Synchronization of sampling in distributed signal processing systems

Károly Molnár, László Sujbert, Gábor Péceli
Department of Measurement and Information Systems, Budapest University of Technology and Economics, and
Embedded Information Technology Research Group, Hungarian Academy of Sciences, Budapest, Hungary
Phone: +36 1 463-4114, fax: +36 1 463-4112, e-mail: sujbert@mit.bme.hu

Abstract – In distributed signal processing systems, every node samples analog signals by its own AD converter. Sampling is controlled by autonomous clocks, that are generally not synchronizable. In order to ensure synchronized operation among the different nodes of the distributed system, both the drift of these clocks, and the jitter of the sampling must be handled. This problem is investigated in this paper, and signal processing methods are proposed to guarantee the synchronized operation and to reduce the effect of the jitter. The proposed solution is based on low-level interpolation and resampling of the sampled signals. A realization of this method is demonstrated on a test-system.

Keywords – Digital Signal Processing, Distributed Systems, Sampling

I. INTRODUCTION

A distributed signal processing system comprises numerous processor nodes (mostly based on DSPs) which are interacting with each other to perform real-time data acquisition and signal processing. An overview of this system is presented in Fig. 1. Such systems are also known as intelligent sensor networks, where processing units are placed near to the sensors. Distributed DSP systems are used, e.g., in seismic wave measurements.

In DSP-based systems, the nodes are performing online signal processing, i.e., the DSP algorithm is executed sample-by-sample. The input data for this algorithm are digital samples of a discrete signal, usually sampled at the clock rate of the DSP algorithm. The algorithm calculates the samples of the output signal in this pace, so the output signal (which is also a discrete digital signal) is available with the same sampling frequency.

Distributed processors and data acquisition units have separate clocks, that may hurt data consistency constraints, due to their jitter and drift [1]. This problem is investigated in this paper. In distributed embedded systems, data consistency asks for synchronous data acquisition and representation. This problem does not exist in centralized one-processor systems, as generally these have only one master sampling clock that schedules all the sampling processes.

In distributed systems, this phenomenon leads to more serious problems due to the online signal processing. As normally the DSP algorithm is the most important task, all the other tasks (e.g. communication) are synchronized to this. The slight drift of the different sampling clocks therefore is a serious problem if the synchronized operation of DSPs is required. The timing jitter of sampling and communication also have to be examined, as these effects are combined with the drift.

The field of distributed signal-processing is not a well explored area, so after the examination of this complex problem, widely utilizable solutions are proposed. These are low-level signal processing methods that assure the synchronized sampling of the different DSP nodes, enabling distributed operation.

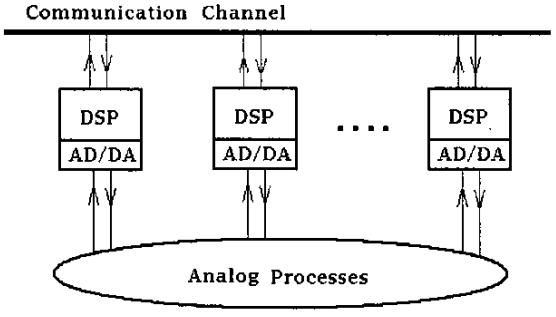

Fig. 1. Distributed DSP system

In the following section, the problem of distributed sampling is investigated in detail. In Section III a truly software solution is proposed for this problem, that uses digital signal processing algorithms, namely interpolation in order to make the asynchronously sampled real-time data of different sources consistent. In Section IV a test system is presented, that is used to implement the previously exposed methods. Test results, and interesting aspects of the implementation are also presented. In the last section, the conclusions are stated.

0-7803-7864-4/03/$17.00 ©2003 IEEE.

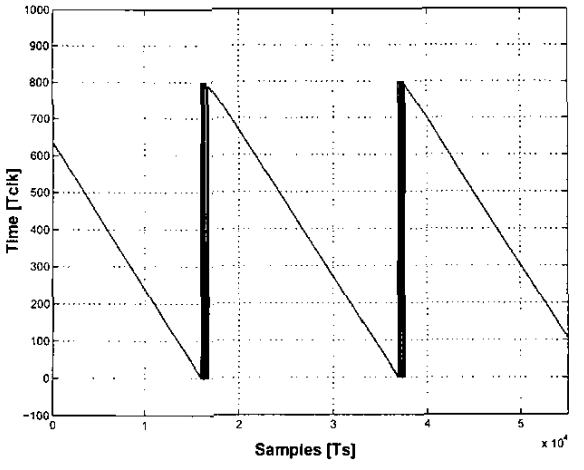

Fig. 2. Time difference between the ticks of the sampling clocks

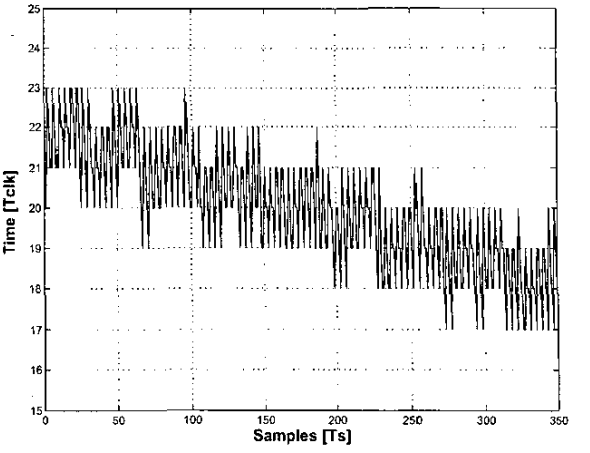

Fig. 3. Time difference between the ticks of the sampling clocks, during the normal period

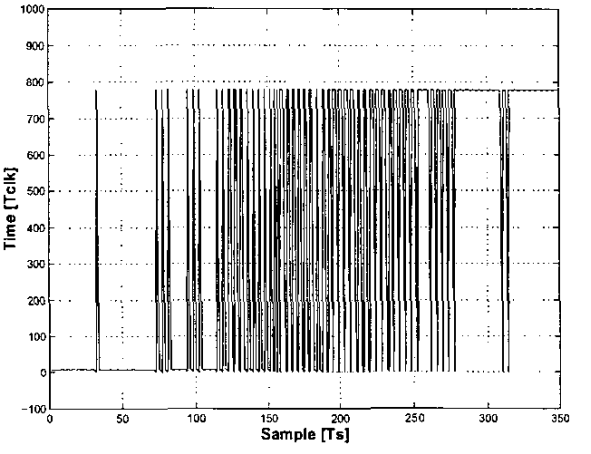

Fig. 4. Time difference between the ticks of the sampling clocks, during the transient period

II. PROBLEMS OF DISTRIBUTED SAMPLING

In distributed DSP systems, a problem arises when two or more DSP nodes having asynchronous clocks are communicating by sending samples of real-time discrete signals.

Consider the case, when the input signal of a certain node (Drain) is the output signal of another node (Source). This kind of real-time communication presumes that the sample rate of the data is the same at both nodes. The sampling is usually controlled by crystal oscillators, where the frequency deviation is typically 50 ppm. The problem arises as the DSP algorithm of the Drain is scheduled by its own crystal oscillator, but the input samples are sampled by the AD of the Source. These sampling frequencies are not exactly the same at the nodes, as the two crystals are not identical. Therefore the data representation of the nodes is not consistent, as during a certain period of time the Source produces different number of samples than the number of samples expected by the Drain.

Additionally, if we suppose that the communication is packet-based, and one sample is sent in one packet, then due to this difference, sometimes two or zero packets arrive during the period when the Drain expects exactly one packet. In this case, data packets are lost. The jitter of the sampling and the communication worsens this effect, as numerous packet are lost with a certain probability when the sampling moments of the nodes are close in time.

This phenomenon is shown on Fig. 2., 3. and 4. In the figures, the time periods between the arrival of the samples from the Source and the ticks of the sampling clock of the Drain is plotted (Time [T_{clk}]). This time period is measured by the internal timer of the Drain, therefore the resolution of the measurement is 1 T_{clk}.

Fig. 2. is an illustrative figure of the whole process, while Fig. 3. and 4. are real measurement results obtained by the Measurement System described in Section IV. In Fig. 2. the effect of the drift can be examined, the ticks of the two sampling clocks get closer and closer with every sample. When this difference is close to zero, the actual order of the two events is stochastic. The length of this "transient" period is proportional to the jitter of the sampling and the communication.

Fig. 3. and 4. are zoomed parts of Fig. 2. Fig. 3. shows a section of the normal descending curve. The jitter can be examined on the curve, as the values are not monotonic. In Fig. 4, the "transient" period can be seen, where the tick of the faster clock outstrips the slower, causing a step in the measured time period. It can be seen, that due to the jitter, this happens several times, as the order of the events are stochastic during this transient period.

The problems due to the asynchronous sample clocks are

summarized in the following:

1. The drift causes a constant error in the frequency of the real-time signal.
2. During the transient period packets (containing samples) are lost, as with a certain probability zero or two samples arrive when exactly one packet is expected.

These problems cannot be solved by buffering or by sending more samples together. This would only cause bigger delay in the overall process, but does not help the fact that the Source produces different number of samples than the Drain expects.

III. PROPOSED SOLUTION

Any solution of this problem has to ensure that the samples of the different input data streams of a certain DSP node are available with the same sampling frequency. In the above shown example, the data stream (samples of discrete signal) from the Source is expected with the sampling frequency of the AD of the Drain.

An obvious hardware solution is to synchronize the sampling clocks of the ADs by e.g. a PLL. However, this requires a strict hardware connection between the nodes, that is not always available in distributed systems, as the nodes usually communicate via some standard communication channel.

We propose a software solution for this problem, that does not influence the hardware layers of the DSP node, so the sampling process of the different nodes remain asynchronous. The solution is based on the concept that the synchronization of the signal processing nodes and the sampling has to be separated.

The solution is presented in detail for two communicating nodes in the following. One of the two nodes has to perform a transformation on the digital signal, namely the signal has to be resampled. In detail, the signal sampled by the clock of the Source has to be interpolated and resampled with the clock of the Drain. This is illustrated in Fig. 5.

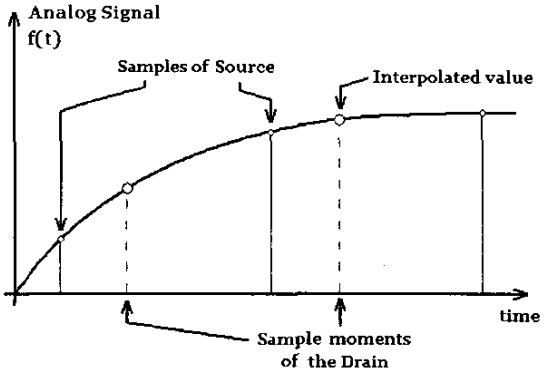

Fig. 5. Resampling

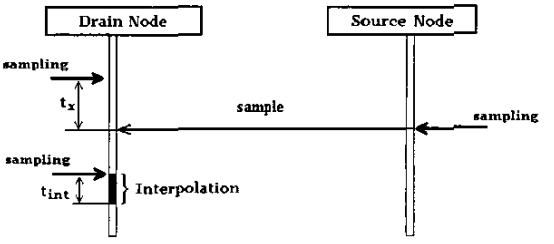

Fig. 6. Interpolation at the Drain

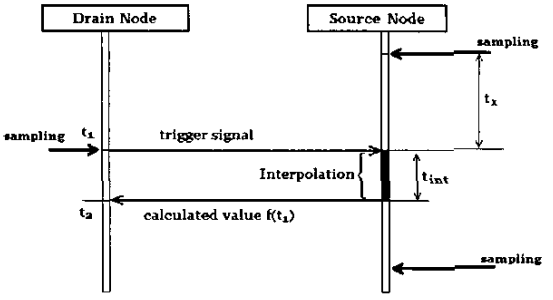

Fig. 7. Interpolation at the Source

The interpolation can be performed at both DSP nodes, and both possibilities should be considered.

1. Interpolation at the Drain - The Source sends the data packets with is own sampling frequency (f_S), the Drain registers the exact arrival time of the packets and performs the interpolation calculation on these received data to evaluate the values of the analog signal at its own sampling moments. This solution is illustrated at Fig. 6.
2. Interpolation at the Source - A trigger signal is sent to the Source when the Drain exactly performs the sampling (t_1). The Source measures the time elapsed from its last sample to this trigger event (t_x). Then the interpolation can be performed, namely $f(t_1)$ is calculated. Then this data is sent to the Drain. So this packet is sent after each trigger, with the frequency of the Drain. This solution is illustrated at Fig. 7.

The two solutions have the same result theoretically, therefore interpolation should be done by that node which has more free capacity.

In a typical distributed DSP system, there are more than two nodes, and normally a node is both receiving and sending real-time data. However, the communication flow in the whole system can be dissolved to point-to-point communications, where the Drains and the Sources can be identified. The first approach, when the interpolation is performed at the Drain, can be adequate if there are many nodes in the distributed system, as this solution ensures that every node prepares the different data streams for itself.

The second solution, when the interpolation is performed at

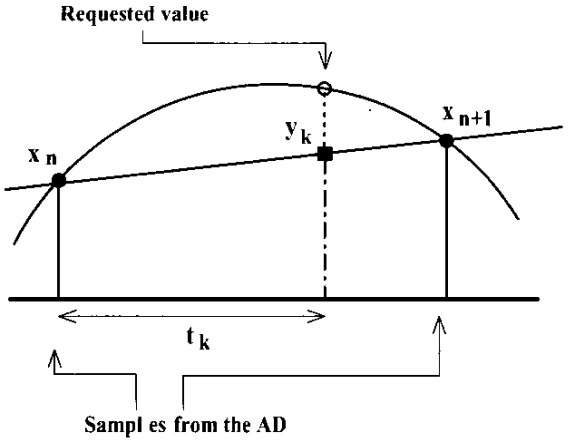

Fig. 8. Linear interpolation

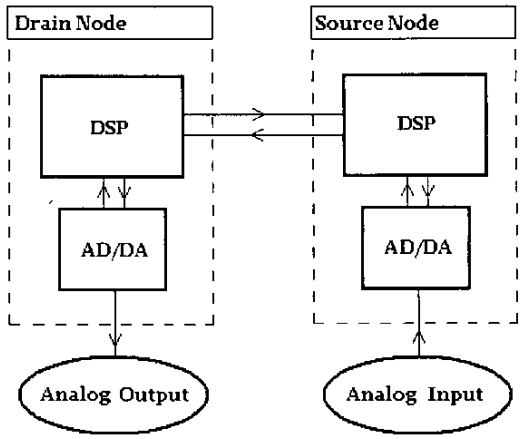

Fig. 9. The built test system

the Source, is more close to a Master-Slave kind of approach, as the Source (Slave) preprocesses the data for the Drain (Master), so the Source is in-fact scheduled by the sampling clock of the Drain. This approach is useful in smaller systems, and in case there is a two-way data-stream between a node pair.

There are numerous possibilities to realize the interpolation. Namely:

1. Interpolation filtering [2], [3] - As the algorithm is realized by a DSP, the first solution that comes to mind is the interpolation filtering. This is the well-known method used to increase the sampling frequency of discrete signals. This practically means an implementation of a linear-phase FIR filter with a known delay: $t_{int} = (N/2) * t_S$, where N is the order of the filter. The precision of the calculation is proportional to the order of the filter. The delay and the computation demand is also growing with the order.
2. Linear interpolation - The linear interpolation is calculated the following way:

$$y_k = x_n + \frac{t_k(x_{n+1} - x_n)}{T_s} \quad (1)$$

where x_n are the values of the analog signal at t_n, and y_k are the interpolated values at t_k. This is a more simple approach(equivalent to an interpolation filter with $N = 2$), that requires minimal calculation. However, the accuracy is limited. See Fig. 8.
3. Prediction - If the delay has to be smaller than T_S, the value $f(t_k)$ can be calculated immediately after the t_k is available. However, the effectiveness of this solution highly depends on the signal processing problem to be solved.

IV. REALIZATION AND TEST RESULTS

In order to examine the stated problem and to implement the proposed solutions, a test system has been built. This system comprises of two identical DSP nodes, that are communicating via a general purpose synchronous serial port. The DSPs are Analog Devices ADSP 21061's [4], the AD/DA converters are realized by AD1847 codecs [5]. (The ADSP-21061 SHARC EZ-KIT Lite evaluation Board is used [6].) This system is shown in Fig. 9.

The test system realizes the "Interpolation at the Source" solution, which was proposed in section III. The Drain DSP signals to the Source at every tick of the sample clock (trigger). The Source is sampling a sinusoid signal by its AD, and performs the interpolation on the sampled data, in order to calculate the value of the signal in the triggered moment. This value is than sent to the Drain. The communication flow is the same to the one illustrated at Fig. 7.

If the sinusoid signal is restored from the samples without interpolation at the Drain, then it is distorted and contains transients due to the problems shown in Section II. In order to measure the quality of the interpolated signal, an Adaptive Fourier Analyzer (AFA) ([7],[8]) is realized on the Drain. This is a DSP application that can be used (apart from other purposes) for precise frequency measurement of periodic signals. We used the AFA to measure the frequency of the interpolated signal. The measurement setup is shown in Fig. 10. The generator was a high precision Bruel and Kjaer Sine Generator Type 1051.

On the test system, the interpolation calculation was realized three different ways: linear interpolation, interpolation filtering and interpolation filtering combined with linear interpolation.

The signal calculated by linear interpolation is quite stable,

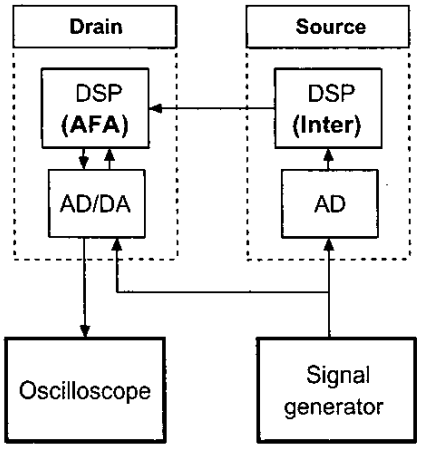

Fig. 10. Measurement setup

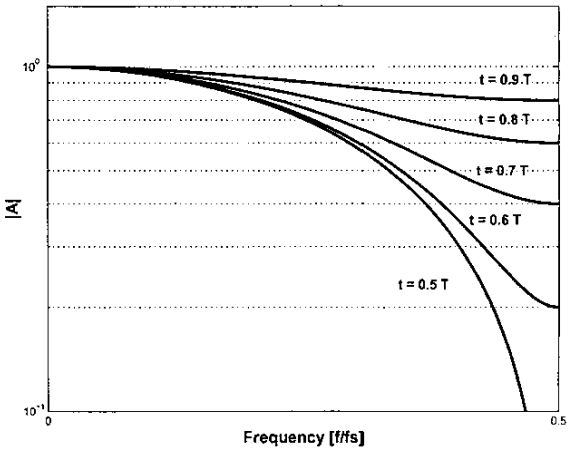

Fig. 11. Frequency characteristics of the linear interpolation

and the frequency is much more accurate than the frequency of the signal without interpolation. See Table. I.

Without interpolation	$1.1164 \cdot 10^{-5}$
Linear interpolation	$0.2580 \cdot 10^{-5}$

Table I. Frequency measurement results

The values in Table. I. are the relative error versus the correct frequency value. These errors are calculated from frequency values obtained by the AFA. The frequency of the sinusoid that is directly sampled by the AD of the Drain was considered to be the *true* value, while the frequency of the sinusoid arriving from the Source was the *measured* value.

However, the linear interpolation can only be used in a limited frequency range, as the interpolated sinusoid has an amplitude deviation at higher frequencies. This error is due to the fact, that the Eq.(1) equation realizes a 2-tap FIR filter with the coefficients a_1 and a_2, where:

$$a_1 = \frac{T_s - t}{T_s}, \quad a_2 = \frac{t}{T_s} \tag{2}$$

The frequency characteristics are shown for a few values in Fig. 11. The coefficients vary with t value, which is the difference between the ticks of the two sampling clocks, so it is periodically (the value shown on Fig. 2) changing. This results an amplitude deviation of the interpolated signal, as the frequency spectra follows this change.

The amplitude variation limits the use of the linear interpolation. This effect is totally avoided if the interpolation is performed by filtering. However, the interpolation filtering method requires more calculation capacity and has a bigger delay.

The accuracy of the calculated sample is limited by the spectra of the interpolation filter. In order to calculate accurate samples, a long FIR filter is needed, which causes bigger delay, and requires increasing calculation capacity. As a trade-off, the filtering is combined with the linear interpolation. This solution calculates the sample in two steps. First, 8 times interpolation filtering is performed, which generates the discrete signal with a sample rate 8 times higher than the original sample rate. In the second step, linear interpolation is performed between the two samples nearest to the requested value. This is illustrated in Fig. 12. This solution provides an acceptable delay, and stable signal with reasonable calculation requirements.

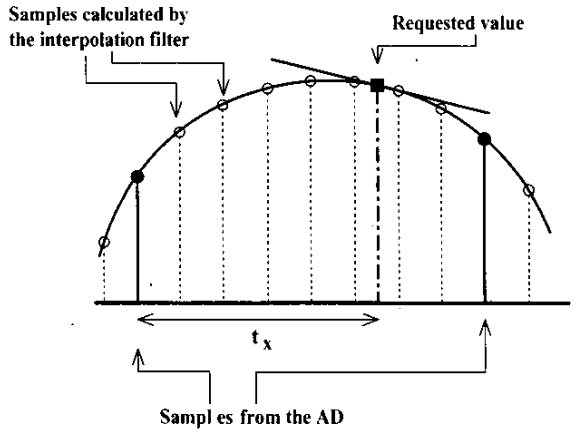

Fig. 12. Interpolation filtering combined with linear interpolation

The calculation requirements of the interpolation filtering is considerably reduced in the DSP implementation, as the samples not needed for the linear interpolation are not evaluated.

The correct implementation of the above presented methods require the accurate measurement of t, the time period between the latest tick of the sampling clock and the trigger signal at the Drain. Normally a timer is started and stopped when these events happen. Care should be taken that these actions are performed as close to the event as possible, because due to e.g. interrupt nesting these action can suffer random delay, causing a jitter-like error in the measurement of t. To avoid this, the priority levels of the ITs of these hardware units have to be high enough to ensure the immediate action.

V. CONCLUSIONS

In our paper a typical problem of distributed signal processing is considered. The drift between two autonomous real-time clocks is a well-known problem in the field of embedded systems. A similar effect is present in distributed signal processing systems, as there is drift between the two sampling clocks of the AD converters of two different nodes. This effect, combined with the jitter of the sampling, can preclude the synchronized operation of the distributed system. To solve these problems, signal processing methods are proposed, and the different possibilities of the realization are examined. A test system is also described, which was built in order to perform measurements and to implement the proposed solution.

As the problems of distributed signal-processing is not a well explored field in the literature, a general signal processing solution is proposed, that is utilizable in a wide range of applications.

In the future, research should be extended to the field of prediction mentioned in Section III. In this case, interpolation and the signal processing task can be jointly handled. For processing of periodic signals, the Adaptive Fourier Analyzer can be modified such a way, that it can perform the resampling and the signal analysis simultaneously.

REFERENCES

[1] Hermann Kopetz, *Real-Time Systems, Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, 1997.

[2] R. E. Crocherie and L. R. Rabiner, *Multirate Digital Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1983.

[3] P.P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial," *Proceedings of the IEEE*, vol. 78, no. 1., January 1990.

[4] Analog Devices, Inc., *ADSP-2106x SHARC User's Manual*, second edition edition, 1997.

[5] Analog Devices, Inc., *Serial-Port 16-Bit SoundPort Stereo Codec AD1847*, 1996.

[6] Analog Devices, Inc., *ADSP-2106x SHARC EZ-KIT Lite Reference Manual*, 1997.

[7] Gábor Péceli, "A common structure for recursive discrete transforms," *IEEE Trans. on Circuits and Systems*, vol. CAS-33, pp. 1035–36, October 1986.

[8] Ferenc Nagy, "Measurement of signal parameters using nonlinear observers," *IEEE Trans. on Instrumentation and Measurement*, vol. IM-41, pp. 152–155, February 1992.