Ad Hoc Networks 24 (2015) 92–108

Contents lists available at ScienceDirect

Ad Hoc Networks

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

Adaptive time window linear regression algorithm for accurate time synchronization in wireless sensor networks

Juan J. Pérez-Solano *, Santiago Felici-Castell

Departamento de Informática, ETSE, Universitat de València, Avd. de la Universidad S/N, 46100 Burjassot, Valencia, Spain

ARTICLE INFO

Article history:
Received 18 June 2013
Received in revised form 1 August 2014
Accepted 4 August 2014
Available online 14 August 2014

Keywords:
Wireless sensor networks
Time synchronization
Linear regression
Adaptive time window

ABSTRACT

In this article we propose a new algorithm for time synchronization in wireless sensor networks. The algorithm is based on linear regression to achieve long-term synchronization between the clocks of different network motes. Since motes are built using low-cost hardware components, usually their internal local clocks are not very accurate. In addition, there are other effects that affect the clock precision, such as: environmental conditions, supply voltage, aging, manufacturing process. Because some of these causes are external and unpredictable, the clock drift between two motes can change in a random way. Due to these changes, the optimum time window used for performing the linear regression varies with time. The proposed time synchronization algorithm adjusts the resynchronization periods and the linear regression window size to these variations, minimizing the synchronization error. Our algorithm has been tested in real multihop network deployments and the results obtained show higher clock accuracy when compared to the related work.

1. Introduction

Time synchronization is of great importance in the design and implementation of Wireless Sensor Networks (WSN) [1, 2, 3], because the nodes within the network (also called motes) are usually required to perform distributed and collaborative tasks in a synchronized way, under severe constraints in terms of limited energy, bandwidth, memory and computational capabilities. Next we give different examples. For instance, time synchronization is needed in data gathering applications because after taking a sample the mote has to transmit this information to the base station, which is usually several hops away from the original mote. Since at every hop, the message containing the data suffers random delays when it is forwarded through the network, without a time mark with a network wide meaning it would be impossible to know exactly at what time the data was sampled. On the other hand, many protocols at the Medium Access Control (MAC) layer also require time synchronization in their operation. One example is the Sensor MAC (SMAC) [4], which is one of the most popular contention based MAC protocols for WSN. For reducing the mote's power consumption, the SMAC protocol proposes the implementation of virtual clusters of motes that are synchronized and share the same scheduling for the radio duty cycle. Synchronization is even more important in Time Division Multiple Access (TDMA) [3] MAC protocols, because in this case, each mote has to determine with a high precision the starting point of the time slots in which it has to receive or transmit a message in order to avoid collisions. Time synchronization is also necessary for carrying out other distributed tasks, such as: detection and classification of events, tracking of objects and data fusion [1].

For implementing time synchronization in WSN, many methods have been proposed mainly based on two

* Corresponding author.
E-mail addresses: juan.j.perez@uv.es (J.J. Pérez-Solano), santiago.felici@uv.es (S. Felici-Castell).

http://dx.doi.org/10.1016/j.adhoc.2014.08.002
1570-8705/

different techniques: *pulse-coupling* and *message transmission* [5]. *Pulse-coupling* synchronization uses the physical layer waveform to transmit the clock information [6]. The pulses of the radio signal are exchanged between the nodes to emulate the behavior of biological systems, that are capable of synchronizing its periodic activities in an autonomous way, e.g. the synchronous flashing of fireflies. These systems are modeled as a network of pulse-coupled oscillators that can converge to a common clock, even with a large number of nodes [6]. In *message exchange* synchronization, the motes exchange periodic messages with time marks that are used to determine the offset between their clocks. The protocols based on this technique can be more easily implemented with the wireless communication transceivers that are integrated in current motes. But since message transmission is usually affected by random delays, the protocol can only estimate the time difference between the clocks, with an accuracy that depends on the wireless link. Furthermore, this estimation becomes out of date after some time due to the clock drift, which makes it necessary to perform periodic retransmissions to maintain the synchronization accuracy.

Time synchronization accuracy does not rely only on the performance of the synchronization protocols but also on the mote clock hardware. Mote local time is kept in a microcontroller internal digital counter that is periodically increased using the impulses of a low-cost quartz crystal. The mote clock precision is limited by the quality of the quartz crystal, measured in number of oscillations per million that are gained or lost, and the counter resolution that depends on the clock signal period. Moreover, changes in the environmental conditions or the supply voltage, imperfections in the manufacturing process and aging can produce slight variations of the crystal oscillation frequency and degrade its accuracy.

The main contribution of this article is the proposal of a new synchronization algorithm based on message exchanges and linear regression methods that can adapt and optimize the time window and the *Resynchronization Periods (RP)* to the clock drift. The algorithm minimizes the synchronization error, taking into account the features of the mote's clock. Also, the algorithm is extended to be network-wide to achieve global synchronization in the WSN. Using the proposed algorithm, we can estimate with more accuracy and fewer packet exchanges compared with the related work, the time offset variation with a neighbor mote. The algorithm has been tested in real multihop network deployments and the results obtained show higher clock accuracy when compared to the related work, in the order of microseconds.

The organization of the article is as follows. Section 2 is devoted to the related work and the results achieved in previous work. In Section 3, we introduce the mote clock model and present the more important features of common synchronization algorithms. In Section 4, we provide an experimental study of the different issues and analysis of the main assumptions in time synchronization. In Section 5 we describe the proposed algorithms for pair-wise time synchronization, with the specification of the proposed synchronization protocols and its network wide extension. Experimental results are presented and discussed in Section 6. Finally, conclusions are given in Section 7.

2. Related work

There is a great collection of time synchronization protocols for distributed systems in the literature [1], although the strict requirements involved in the development of WSNs prevent the direct adaptation of traditional synchronization protocols to these networks. This is the case for example of the Network Time Protocol (NTP) [7] that is very common in Internet. This protocol uses an external clock source, like the Universal Time Controlled (UTC) or the Global Positioning System (GPS) [8], as the main reference to synchronize the rest of the network nodes. However, the use of GPS hardware is not suitable for the low-power operation required in WSN motes and usually access to external clock sources or Internet is not available in autonomous applications. Moreover, NTP does not take into account the particular features of the WSN wireless channels and it is not optimized to run in scenarios with severe resource constraints and power limitations. Another synchronization standard used in Internet is the IEEE 1588, called the Precision Time Protocol (PTP). It was conceived for reaching a synchronization precision in the scale of 100 ns, outperforming the NTP accuracy. The main objection to this protocol is that it requires *RP* of only a few seconds to achieve such precision [9], a fact that would significantly increase mote power consumption.

For WSN there is also a great amount of research published specifically on time synchronization. The particularities of this technology impose serious limitations on these protocols [2] and as it is formally stated in [10], we can find solution to the clock synchronization under certain conditions. In particular [10] shows for an Ideal Scenario (IS) of unknown but constant delays and clock skew that we can always determine the clock skew between two motes, as the ratio of the time intervals for each mote between two time marks using packet exchanges. Thus, the development of simple algorithms with a low computational complexity and reduced power consumption is necessary, but with a high accuracy at the same time. This is difficult to accomplish and even more when the mote hardware is unreliable and the wireless communication links are quite unstable. At a first approach, time synchronization is implemented estimating the time difference (offset) between the local clocks of two or more motes at a certain instant, also called instantaneous synchronization protocols, where neighboring nodes carry out an individual estimation of the offset between their local clocks. A well-known procedure for performing instantaneous time synchronization in WSNs is the two-way message exchange mechanism between two motes, also referred to as the sender–receiver algorithm. Timing-sync Protocol for Sensor Networks (TPSN) [11] and Lightweight Time Synchronization (LTS) [12] protocols are examples of sender–receiver algorithms. This is a pair-wise synchronization algorithm in which two messages are exchanged, each one sent in an opposite direction, to capture four time marks with which the offset is estimated. The algorithm