided that the period is smaller than the window size: for example, the accuracy on ADSL1 (see figure 10 in appendix) drops down from 6.8 to 1.2 when the window size increases from 220 to 230, simply because the initial high latency period lasts over 225 points.

However, in the general case there is a size over which there is no significant improvement: for our data set, this window size is 250. But it is noteworthy that for stable low latency delays (such as LOCAL1 and LOCAL2 see figure 11 in appendix), this size drops down to 20.

5.1.3 Initial stage

The EPTMA algorithm dynamically computes the count of retained values according to the available values up to the window size (eq. 3). It appears that a better strategy consists in waiting for the

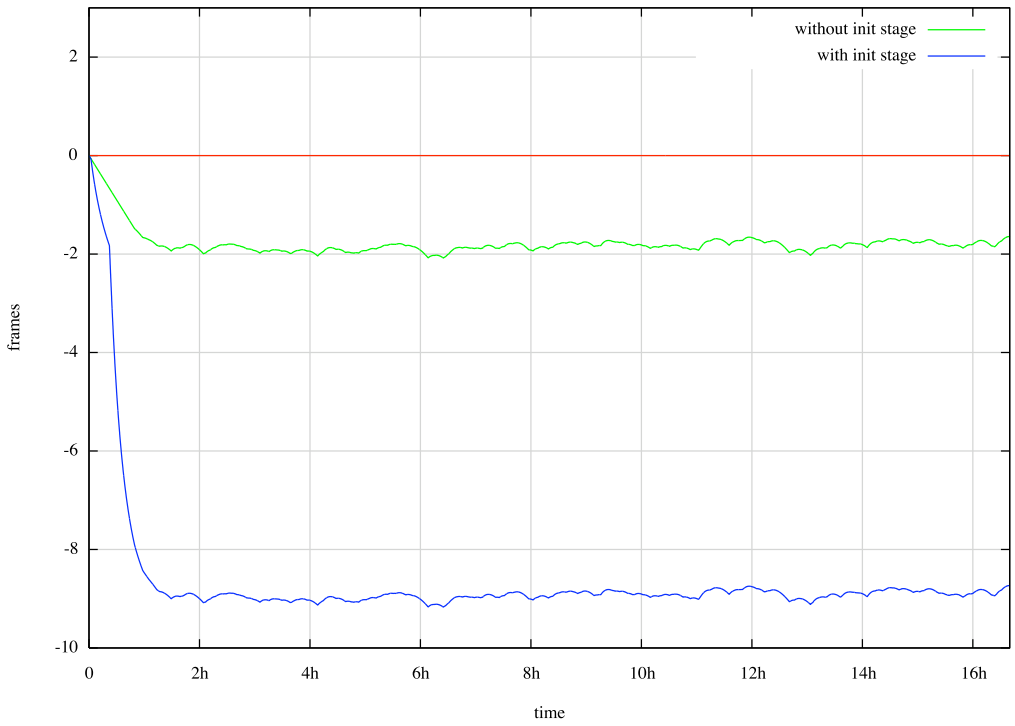

Figure 5: ADSL1 skew estimation with and without initial stage.

window completion before producing the estimation and to normalize this values to zero. The figure 5 illustrates the difference: it corresponds to estimations computed from the ADSL1 data set (see figure 10 in appendix) with a 0.003 skew. Without initial stage, the skew estimation over the first window w is:

$$s(i) = min\left[\delta(1) \cdots \delta(w)\right] \qquad \text{with } 1 \leqslant i \leqslant w$$

and with initialization stage:

$$\begin{aligned} s(i) &= min\left[\delta(1) \cdots \delta(10)\right] && \text{with } 1 \leqslant i \leqslant 10 \\ s(i) &= min\left[\delta(1) \cdots \delta(i)\right] && \text{with } 10 < i \leqslant w \end{aligned}$$

5.2 Replacing the exponential averaging

We tried to use the DLL presented in [1] or a second order low pass filter in place of the exponential averaging. Input of the DLL or of the filter was the output of the first stage selection process. The DLL function is defined as:

$$y(n) = bx(n) + cx(n-1) - (b-1)y(n-1) - cy(n-2) \tag{6}$$

with F sample frequency of the loop
B required bandwith of the filter

and

$w = 2\pi B/F$
$b = \sqrt{2}w$
$c = w^2$

Using the adequate parameters, it behaves exactly as the exponential averaging function (eq. 2).

We tried also to use a low pass second order filter defined as:

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) - a_1y(n-1) - a_2y(n-2) \tag{7}$$

with f_c analog cut-off frequency
ζ damping factor

f_s sampling frequency
$C = 1/[\tan(\pi f_c/f_s)]$

and

$b_0 = 1/(1 + 2\zeta C + C^2$
$b_1 = 2b_0$
$b_2 = b_0$
$a_1 = 2b_0(1 - C^2)$
$a_2 = b_0(1 - 2\zeta C + C^2)$

Although its behavior is slightly different, it produces results very similar to exponential averaging.

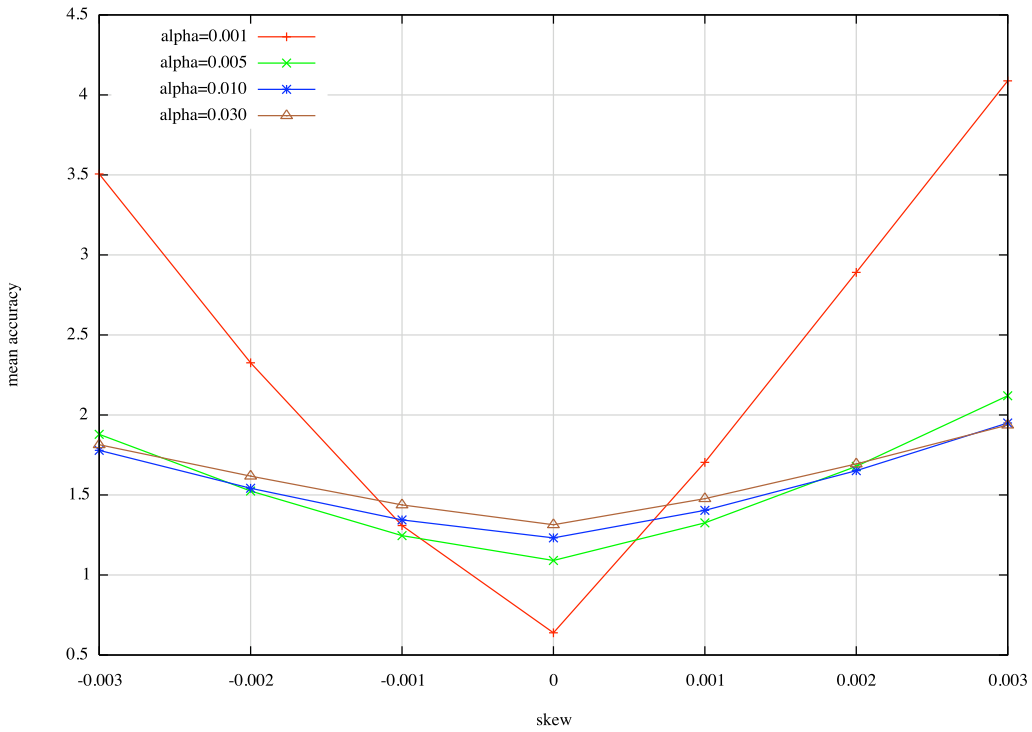

Figure 6: Weight parameter impact on the accuracy with different skews.

5.3 Exponential averaging parameters

The weight parameter α of the exponential averaging function given by equation 2 has been explored thru the following set of values: [0.001, 0.003, 0.005, 0.008, 0.010, 0.015, 0.020, 0.030] . The figure 6 gives one viewpoint over the results: it presents the mean accuracy obtained using different α values when the skew is varying from -0.003 to 0.003. For these measures, the window size was $w = 250$. It is obvious from the results that a small α value gives better results when the skew is small but degrades very significantly when the skew increases.

This view however masks the particularities of the different data sets: the best results are actually obtained using α values varying from 0.003 to 0.03, depending on the delays shape. To get a finer view on the results, we need to differentiate the responsiveness and the stability. The figure 7 shows normalized estimation curves computed using the ADSL2 data set (see figure 10 in appendix) with α values varying from 0.001 to 0.02 and applied to a +0.003 and -0.003 deviations. The graphic clearly illustrates a non-obvious point: positive and negative deviation slopes produce asymmetric results regarding accuracy. In particular, for ascending deviations, instability is generally covering the responsiveness area so that the accuracy A is not $R + I$ and could likely be $max(R, I)$. On the contrary, for descending deviations, we'll generally have $R < A < R + I$. It is also clear from the results that the responsiveness is an issue only for weight parameter $\alpha < 0.005$.

Actually, using $\alpha = 0.008$ gives the best results over the whole data set. The figure 8 presents the mean accuracy obtained over the data set with the distance to the best α value for each deviation. As expected, the distance is maximal for 0 deviation but very small anywhere else.

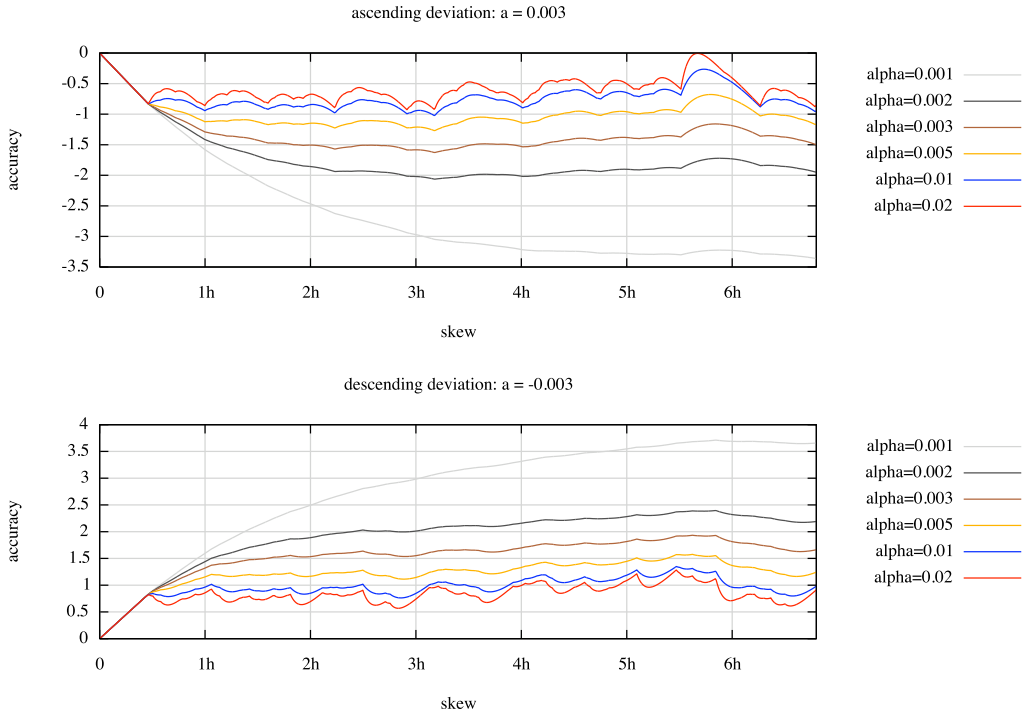

Figure 7: Normalized skew estimations with varying α factor.

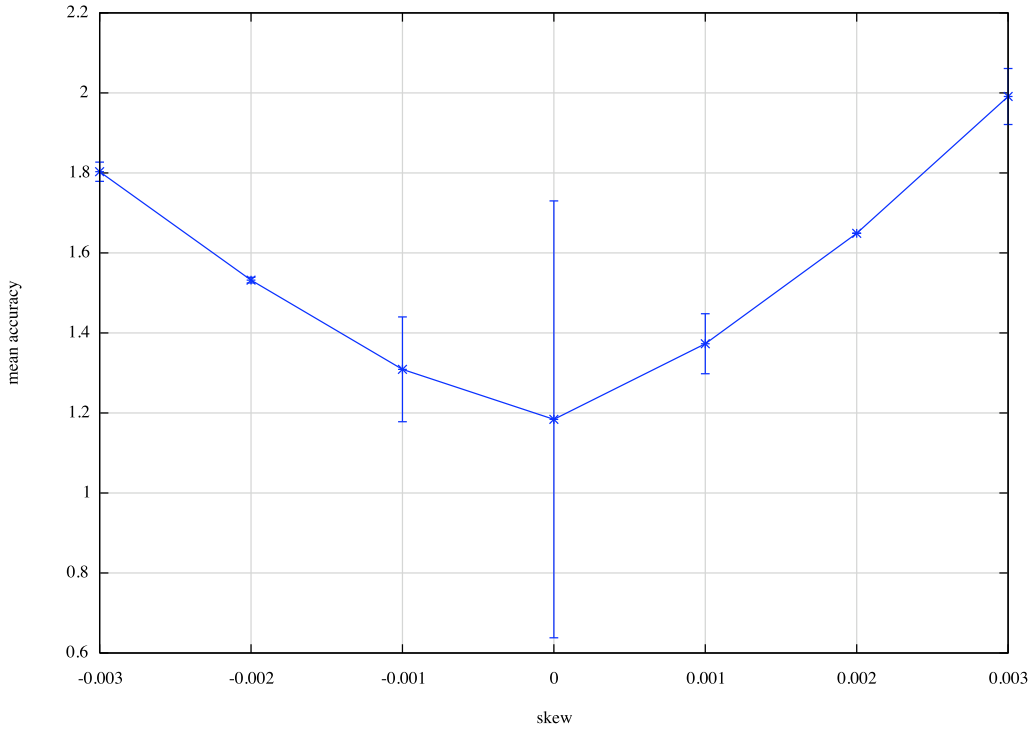

Figure 8: Mean accuracy and distance to optimal values using $\alpha = 0.008$.

5.4 New to previous comparison

The table 3 compares the results obtained using the previous and the new implementation of the algorithm. For the previous implementation, the EPTMA parameters used for the Internet and for the local network are those previously defined i.e.:

mode	window size w	retained values k	weight α
Wan	200	20	0,01
Lan	30	10	0.2

Table 2: EPTMA parameters.

The table presents the accuracy difference between the old and new implementations over the whole data set. Increase of accuracy is obvious except for the LOCAL data sets: this is due to the small weight factor α that affects the responsiveness of the results.

data set	params mode	-0.003	-0.002	-0.001	0.000	0.001	0.002	0.003
AUDIO-1	Lan	7.15	7.18	7.19	7.17	7.13	7.15	7.18
AUDIO-2	Lan	4.79	4.84	4.89	4.92	4.95	4.96	4.66
LOCAL-1	Lan	-1.03	-0.67	-0.30	0.06	-0.24	-0.61	-0.99
LOCAL-2	Lan	-1.03	-0.66	-0.27	0.07	-0.29	-0.65	-1.01
ADSL-1	Wan	24.95	25.53	26.16	26.37	26.13	25.82	25.48
ADSL-2	Wan	0.91	1.33	1.62	1.80	1.73	1.58	1.27
MIDI-1	Wan	0.56	0.54	0.42	0.30	0.05	-0.37	-0.74
MIDI-2	Wan	2.12	2.60	2.93	3.13	3.11	2.95	2.78

Table 3: Accuracy gain using the new algorithm.

data set	-0.003	-0.002	-0.001	0	0.001	0.002	0.003
AUDIO-1	2.90	2.87	2.85	2.88	2.92	2.90	2.88
AUDIO-2	3.99	3.95	3.90	3.86	3.84	3.84	4.09
LOCAL-1	1.18	0.81	0.44	0.08	0.38	0.75	1.13
LOCAL-2	1.12	0.74	0.38	0.06	0.43	0.81	1.19
ADSL-1	1.40	0.95	0.65	0.47	0.57	0.73	1.08
ADSL-2	1.41	0.99	0.55	0.54	0.98	1.52	2.08
MIDI-1	1.34	1.21	1.13	1.04	1.08	1.45	1.81
MIDI-2	1.03	0.75	0.64	0.63	0.85	1.20	1.56

Table 4: Accuracy over the whole data set.

The table 4 gives in detail the accuracy obtained over the whole data set and the figure 9 summarizes these results by plotting the percentage of results obtained according their accuracy. These results are quite encouraging: 41% of the estimations show an accuracy under 1 time unit and 98% are better than 4 time units.

6 Conclusion

We have showed that a minor change in the first selection phase of EPTMA may greatly improve the algorithm performances. Consequently, we suggest to rename EPTMA to EPTLA (standing for Exponential Peak Tolerant Low Point Averaging). Next we were expecting to discover a *killer* function to replace exponential averaging and providing better results: at this stage there is none and only a persevering exploration of the algorithm parameters have brought these expected significant improvements. The good news is that an adequate parameters set is critical and since we've now a better knowledge of the parameters space, we should now make the optimal choice. This choice is however never obvious since we're always facing the compromise between responsiveness and stability.

Another point that has been explored without success concerns the dynamic adaptation of the weighting factor α to improve the stability of the estimation. Failure of this strategy is due to the lack of control over α adjustments that were often too strong, resulting in a distortion of the estimation going the reverse

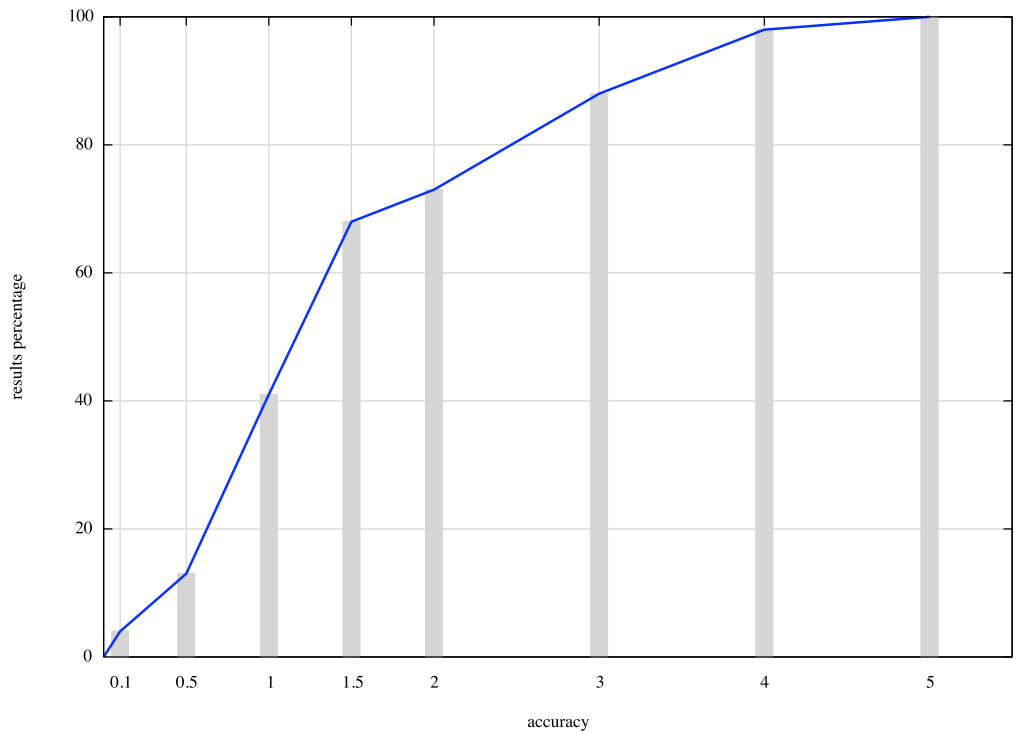

Figure 9: Percentage of results obtained according their accuracy.

way of the delays increase. The good point is that using the difference between the mid point and the low point averaging provides a good indicator of the transient latency changes. This track is however not a dead end and we should continue working on it.

Finally, one investigation of interest concerns the linear programming algorithm: it actually gives the better results a posteriory. Thus defining a reasonable real-time version of the algorithm could lead to significant improvements, notably in term of stability.

References

[1] F. Adriaensen, *Using a DLL to filter time.* http://users.skynet.be/solaris/linuxaudio/downloads/usingdll.pdf, 2005

[2] E. Brandt, R.B. Dannenberg, *Time in Distributed Real-Time Systems.* Proceedings of the International Computer Music Conference, ICMA San Francisco, 1999 pp.523 - 526

[3] P.Dutilleux, U.Zölzer, *Filters.* in DAFX - Digital Audio Effects, John Wiley & Sons Ltd, Editor U.Zölzer, 2003, pp.31-62

[4] D. Fober,Y. Orlarey, S. Letz, *Clock Skew Compensation over a High Latency Network.* Proceedings of the International Computer Music Conference, ICMA San Francisco, 2002 pp.548 - 552

[5] D. Fober,Y. Orlarey, S. Letz, *Real Time Musical Events Streaming over Internet.* Proceedings of the International Conference on WEB Delivering of Music, IEEE, 2001 pp. 147 - 154

[6] D. Fober, *Audio Cards Clock Skew Compensation over a Local Network.* Technical Report - 02-04-01 Grame 2002

[7] Mills, D.L., *Network Time Protocol (NTP) version 3.* IETF, RFC 1305, 1992

[8] Mills, D.L., *Adaptive hybrid clock discipline algorithm for the network time protocol.* IEEE/ACM Transactions on Networking Volume 6, Issue 5, Oct. 1998 pp.505 - 514

[9] Sue B. Moon, Paul Skelly, Don Towsley, *Estimation and Removal of Clock Skew from Network Delay Measurements.* Proceedings of 1999 IEEE INFOCOM, New York, NY, March 1999, pp.227 - 234

[10] L. Lamport, R. Shostak, M. Pease, *The Byzantine Generals Problem.* ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, pp.382-401.

[11] V. Paxson, *On Calibrating Measurements of Packet Transit Times.* Proceedings of SIGMETRICS '98, June 1998

[12] Schulzrinne, H., Casner, S., Frederick, R. & Jacobson, V., *RTP : A Transport Protocol for Real-Time Applications.* RFC 3550, IETF, 2003.

Appendix 1: measurements set

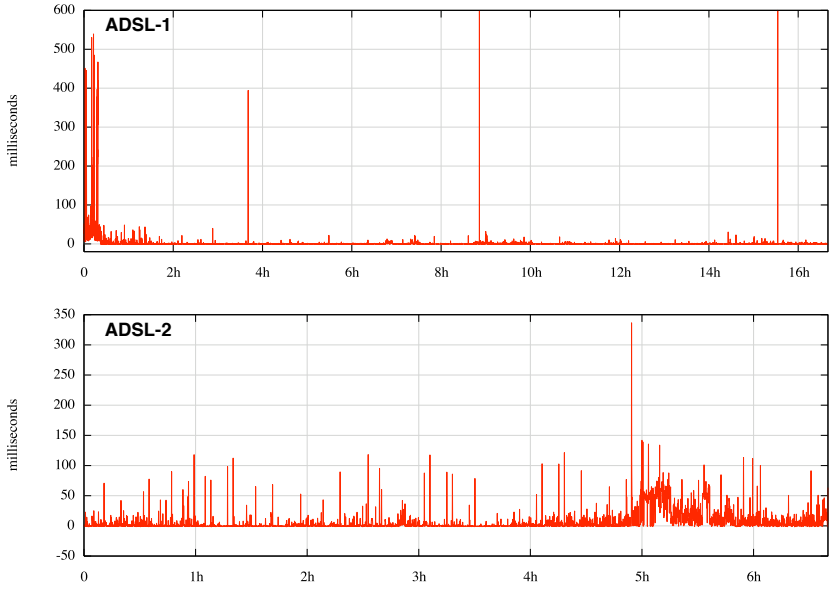

Figure 10: Ping measurements over Internet via adsl. The first transmission exhibits a high delay period at the session start time. The second transmission shows a significant delay increase starting at hour 5.

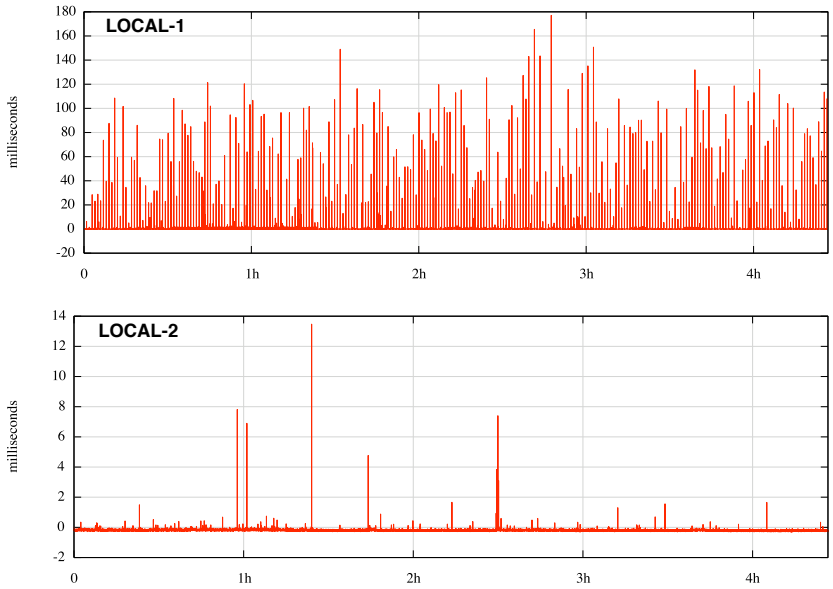

Figure 11: Ping measurements over a local Ethernet network. The first transmission has been measured on a general purpose 10Mb Ethernet network. The second transmission has been measured on Ethernet 100Mb isolated by a switch.

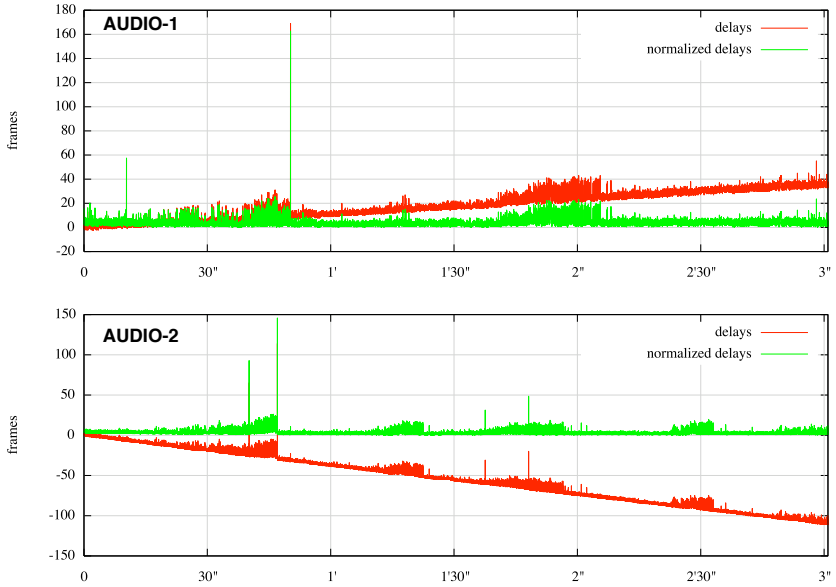

Figure 12: Audio buffers transmission over Ethernet. 4 different stations were involved. Measurements of 2 stations are presented along with the corresponding normalized delays.

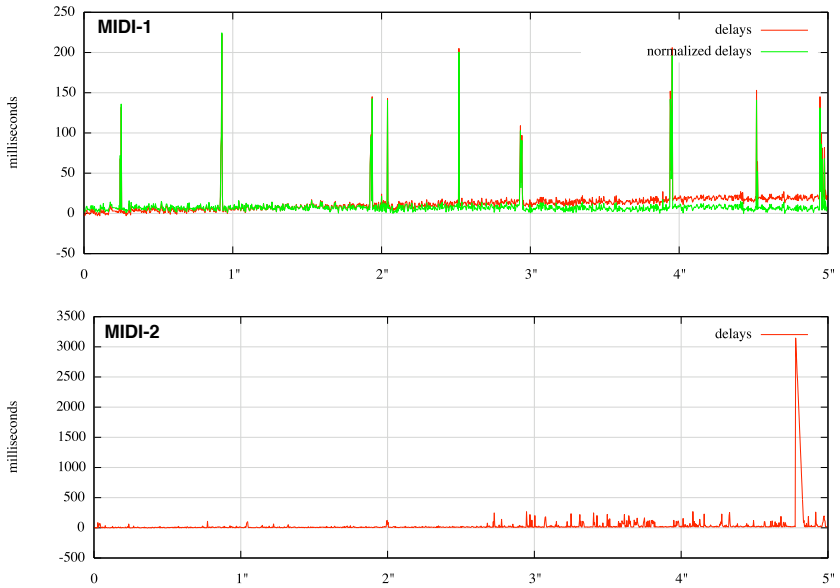

Figure 13: MIDI transmission over Internet via a modem line. For the second transmission, note the latency change starting at 3". Note also that the normalized delays are not plotted because they were not significant.

Appendix 2: EPTLA scilab code

```
//-------------------------------------------------------------
// the eptla algorithm
// parameters:
//  - x    : a vector of latency variations
//  - win  : the sliding window size
//  - w    : the exponential averaging weighting factor
// result:
//  - y    : the clock deviations vector
//-------------------------------------------------------------
    function y=eptla(x, win, w)
        n=length(x);
        low=min(x(1:win));
        y=[ones(1:win)'*low; zeros(1:n-win)'];
        for i=win+1:n
            low=min(x(i-win:i));
            y(i)=w*low + (1-w)*y(i-1);
        end
    endfunction
```