Engineering Applications of Artificial Intelligence 45 (2015) 361–375

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Survey Paper

Survey on synchronization mechanisms in machine-to-machine systems

Iva Bojic [a,*], Kristian Nymoen [b]

[a] Massachusetts Institute of Technology, Department of Urban Studies and Planning, 77 Massachusetts Ave, Cambridge, United States
[b] University of Oslo, Department of Informatics, Postboks 1080 Blindern, 0316 Oslo, Norway

ARTICLE INFO

Article history:
Received 13 November 2014
Received in revised form
5 May 2015
Accepted 8 July 2015
Available online 5 August 2015

Keywords:
Biological inspiration
Fireflies
Pulse coupled oscillator model
Real-world environments
Self-organization

ABSTRACT

People have always tried to understand natural phenomena. In computer science natural phenomena are mostly used as a source of inspiration for solving various problems in distributed systems such as optimization, clustering, and data processing. In this paper we will give an overview of research in field of computer science where fireflies in nature are used as role models for time synchronization. We will compare two models of oscillators that explain firefly synchronization along with other phenomena of synchrony in nature (e.g., synchronization of pacemaker cells of the heart and synchronization of neuron networks of the circadian pacemaker). Afterwards, we will present Mirollo and Strogatz's pulse coupled oscillator model together with its limitations. As discussed by the authors of the model, this model lacks of explanation what happens when oscillators are nonidentical. It also does not support mobile and faulty oscillators. Finally, it does not take into consideration that in communication among oscillators there are communication delays. Since these limitations prevent Mirollo and Strogatz's model to be used in real-world environments (such as Machine-to-Machine systems), we will sum up related work in which scholars investigated how to modify the model in order for it to be applicable in distributed systems. However, one has to bear in mind that there are usually large differences between mathematical models in theory and their implementation in practice. Therefore, we give an overview of both mathematical models and mechanisms in distributed systems that were designed after them.

1. Introduction

In centralized systems, there is no need to use synchronization mechanisms. Namely, every application that is being run on one device, which represents one example of a centralized system, can send a request for the current time to the device system clock. Since there is only one *physical clock* in the system, time in centralized systems is always unambiguous. When a decentralized architecture for clocking circuits was devised, it was realized that in practice it was not possible to manufacture two exactly identical clocking circuits (i.e., physical clocks). Since these circuits were different, they could easily drift seconds per day,[1] accumulating significant errors over time. Because of such imperfections of physical clocks, in distributed systems a *common notion of time* does not exist.

Many applications in distributed systems (such as Machine-to-Machine systems) rely on a common time that all devices can appeal to. Depending on a type of an application, sometimes this common notion of time should also be *global*, and sometimes it is enough that it is only *local*. If applications rely on a global notion of time, then devices should become synchronized with an external source of standard time (e.g., Coordinated Universal Time, International Telecommunication Union, 2002). Otherwise, they can reach a common consensus on time locally, i.e., independently of an external source of time. The process of achieving a common *synchronized* notion of time, which can be either local or global, is called *time synchronization* and mechanisms that are used to achieve that goal are called *synchronization mechanisms*.

Some applications do not require devices to become synchronized with an external time source nor to reach consensus on common time. Still, it may be required that consensus is reached on some common value other than time (e.g., a firing period or a phase). In such cases, devices can achieve *synchrony*. In this paper we will use the same name for mechanisms that are being used to achieve synchrony as we did for the mechanisms used to achieve time synchronization: *synchronization mechanisms*. The difference between achieving common notion of time and common notion of some other value is that when reaching consensus on time, there is a set of rules that have to be obeyed (Sundararaman et al., 2005). Particularity, there is a rule saying that during a time synchronization process, time should never run backwards.

Time not being able to run backwards has certain implications. For instance, when synchronization mechanisms are used to synchronize the physical clocks of devices, there is a possibility

* Corresponding author.
E-mail addresses: ivabojic@mit.edu (I. Bojic), krisny@ifi.uio.no (K. Nymoen).

[1] More about reasons why physical clocks drift apart can be found in Tanenbaum et al. (2001), Coulouris et al. (2005) and Kshemkalyani and Singhal (2008).

http://dx.doi.org/10.1016/j.engappai.2015.07.007
0952-1976/© 2015 Elsevier Ltd. All rights reserved.

that some time moments are skipped causing a discontinuity of their physical clocks. Consequently, the risk of errors in other mechanisms that rely on readings of physical clocks is increased. One example would be a mechanism that is used as a trigger for some action: the time moment of a scheduled action could be skipped because of a time synchronization process, and the action would not occur as planned.

Fig. 1 gives an example of how time shown by some physical clock (shown on *y*-axis) changes over time compared to the global time change (shown on *x*-axis). We can see that if some action is scheduled for the time moment when the physical clock shows 18, then due to a time synchronization process, this time moment is skipped and consequently the scheduled action is not performed on time. In order to avoid a physical clock discontinuity, we can use synchronization mechanisms for time synchronization of *virtual clocks*. A virtual clock is a function that performs mapping from a physical clock time to a set of *virtual timestamps*. A process of achieving time synchronization of virtual clocks does not affect the time shown by physical clocks, thus avoiding the problem of a physical clock discontinuity.

Although there is a clear difference between synchrony and time synchronization, in the rest of the paper we will mostly use the term time synchronization when talking both about time synchronization and synchrony. The reason why we are going to do that is because once when we achieve time synchronization, it is easy to achieve synchrony as well, and also vice versa. Moreover, as mentioned earlier, we use the term synchronization mechanisms both when talking about mechanisms for time synchronization and synchrony.

Besides applications whose prerequisite is a common notion of time of all devices in the system, there are also applications in which it is only important to provide an ordering of all events in the system. Although the mechanisms used in such applications are not the main focus of this paper, synchronization of *logical clocks* should be mentioned as an alternative to mechanisms for synchronizing the physical or virtual clocks of devices. A logical clock is a mechanism for capturing chronological and causal relationships between different events in the system. Here it is not important *when* some event occurred, but to know the *order* in which all events in the system occurred.

A time synchronization process is not an instant process. Its duration depends on a synchronization mechanism *convergence rate*. We can thus define a *convergence time* as the total time required to achieve time synchronization. We also have to be very careful when claiming that time synchronization is achieved. Namely, even when devices are synchronized, their physical or virtual clock values are not completely the same (i.e., there are always some errors). We thus define two types of a *time synchronization precision*: an *absolute precision* and a *relative precision*. The absolute precision can be defined for synchronization mechanisms that use external sources of time as the maximum error of physical clock values of devices with respect to an external source of standard time. On the other hand, the relative precision is defined for synchronization mechanisms in which devices want to achieve a local common notion of time as the maximum deviation among physical or virtual clock readings of all devices in the system.

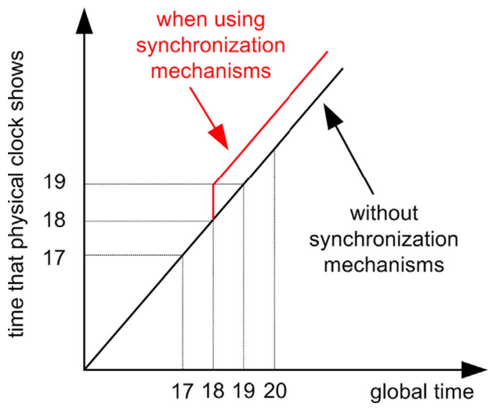

Fig. 1. One example of a physical clock discontinuity.

Finally, for the same reasons why it is necessary to use synchronization mechanisms in the first place (i.e., the imperfections of physical clocks), it is also important to repeat time synchronization process after its *lifetime* expires. A time synchronization lifetime is a time period during which the required time synchronization precision is maintained. After that time, the consistency of device clock readings is not supported anymore (i.e., devices are no longer synchronized).

Machine-to-Machine (M2M) systems are composed of networked computational devices, where the heterogeneity of the devices makes it difficult to implement system-specific mechanisms, for instance for synchronization. One example of an M2M system with the prerequisite of device synchrony or consensus on a notion of time is applications in which devices have to share a common resource (e.g., a shared communication channel in wireless networks). It is also important to maintain the consistency of data that is stored in distributed databases. Moreover, in applications in which data is collected distributedly, it is important to know that all data is collected simultaneously. More about the reasons why time synchronization is required in distributed systems can be found in Liskov (1991).

The rest of this paper is organized as follows. Section 2 gives an overview of characteristics of M2M systems and efforts that have been put into their standardization. In this section we also propose a taxonomy that can be used in such systems for classification of different synchronization mechanisms. Section 3 shows how biological organisms that evolve, self-organize, self-repair, navigate, and flourish can be used as an inspiration for various mechanism designs in distributed systems. This section also covers Mirollo and Strogatz's pulse coupled oscillator model, which explains firefly synchronization in nature, together with its limitations. Section 3 is followed by the section in which we give an overview of research that was conducted after Mirollo and Strogatz. Namely, scholars investigated how to modify their model in order for it to be applicable in distributed systems. Finally, Section 5 concludes the paper.

2. Machine-to-machine systems

Since there is no one unique definition of Machine-to-Machine (M2M) systems, we will mention several definitions that are used:

- M2M is a term used to describe the technologies that enable computers, embedded processors, smart sensors, actuators and mobile devices to communicate with one another, take measurements and make decisions — often without human intervention (Watson et al., 2004).
- M2M communication is the automated exchange of information between technical end devices such as machines, vehicles or containers and a central control center. M2M allows machines to exchange information without the manual assistance of humans (SingTel Machine-to-Machine (M2M), 2013).
- The role of M2M is to establish the conditions that allow a device to (bidirectionally) exchange information with a business application via a communication network, so that the device and/or application can act as the basis for this information exchange (Darmois and Elloumi, 2012).