Signal Processing 107 (2015) 171–184

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

A combined hardware–software approach for acoustic sensor network synchronization

Joerg Schmalenstroeer *, Patrick Jebramcik, Reinhold Haeb-Umbach

Department of Communications Engineering, University of Paderborn, Germany

A R T I C L E I N F O

Article history:
Received 17 January 2014
Received in revised form
16 May 2014
Accepted 23 June 2014
Available online 14 July 2014

Keywords:
Gossip algorithm
Clock skew
Sensor network

A B S T R A C T

In this paper we present an approach for synchronizing a wireless acoustic sensor network using a two-stage procedure. First the clock frequency and phase differences between pairs of nodes are estimated employing a two-way message exchange protocol. The estimates are further improved in a Kalman filter with a dedicated observation error model. In the second stage network-wide synchronization is achieved by means of a gossiping algorithm which estimates the average clock frequency and phase of the sensor nodes. These averages are viewed as frequency and phase of a virtual master clock, to which the clocks of the sensor nodes have to be adjusted. The amount of adjustment is computed in a specific control loop. While these steps are done in software, the actual sampling rate correction is carried out in hardware by using an adjustable frequency synthesizer. Experimental results obtained from hardware devices and software simulations of large scale networks are presented.

1. Introduction

Sensor networks have been an area of active research for many years [1,2]. They have been widely used in various applications such as industrial process monitoring [3], environmental sensing [4], object tracking [5,6] and even underwater monitoring and communication [7].

One research topic of continued interest in sensor networks is the synchronization of the clocks of the distributed devices. A common time base is a prerequisite for many signal and information processing tasks. It allows us to distinguish between different events or multiple observations of the same event by the spatially distributed sensor nodes. Furthermore, energy saving approaches imply the availability of precisely timed sleeping periods and synchronous wake-up time slots.

There are well established methods for the synchronization of the nodes of a communication network to a centralized master processing unit, e.g., the precise time protocol (PTP) [8]. The PTP and other approaches build upon the exchange of time stamps using either a one [9] or a two-way [10] message exchange in a single-cast [11] or broadcast mode [12].

In a centralized processing, clock corrections of the slave nodes are initiated by a clock correction command issued by the master, which the slaves have to adhere to. However, a centralized processing has several disadvantages. It introduces a single point of failure (the master), all nodes have to communicate with the master resulting in a communication overhead, and it requires a strategy for selecting and removing the master node if two networks join or one network falls apart.

A distributed clock synchronization avoids these disadvantages. The authors of [13], for example, propose an approach for synchronizing a network by only locally exchanging time stamp information between neighboring nodes and by this establishing a global time across all nodes. A similar approach is analytically discussed in [14] where the authors show that their approach converges to the optimal estimates and that it reaches the Cramer–Rao

* Corresponding author. Tel.: +49 5251 603623.
E-mail address: schmalen@nt.uni-paderborn.de (J. Schmalenstroeer).

http://dx.doi.org/10.1016/j.sigpro.2014.06.030
0165-1684/

bound for all network topologies. In [15] a distributed Kalman filter is developed to estimate a virtual master clock. However, it requires a network in beacon-enabled mode with dedicated time-slots to work properly.

Beside the widely used one- or two-way message exchange algorithms an alternative approach has been published: closed-loop message exchange algorithms. Here, sequences of nodes forming closed-loops in the network are determined before the synchronization starts. Then the algorithms utilize the fact that the sum of all clock offsets within the closed-loops of different nodes ideally should sum up to zero [16]. In [17] three approaches for time synchronization are compared where two of them use message passing through closed-loops and the third one exchanges local information to synchronize the network.

Approaches employing closed-loops require the selection of reference nodes to provide global timing information and loop finding algorithms to build subsets of nodes in closed-loops, which makes it difficult to apply the approaches to dynamically changing network topologies. Local information exchanging algorithms (e.g., the diffusion algorithm in [13,14,17]) do not have this drawback and thus are more suitable for sensor networks with varying topologies.

The application area targeted in this publication is distributed audio signal processing. Distributed audio signal processing algorithms for beamforming [18] or source localization [19] require a tight sampling rate synchronization. For example, if the time difference of arrival of an impinging audio signal is to be estimated in the presence of a sampling frequency mismatch between two microphones, it cannot be decided whether an observed delay change is caused by a moving speaker or is just due to differences in the sampling frequencies as can be seen by the following example.

Let us assume a sensor network consisting of two nodes each equipped with a microphone at a distance of 5 cm. If the sampling frequency of the first node is 16 kHz, and the second node has a sampling frequency offset of 10 ppm compared to the first one, the data stream of the second node has additional 0.16 samples/s. Direction of arrival estimation approaches, e.g., GCC-PHAT [20], correlate windowed segments of the data streams to estimate the time difference of arrival between the recorded audio signals and, from that, the direction of arrival. An audio signal requires $0.05\ \text{m}/340\ \text{m/s} \times 16\,000\ \text{samples/s} = 2.35$ samples for the distance between the microphones. A non-moving speaker is detected as a moving source, since the data streams of both nodes diverge. Within only 2.35 samples/0.16 samples/s $= 14.7$ s the speaker is "moved" from the front (0°) to the end fire position (90°) just by the effect of the sampling frequency mismatch.

In audio applications, the estimation of the sampling frequency mismatch can either be accomplished at the baseband audio signal processing side by analyzing the sampled signals or by the exchange of time stamps via a communication link between the sensors [21]. Analyzing the audio signal has the benefit that no extra timing information has to be transported via the network, but it requires the availability and observability of some kind of effect resulting from the different sampling frequencies. For example in [22] the authors assume the existence of speech absence times with slow time-varying interference statistics. During the speech pauses the phase drift of the coherence function between the microphone signals is utilized to estimate the frequency offset. Another idea based on the assumption of motionless and spatially stationary sound sources is presented in [23]. In case of a sampling frequency mismatch the time difference of arrival (TDOA) between microphone signals changes, as if the sources are moving. The authors present a maximum likelihood estimator based on TDOA observations for the frequency mismatch, which is compensated by a linear phase shift in the short-time Fourier domain. Obviously, the assumptions about the input signals are quite restrictive for both approaches, and it is unclear how these approaches can be scaled to large sensor networks with many nodes.

The authors of [24] regard the problem of sampling frequency mismatch between recording and playback devices in acoustic echo cancelation tasks. In their approach the microphone signal is resampled and subsequently processed by an LMS algorithm. It alternatingly estimates a time stretching factor for resampling and the unknown transfer function between the loudspeaker and the microphone signal.

Here we are achieving a sampling clock synchronization of the A/D devices of the acoustic sensors by a message exchange over the (wireless) communication link, thus taking advantage of the many results on wireless sensor network synchronization discussed earlier.

We successfully used the method by Chaudhari [11,25] in our previous work for the synchronization of the sampling rate of a microphone to a master clock via a wireless link [26]. It is a simple and robust two-way, single-cast time stamp exchange algorithm which can be established even on computationally limited devices. As a key component for achieving high precision clock difference estimation we designed a Kalman filter, which was specifically designed to estimate the clock frequency and phase differences in the presence of unpredictable medium access delays and random hardware delays. In this paper we will present a more detailed view on the observation error and the design of the Kalman filter.

In [27] we extended the pair-wise clock synchronization to clock synchronization of a network comprising many nodes and arbitrary topologies. To this end a gossiping strategy [28] was adopted, such that a global synchronization is achieved by only local message passing between neighboring nodes. Here, we summarize the gossiping algorithm and give a full overview about the hardware and software components. We further discuss in detail the essential control mechanism for computing the correction values for the clocks of the sensor nodes.

The actual sampling rate and sampling phase corrections can be either accomplished by resampling the microphone signals or in hardware by adjusting the oscillators of the A/D devices. While the first can be carried out in software or in hardware, the latter is a pure hardware solution that does not incur any additional computational effort caused by resampling. It, however,