

Demo Abstract: Design and Implementation of a PCO-based Protocol for Sensor Networks

Roberto Pagliari
rp294@cornell.edu

Anna Scaglione
anna@ece.cornell.edu

Department of Electrical and Computer Engineering
Cornell University, Ithaca - NY, USA

Abstract

Sensor networks have been used in a wide range of applications. Considerable research effort is currently devoted to design protocols that allow networks of inexpensive sensors to perform reliable remote control and monitoring functions, in spite of the limitations of each device.

Our objective is to demonstrate a fully software implementation of the so called Pulse Coupled Oscillator Protocol, proposed in [2, 6] for the decentralized synchronization of radio devices. The PCO protocol is inspired by a model found in mathematical biology [5]. It is decentralized, scalable and simple, as we will show in our demonstration.

Categories and Subject Descriptors

H.4.m [Information Systems Applications]: Communications Applications

General Terms

synchronization, biologically inspired algorithms, pulse coupled oscillators, fireflies

Keywords

sensor networks, synchronization

1 Introduction

Many synchronization algorithms have been proposed in literature for sensor networks. Most of them are derived from traditional wired/wireless networks and based on flooding algorithms [8, 4]; more specifically, in these protocols one or more seed nodes disseminate a message containing a common clock reference to all the other nodes over the network through a message forwarding algorithm. A different approach is taken in algorithms such as RBS [1], where the nodes maintain a local clock, and learn the clock difference respect to their neighbors, by measuring the time at which they registered a common reference signal sent by a sink node.

A "biologically inspired" algorithm is the Pulse-Coupled-Oscillator (PCO) [5]. Inspired by simultaneous flashing of large fireflies aggregation, first noticed in south-east Asia, the PCO algorithm is a fast, decentralized, synchronization mechanism.

2 Pulse coupled oscillators

The code to implement the PCO protocol is extremely simple. The node increases at each tic a state variable as follows

$$x_i(k) = x_i(k-1) + c + \epsilon\delta[k]$$

if $x_i(k-1) + c + \epsilon\delta[k] < 1$ ($x_i(k) = 0$ else), where $\delta[k] = 1$ if a packet is received at the k -th tic, and ϵ is the coupling strength. If $x_i(k-1) + c \geq 1$ the node sends a message at tic k .

If $x_i(k-1) + c + \epsilon\delta[k] \geq 1$, but $x_i(k-1) + c < 1$ the node waits another cycle prior to sending a packet.

The packet lasts for $952\mu s$, during which the transmitting node cannot receive. This duration, called *refractory time*, is the fundamental limit in the accuracy of the synchronization.

The message sent is a packet containing the following fields: i) Preamble and Start Frame Delimiter (SFD) [4 + 1 = 5 bytes], ii) MAC payload length field and a fake MAC address without additional payload [1 + 9 = 10 bytes].

It is actually unnecessary to decode the message since what we need is to just detect its presence, by detecting the Preamble and the SFD.

The main problem one faces using the PCO in a wireless network is that the commercial radio-frequency transceivers and drivers are specifically designed to perform collision resolution and avoidance protocols, to protect the integrity of the message payload. In contrast, pulse coupled oscillators make use of signal superpositions to speed the convergence time, and therefore it is impossible to observe good performance on PCO protocols that are implemented above a MAC layer, since the MAC is trying to prevent what the PCO is trying to attain, which is the coalescence of all transmissions at a single point in time.

3 Contribution

We implemented a new radio driver that allows a node to behave as a pulse coupled node, eliminating the medium access contention phase. Each node, while performing the PCO algorithm, sends a message when its phase reaches the maximum value, i.e., $x = 1$ and pull up its phase whenever a pulse has been received. The developed radio stack is entirely software, based on the micaZ platform under the TinyOS environment, and uses the common radio transceiver Chipcon CC2420 to achieve synchronization. It is not required any additional hardware, since the radio is provided

with the node platform.

Other implementations of the PCO in a testbed have been proposed [9, 3]. Unlike CMU, our solution does not add new hardware while preserving large part of the benefits of scalability that were highlighted in [2]. The protocol implementation performs better than the implementation in [9]. The reasons for these gains is that in our implementation we completely restrained the MAC protocol from applying mechanisms for collision resolution. This is because, as explained in [7], PCO naturally leads the signals of the nodes to coalesce in time and the superposition of signals from nodes that are mutually synchronized is actually beneficial to speed up the convergence.

Our software implementation supports the presence of both regular radio and PCO radio drivers. Hence, it is possible to perform the PCO and use, before or afterwards, the regular CSMA-based radio drivers to exchange messages for a custom application. The most important point, and, probably, a disadvantage, is that the PCO cannot run at the same time as the regular radio, since they both share a common hardware device.

The testbed deployment is shown in Fig. 1. A laptop, or a regular pc, is connected to the sensor network through a gateway, the access point. A small number of routers is used to send a start message over the network, since the gateway itself cannot reach, in most of cases, all the sensors.

After the start message has been delivered, each mote generates a initial, randomly chosen, phase and the PCO starts. The regular radio drivers are shut down, and each node starts behaving as a PCO oscillator.

The PCO process works for a fixed number of iterations, after which the motes turn to the CSMA-based MAC protocol. Hence, it is possible to retrieve feedback data regarding the accuracy of the synchronization protocol.

Through the gateway and the routers it is possible to send a reference signal, after which every node record the actual value of its phase, and then send that value to the access point by multihop transmissions.

A Windows graphical user-friendly interface is available to interact with the network. It is possible to check if the nodes are synchronized, and the accuracy of the protocol. Some fundamental parameters, as the coupling strength, the evolution function shape, and the number of iterations can be set directly from that GUI, and they are automatically sent to the motes.

4 Conclusions

Our preliminary results indicate that the reachable accuracy of the PCO-based protocol we implemented is of the order of $500\mu\text{s}$ with respect to a network composed of 100 motes, 40 iterations each. We are going to evaluate the performance of our proposed protocol in order to characterize the performance as a function of the parameters choice.

5 References

[1] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[2] Y. Hong and A. Scaglione. A scalable synchronization protocol for large scale sensor networks and its applica-

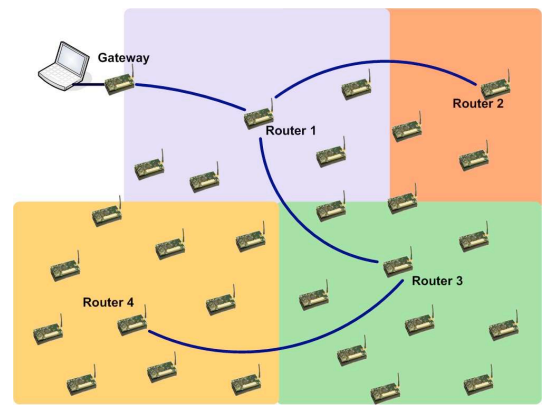


Figure 1. Testbed deployment.

tions. *IEEE Journal on Selected Areas in Communications*, 23(5):1085–1099, 2005.

[3] R. Mangharam, A. Rowe, and R. Rajkumar. Firefly: A cross-layer platform for wireless sensor networks. *Real Time Systems Journal, Special Issue on Real-Time Wireless Sensor Networks*, 2007, to appear.

[4] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM Press.

[5] R. Mirolo and S. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM J. Appl. Math.*, 50(6):1645–1662, 1990.

[6] A. Scaglione and Y. Hong. Cooperative models for synchronization, scheduling and transmission in large scale sensor networks: an overview. In *CAMSAP 2005, Puerto Vallarta, Mexico*, 2005.

[7] A. Scaglione and R. Pagliari. Non-cooperative versus cooperative approaches for distributed network synchronization. In *Percom 2007, NY*, 2007.

[8] W. Su and I. Akyildiz. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 13(2):384–397, 2005.

[9] G. Werner-Allen, G. Tewari, A. Patel, R. Nagpal, and M. Welsh. Firefly-inspired sensor network synchronicity with realistic radio effects. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (Sensys'05)*, 2005.