

SELF-AWARE DISTRIBUTED AV SENSOR AND ACTUATOR NETWORKS FOR IMPROVED MEDIA ADAPTATION

Rainer Lienhart

Intel Research, Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052, USA
Rainer.Lienhart@intel.com

Igor Kozintsev

Intel Research, Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052, USA
Igor.V.Kozintsev@intel.com

ABSTRACT

Most of the existing research work in the area of media adaptation is currently concentrated on content adaptation, transcoding and delivery mechanisms without addressing the actual input and output of multimedia data. However, it is the I/O stage of media processing that humans are concerned about. Up until now most multimedia applications were relying on standalone I/O devices (microphone, headphones, monitor, camera) to capture or render multimedia data. This situation is about to change. Nowadays we are surrounded by a vast number of Audio/Video (AV) sensors and actuators. They are built into our cellular phones, PDAs, tablets, laptops, and surveillance systems. A natural idea that comes out of this fact is to combine multiple I/O devices into a distributed array of sensors and actuators. The goal of this paper is to show the feasibility of this idea and to shift the media adaptation research away from a single device/stream paradigm towards array multimedia processing. We demonstrate how to transform a network of off-the-shelf devices into a distributed I/O array by providing common time (with tens of microseconds precision) and 3D space coordinates (with a few centimeters precision). We also discuss the implications and potentials of self-calibrating distributed AV-sensor/actuator networks for improved media adaptation.

1. INTRODUCTION

Current media-adaptation and media-transcoding research focuses on transforming a given media stream subject to a set of target requirements. While usually the target (presentation) format of the media stream under transformation is well defined, none or only very little meta data accompanies the source media stream to guide the media adaptation process. Also most media-adaptation and media-transcoding research implicitly assumes that the media stream consists of only a few streams such as a single video stream accompanied by its respective stereo audio stream and targets a single input or output device.

In contrast, our approach assumes that a vast number of distributed sensors (e.g., microphones, cameras), actuators (e.g., loudspeakers, displays), and computing resources are used to capture and render transformed media streams. Densely sampled multi-channel media recordings together with basic meta data such as precise sample position in time and space can enable enhanced media adaptation on the data acquisition side. On the rendering side, availability of multiple rendering devices makes it possible to construct true 3D multimedia scenes and cater them to user's position and orientation as well as to the surrounding environment. The

main goal of this paper is to propose a new dimension for media adaptation research by including array I/O into the picture, and to prove that it is indeed possible to implement distributed I/O arrays using a network of general purpose computing and communicating (GPC) devices.

To illustrate this idea consider a standard teleconferencing application. In the systems available on the market today a single camera and display is used to transfer the video portion while a single microphone (or in some cases dedicated microphone array) and standard speakers are used to send a single audio channel. Such systems typically require dedicated infrastructure and tedious setup procedures leading to high costs.

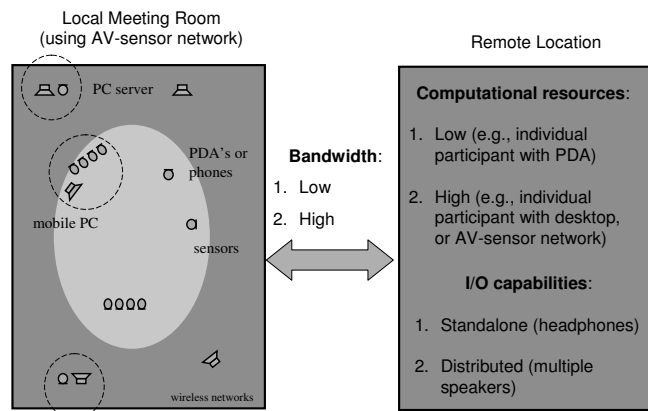


Fig. 1. Scenario of distributed multi-microphone/multi-speaker teleconferencing with GPCs.

In contrast, Figure 1 shows a new scenario of teleconferencing using distributed I/O devices. GPCs such as laptops, PDAs, tablets, and/or cellular phones of the teleconference participants are connected wirelessly via IEEE 802.11a/b/g in a local meeting room. Their sensors record the auditorial scene with N synchronized streams. In addition, the 3D relative positions of all sensors and actuators are available as metadata. No dedicated infrastructure is required in this case and the I/O devices are capable of performing self-calibration and self-adaptation. Figure 1 further suggests possible ways of media adaptation for rendering multimedia scene at a remote location (that might be similar to the local room). Depending on the bandwidth, computational and I/O capabilities at the remote site the following cases are possible (we do not list

all possible combinations):

1. **Low bandwidth and computational resources, standalone I/O:** Due to the low bandwidth and low remote CPU resources complex audio array signal processing is performed at the local site. For instance, given the 3D positions of all sensors and their recordings, all sound sources of interest (usually speakers) can be identified and separated into different audio tracks by beamforming using the distributed microphone array¹. Then, a single audio track of a selected speaker is sent to the remote site where it is rendered on a standalone output device.
2. **Low bandwidth, high-performance GPC, distributed I/O:** More sophisticated audio rendering can be used in the above scenario. For example, a 3D sound position can be recreated at the remote site.
3. **High bandwidth, low computational resources, distributed I/O:** In addition to the possibilities above, multiple audio tracks can be sent to the remote site to perform 3D rendering of several speakers.
4. **High bandwidth, high computational resources, distributed I/O:** In addition to all previous possibilities, the unprocessed audio streams together with the meta data can be sent to the remote site, that is free to perform any processing and rendering desired.

However, in order to enable teleconferencing with distributed array I/O, as well as other multimedia applications, several important technical and theoretical problems need to be addressed. Two of the most important problems are (1) to provide a common reference time to a network of distributed computers and their I/O channels and (2) to determine the relative 3D positions of all sensors and actuators in the system. The rest of the paper describes our proposed solutions to these two problems.

2. COMMON TIME

We tackle the problem of distributed I/O synchronization in two steps: (1) (inter-platform) the local CPU clocks of the GPCs are synchronized against a global clock, and (2) (intra-platform) I/O is synchronized against the local clocks and thus also against the global clock. One of the CPU clocks may arbitrarily be chosen as the global clock.

Problem Formulation:

Each GPC has a local CPU clock (e.g., the Read-Time Stamp Counter (RDTSC) instruction of the Pentium[®] processor family; the RDTSC counts clock ticks since the processor was started). Let $t_i(t)$ denote the value of this clock on the i -th GPC at some global time t . Assuming a linear model between the global clock and the local platform clock, we get

$$t_i(t) = a_i(t)t + b_i(t), \quad (1)$$

where $a_i(t)$ and $b_i(t)$ are timing model parameters for the i -th GPC. The dependency of the model parameters on global time t approximates instabilities in the clock frequency due to temperature variations and other factors. In practice, these instabilities are in the order of 10^{-5} . In the rest of the paper we will omit explicit

¹Alternatively, Blind Signal Separation (BSS) can be used [1].

time dependency to simplify our notations. Similarly, the sampling times of audio A/Ds and D/As on GPC's are approximated as:

$$\tau_i(t_i) = \alpha_i(t_i)t_i + \beta_i(t_i). \quad (2)$$

In this model τ_i is the number of samples produced by A/D (or consumed by D/A) converter since the start of the audio I/O. Note that two different timing models are required since the audio I/O devices on a typical PC platform have their own internal clock that is not synchronized to other platform clocks such as the RDTSC.

Given the two timing models above the problem is to continuously estimate their parameters as well as the inverse functions such as $t(\tau_i)$ - the global time stamp of audio sample τ_i , $t_i(\tau_i)$ - the local time stamp of audio sample τ_i , and $t(t_i)$ - the global time given the local platform time t_i .

Timing relationships on GPC platform:

In order to understand the inter and intra platform synchronization methods proposed we briefly describe the operations and timing relationships on a typical GPC. Figure 2 shows a processing di-

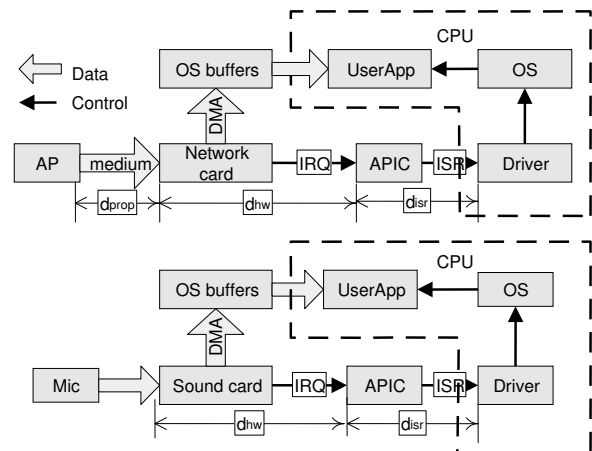


Fig. 2. Network (top part) and audio (bottom part) data and control flows on a typical GPC platform.

agrams of networking and audio I/O. Both I/O operations have a very similar structure that can be described by the following sequence of actions (only input path is described):

1. Incoming data is received and processed by a hardware device, and eventually is put into a Direct Memory Access (DMA) buffer. This is modeled in Figure 2 by the delay d_{hw} , which is approximately constant for similar hardware.
2. The DMA controller transfers the data to a memory block allocated by the system and signals this event to the CPU by an Interrupt ReQuest (IRQ). This stage introduces variable delay due to memory bus arbitration between different agents (i.e., CPU, graphics adapter, other DMA's).
3. The interrupt controller (APIC) queues the interrupt and schedules a time slot for handling. Because APIC is handling requests from multiple I/O devices this stage introduces variable delay with standard deviation of around 6 *ms* and the maximum deviation of 30 *ms*. Both previous stages are modeled by d_{isr} in Figure 2.

4. The Interrupt Service Routine (ISR) of the device driver is called, and the driver sends notification to the Operating System (OS).
5. The OS delivers a notification and data to the user application(s). This stage is executed in a multitasking software environment and this leads to significant variable delays that depend on CPU utilization and many other factors.

In summary, data traverses multiple hardware and software stages in order to travel from an I/O device to the CPU and back. The delay introduced by the various stages is highly variable making the problem of providing a global clock to the GPCs and distributing it to I/O devices very challenging. It is advantageous to perform synchronization as close to hardware as possible, therefore our solution is implemented at the driver level (during ISR) thus avoiding additional errors due to OS processing.

Providing global clock (inter-platform synchronization):

For the synchronization of CPU clocks over a wireless network we propose to use a series of arrival times of multicast packets sent by the wireless access point (AP). In our current approach we implement a pairwise time synchronization with one node chosen as the master (say $t(t_0) = t_0$). All other nodes (clients) are required to synchronize their clocks to the master. A similar approach was also suggested in [2, 3]. Our solution, however, extends it by introducing additional constraints on the timing model. In order to provide a global clock to distributed platforms that is useful to several applications (e.g. joint stream processing and distributed computations) we impose the clock monotonicity condition to make sure that the global clock is monotonically increasing during model parameter adaptation. In addition we smooth the clock model (a_i and b_i in equation (1)) variation by limiting the magnitude of its updates.

The algorithm consists of the following steps:

1. AP sends next beacon packet.
2. Master node records its local time of packet arrival and distributes it to all other nodes.
3. Client nodes record both their local times of arrival of beacon packets from AP, and the corresponding times received from the master.
4. Clients update local timing models based on the set of local timestamps and corresponding master timestamps.

Let us assume that in Figure 2 the packet j arrives to multiple platforms approximately at the same global time corresponding to local clocks t_i^j ($d_{prop} \approx 0$). The set of observations available on the platforms consist of pairs of timestamps $(\tilde{t}_0^j, \tilde{t}_i^j)$. From Figure 2 we have $\tilde{t}^j = t^j + d_{hw} + d_{isr}$ (we omitted dependency on i) that we further model as $\tilde{t}^j = t^j + d + n$. In this approximation d models all constant delay components and n represents the stochastic component. Given the set of observations $(\tilde{t}_0^j, \tilde{t}_i^j)$ we are required to estimate the timing model parameters \hat{a}_i and \hat{b}_i for all client platforms. In our system a window of 3 minutes is used to estimate current values of \hat{a}_i and \hat{b}_i using the least trimmed squares (LTS) regression [4]. LTS is equivalent to performing least squares fit, trimming the observations that correspond to the largest residuals (defined as the distance of the observed value to the linear fit), and then computing a least squares regression model for the remaining observations.

Synchronizing audio to CPU clock (intra-platform synchronization):

In order to synchronize the audio clock to the CPU clock we use a similar approach as the one presented in the previous section. The ISR of the audio driver is modified to timestamp the samples in the OS buffer using the CPU clock to form a set of observation pairs $(\tilde{t}_i^j, \tau_i^j)$, where j now represents the index of an audio data packet.

Following our model in Figure 2 we have $\tilde{t}^j = t^j + d_{hw} + d_{isr}$ (we omitted dependency on i) that we further represent as $\tilde{t}^j = t^j + d + n$. Except for the fact that the τ^j are available without any noise (it is simply the number of samples processed!) we are back to the problem of determining the linear fit parameters for pairs of observations that we solved in the previous section using the LTS method.

In summary, by using LTS procedure twice both local and global synchronization problems are solved and the audio samples can be precisely synchronized on the distributed GPCs. Experimental results demonstrate a precision in A/D and D/A synchronization better than 50 μs across different distributed platforms. See [5] for more details.

3. COMMON SPACE

Given a set of M acoustic sensors (microphones) and S acoustic actuators (speakers) in unknown locations, our goal is to estimate their three dimensional coordinates. Let \mathbf{m}_i for $i \in [1, M]$ and \mathbf{s}_j for $j \in [1, S]$ be the three dimensional vectors representing the spatial coordinates of the i^{th} microphone and j^{th} speaker, respectively. Each of the acoustic actuators is excited using a known calibration signal such as maximum length sequences or chirp signals, and the Time Difference Of Arrival (TDOA) is measured. For each pair of microphones and a speaker the TDOA is defined as the time difference between the signal received by the two microphones. Let $TDOA_{ikj}^{estimated}$ be the estimated TDOA between the i^{th} and the k^{th} microphone when the j^{th} source is excited. Let $TDOA_{ikj}^{actual}$ be the actual TDOA. It is given by

$$TDOA_{ikj}^{actual} = \frac{\|\mathbf{m}_i - \mathbf{s}_j\| - \|\mathbf{m}_k - \mathbf{s}_j\|}{c} \quad (3)$$

where $c = (331 + 0.6T)m/s$ is the speed of sound in the acoustic medium of temperature T (in Celsius). $TDOA_{ikj}^{actual}$ assumes that all the sensors start capturing at the same, and that the loudspeaker emits the signal at the same time. In our setup, however, the precision of the I/O synchronization is limited. Thus the TDOF which we measure from the signal captured includes both the uncertainty in speaker emission start time and the microphone capture start time. The speaker emission start time is defined as the time at which the sound is actually emitted from the speaker. The microphone capture start time is defined as the time instant at which capture is started.

Including the source emission and capture start times, it becomes

$$T\hat{D}O_{ikj}^{actual} = \frac{\|\mathbf{m}_i - \mathbf{s}_j\| - \|\mathbf{m}_k - \mathbf{s}_j\|}{c} + tm_k - tm_i \quad (4)$$

In the case of TDOA the source emission time is the same for both microphones and thus gets cancelled out.

We start the audio capture on each GPC one by one. We define the microphone on which the audio capture was started first as our first microphone. In practice, we set $tm_1 = 0$ i.e. the time at which the first microphone started capturing is our origin. We define all other times with respect to this origin. We can jointly

estimate the unknown capture start times along with microphone and source coordinates.

Maximum Likelihood (ML) Estimate:

Assuming a Gaussian noise model for the TDOA observations we can derive the ML estimate as follows. Let Θ , be a vector of length $P \times 1$, representing all the unknown non-random parameters to be estimated (microphone and speaker coordinates and microphone capture start times). Let Γ , be a vector of length $N \times 1$, representing noisy TDOA measurements. Let $T(\Theta)$, be a vector of length $N \times 1$, representing the actual value of the observations. Then our model for the observations is $\Gamma = T(\Theta) + \eta$ where η is the zero-mean additive white Gaussian noise vector of length $N \times 1$ where each element has the variance σ_j^2 . Also let us define Σ to be the $N \times N$ covariance matrix of the noise vector N . The likelihood function of Γ in vector form can be written as:

$$p(\Gamma/\Theta) = (2\pi)^{-\frac{N}{2}} |\Sigma|^{-\frac{1}{2}} \exp -\frac{1}{2}(\Gamma - T)^T \Sigma^{-1} (\Gamma - T) \quad (5)$$

The ML estimate of Θ is the one which maximizes the log likelihood ratio and is given by

$$\hat{\Theta}_{ML} = \arg_{\Theta} \max F(\Theta, \Gamma)$$

$$F(\Theta, \Gamma) = -\frac{1}{2}[\Gamma - T(\Theta)]^T \Sigma^{-1} [\Gamma - T(\Theta)] \quad (6)$$

Assuming that each of the TDOAs are independently corrupted by zero-mean additive white Gaussian noise of variance σ_{ikj}^2 the ML estimate turns out to be a nonlinear least squares problem (in this case Σ is a diagonal matrix), i.e.

$$\hat{\Theta}_{ML} = \arg_{\Theta} \min [\tilde{F}_{ML}(\Theta, \Gamma)]$$

$$\tilde{F}_{ML}(\Theta, \Gamma) = \sum_{j=1}^S \sum_{i=1}^M \sum_{k=i+1}^M \frac{(TDOA_{ikj}^{estimated} - \hat{TDOA}_{ikj}^{actual})^2}{\sigma_{ikj}^2} \quad (7)$$

Since the solution depends only on pairwise distances, any translation, rotation and reflection of the global minimum found will also be a global minimum. In order to make the solution invariant to rotation and translation we select three arbitrary nodes to lie in a plane such that the first is at $(0, 0, 0)$, the second at $(x_1, 0, 0)$, and the third at $(x_2, y_2, 0)$. In two dimensions we select two nodes to lie in a line, the first at $(0, 0)$ and the second at $(x_1, 0)$. To eliminate the ambiguity due to reflection along Z-axis(3D) or Y-axis(2D) we specify one more node to lie in the positive Z-axis(in 3D) or positive Y-axis(in 2D). Also the reflections along X-axis and Y-axis(for 3D) can be eliminated by assuming the nodes which we fix to lie on the positive side of the respective axes i.e $x_1 > 0$ and $y_2 > 0$. Similar to fixing a reference coordinate system in space we introduce a reference time line by setting $tm_1 = 0$.

Problem Solution:

The ML estimate for the node coordinates of the microphones and loudspeakers is implicitly defined as the minimum of a nonlinear function. The solution is same as a nonlinear weighted least squares problem. The Levenberg-Marquardt method is a popular method for solving non-linear least squares problems. For more

details on nonlinear minimization refer to [6]. Least squares optimization requires that the total number of observations is greater than or equal to the total number of parameters to be estimated. This imposes a minimum number of microphones and speakers required for the position estimation method to work. Assuming $M=S=K$, Table 1 lists the minimum K required for the algorithm. In our experiments on real platforms we found the precision of localization to be on the order of several centimeters [8].

Table 1. Minimum value of Microphone Speaker Pairs (K) required for different estimation procedures (D-Dimension)

$K \geq$	$D = 2$	$D = 3$
TDOA Position Estimation	5	6
TDOA Joint Estimation	6	7

4. CONCLUSION

In this paper we addressed the problem of multimedia adaption from a new angle by introducing multichannel I/O into the adaptation framework. Arrays of sensors and actuators offer rich possibilities for future media applications and represent an additional dimensionality in the problem of multimedia adaptation. We have also demonstrated that arrays of I/O devices can be created out of a network of general purpose computing and communication devices such as laptops, PDAs, tablets, and that those arrays can perform tasks of self-synchronization and self-localization with a precision of tens of microseconds and several centimeters respectively. Researchers interested in using the common time and space infrastructure are encouraged to contact the authors for a research prototype of the system implemented for laptops with Intel®Centrino™Mobile Technology.

5. REFERENCES

- [1] R. Lienhart, I. Kozintsev, M. Yeung, and S. Wehr, "On the importance of exact synchronization for distributed audio signal processing," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2003)*, 2003, p. 147.
- [2] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Clock synchronization for wireless local area networks," in *IEEE 12th Euromicro Conference on Real-Time Systems (Euromicro RTS 2000)*, 2000, pp. 183–189.
- [3] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *5th Symposium on OS Design and Implementation*, Dec 2002.
- [4] P. J. Rousseeuw, "Least median-of-squares regression," *JACM*, vol. 79, pp. 871–880, 1984.
- [5] S. Egorychev, I. Kozintsev, R. Lienhart, D. Budnikov, I. Chikalov, "Providing common i/o clock for wireless distributed platforms," in *ICASSP2004*, May 2004.
- [6] Philip E. Gill, Walter Murray, and Margaret H. Wright, *Practical Optimization*, 1981.
- [7] Vikas Raykar, Igor Kozintsev, and Rainer Lienhart, "Position calibration of audio sensors and actuators in a distributed computing platform," in *ACM Multimedia 2003*, Nov 2003.