Firefly Synchronization with Asynchronous Wake-up

Dan Alistarh
MIT

Alejandro Cornejo
Harvard

Mohsen Ghaffari
MIT

Nancy Lynch
MIT

Abstract

This paper considers biologically-inspired synchronization algorithms for a model with primitive communication and asynchronous wake-up. Existing averaging-style algorithms for synchronization implicitly require a coherent round structure to guarantee convergence. We argue that this assumption is too strong in a distributed setting, and thus investigate the synchronization problem assuming nodes wake up asynchronously. Our preliminary results show that, under some assumptions, it is possible to approximate the properties of a coherent round structure given asynchronous wake-up, and leverage this observation to achieve synchronization in this setting. Achieving general solutions remains an interesting open question.

1 Introduction

The emergence of synchronous behavior in groups of fireflies [13, 14] is a natural phenomenon which has garnered significant interest from the scientific community. Some research groups have studied the synchronization of fireflies from a biological perspective, e.g., [4], while others have taken a mathematical modeling approach, e.g., [9, 10]. Seminal work by Peskin [11], generalized by Mirollo and Strogatz [9], presented mathematical models which explain the convergence behavior of fireflies. Considerable subsequent research extended their models and used these insights to design efficient synchronization algorithms for systems such as wireless ad-hoc networks.

Early work considered *continuous-time* models, where the agents are modeled as "integrate-and-fire" oscillators [11, 9], where the evolution of each agent is described by a differential equation. The collective dynamics are expressed as a linear dynamical system defined in terms of the communication graph [10]. The behavior of such systems is now relatively well understood, and it has been shown to be related to the convergence of consensus algorithms [3], and to the collective behavior of flocks and swarms [10].

More recent work [8, 12] considers *discrete-time* models, where agents are allowed to emit pulses at continuous points in time, and update their state at the end of rounds of fixed length. Similar to work on continuous-time models, most discrete-time algorithms [8, 12], use some form of *averaging* to guarantee synchronization.

Specifically, the canonical form of an averaging-style synchronization algorithm, e.g., [10], is as follows. Every node i starts with an arbitrary time offset $t_i(0)$. During each round k, which is of fixed length T, every node i emits a pulse at time $t_i(k)$, and records the times when it received the pulses of each of its neighbors. At the end of round k, every node i sets the next transmission time $t_i(k+1)$ to be the (possibly weighted) average of the times where itself and its neighbors emitted a pulse in round k. The goal is *synchronization*: eventually, all nodes must beep with the same period, and in the same time slot.

This paper revisits the synchronization problem in a discrete-time model with asynchronous wake-up. We first notice that most known algorithms, e.g. [10, 12], assume a *coherent round structure*: in each round,

each node receives a single pulse/message from each of its neighbors. However, in an asynchronous wake-up model, this coherent round structure is not necessarily present, which can prevent averaging algorithms from converging. We then present a time-efficient averaging algorithm which ensures synchronization under asynchronous wake-up.

Asynchronous wake-up is a natural requirement for synchronization—if nodes wake up at the same time, they can simply decide to beep in the first slot of every period, trivially solving the problem. On the other hand, asynchronous wake-up times may cause a node to receive multiple messages, or no messages, from its neighbors in some rounds. Since rounds are no longer communication-closed, the linear equations used to characterize the system behavior are no longer meaningful, and in fact the algorithm may not converge. Figure 1 describes such an execution for the simple case when the communication graph is a clique. (The reader is also encouraged to try the online version of our simulator [1].)

It appears that divergence is caused by breaking the following assumption: *in each period, each node must perceive exactly one message from each of its neighbors*. In a model where nodes are allowed to communicate by sending full messages (instead of pulses), it would be possible to ensure this assumption by appending identifiers or counters to the nodes' messages. Unfortunately, these strategies are not applicable in primitive communication models such as the one we consider.

These considerations motivate us to investigate the existence of averaging algorithms for the synchronization problem, that work in a primitive communication model, and support asynchronous wake-up.

Specifically, we consider a multi-hop network where time is divided into *slots*, and at every slot each node can decide to either listen or beep. A node that beeps receives no feedback in that slot, and a node that listens in a slot can only distinguish between silence (no neighbor beeped) or the presence of a beep (one or more neighbors beeped). The beeping model has been considered previously in the context of wireless deployment [5], and biologically-inspired algorithms [2]. Initially, all nodes are asleep, and they are woken up at a particular slot either by an adversary, or when one of their neighbors beeps for the first time (whichever happens first). This asynchronous wake-up model is standard in the clock synchronization literature, e.g., [6, 7].

Our preliminary work finds that it is possible to design a correct averaging synchronization algorithm, assuming that the period T is at least $4n$, where n is the number of nodes. Moreover, the algorithm converges in $O(TD)$ time slots, where D is the diameter of the communication graph. We also note that a simple "adopt-minimum" strategy works in $O(TD)$ time given the same model assumptions.

In sum, our results suggest that there exist simple algorithms that converge under asynchronous wake-up. Several interesting questions remain open.

- Our solutions make two strong assumptions: that a node automatically wakes up its neighbors by beeping, and that the period length T is $\Omega(n)$. Both assumptions should be relaxed. In particular, we conjecture that there exist averaging algorithms which work for lower values of T and allow for asynchronous wake-up. (Lower values of T would also imply faster convergence.)

- Initial synchronization is a prerequisite for many distributed algorithms. It is therefore surprising that no solution is known for synchronization in the beep model, even for specific values of n and T. For instance, in the $T = 2$ case, nodes are simply required to beep in one of every two time slots.

- Even though extremely efficient clock synchronization algorithms are known, e.g. [7], these solutions tend to be extremely complex. On the other hand, biologically-inspired algorithms are usually simple, and work under minimal assumptions. Thus, it would be interesting to see if an efficient variant of the averaging algorithm can be shown to have fast convergence under the standard assumptions of the gradient clock synchronization problem [6], such as clock drifts and different period lengths.

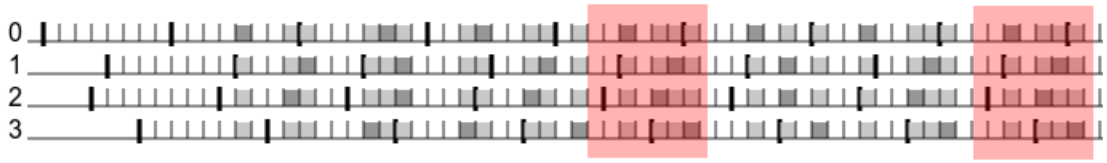

Figure 1: An execution with four nodes connected in a clique with period of eight slots, running an averaging algorithm. The slot is marked white if the node listens and hears silence, gray if it listens and hears a beep, and dark gray if the node beeps. The two segments highlighted in red show a cycle in the execution, which implies that the algorithm will not converge to a solution.

- Many averaging algorithms support *dynamic* communication graphs seamlessly [10]. It would be interesting to identify a suitable dynamic graph model for the synchronization problem, and to devise algorithms which support such a dynamic model.

2 Model and Problem Statement

We assume time is divided into *slots*, where slot boundaries are synchronized across nodes. (This assumption can be relaxed to consider the case when slot boundaries are not aligned across nodes, at the cost of a small constant blowup in the running time [5].) We assume a system of n nodes. Initially, every node is asleep, and it is woken up either at the first slot when one of its neighbors emits a beep, or at an arbitrary slot chosen by an adversary (whichever comes first). These wake-up assumptions are consistent with those in the clock synchronization literature [6, 7].

We model the communication network with a connected graph $G = (V, E)$, where V denotes the set of $n = |V|$ nodes and E denotes the set of undirected edges between nodes. We define $N(v) = \{u \mid (u, v) \in E\}$ as the set of neighbors of a node $v \in V$, and we use Δ and D to denote the maximum degree and the diameter of G respectively.

Nodes communicate with each other using *beeps*. In each slot, a node can either list or beep (emit a pulse). A beeping node receives no feedback, and a node v that listens at slot i can only distinguish between silence (all nodes in $N(v)$ listened at slot i) or the presence of a beep (one or more nodes in $N(v)$ beeped at slot i).

Informally, the synchronization problem requires that every node eventually agrees on a frequency and offset in time. Formally, we say the system reached a synchronized state at slot i, if there exists positive integers i_0 and $T > 1$ such that for every slot $j \geq i$ we have all nodes beeping if $(j - i_0) \mod T \equiv 0$, and all nodes listening otherwise.

In the definition above, T is the *period* of synchronization, and it counts the number of slots between consecutive beeps. We remark that the algorithm presented in this paper requires that $T \geq 4n$ and that T should be known a priori by every node.

3 An Algorithm for the Synchronization Problem

Description. The algorithm is composed of two phases, the wake-up phase, and the averaging phase. We assume that the parameter $T \geq 4n$ is fixed and known by all nodes. Each node i divides time locally in periods of length T, starting at the time when the node wakes up.

The Wake-up Phase. Recall that each node v wakes up either when woken up by the adversary, or just after receiving a first beep from a neighbor. Upon wake-up, each node beeps in the next slot, which wakes up its sleeping neighbors. It then sets the beginning of its period to be this slot, and its first beeping in the synchronization phase $\tau_v(0)$ to be at slot $T/2$. (For simplicity, we assume that T is even.) For example, node v wakes up in global slot ω_v, then beeps in slot $\omega_v + 1$, and sets its next beeping time in slot $\omega_v + T/2$.

The Synchronization Phase. In the second phase, nodes converge on a single slot. (See Algorithm 1 for the pseudocode.) During every period $k \geq 0$, each node i beeps once at time $\tau_i(k)$, and listens for the rest of the period. Time slots in which beeps are detected are marked in an array *buffer* of length T.[1] The only computation is the update of the beeping time for the next period. To determine $\tau_i(k+1)$, the process averages the times in which it has heard beeps during the period (including its current offset). Recall that multiple neighbors may broadcast during the same slot, in which case the node only perceives one beep. Once synchronized, a node will no longer update its offset.

$\tau_i(0) \leftarrow 1$;
for *each period* $k \geq 0$ **do**
  buffer $\leftarrow \emptyset$;
  for *slots* $j = 1$ *to* T **do**
    if $\tau_i(k) = j$ **then**
      beep;
      $buffer[j] \leftarrow 1$;
    else
      listen;
      $buffer[j] \leftarrow$ (0 if silence, 1 otherwise);
  count $\leftarrow$ number of 1's in the *buffer*;
  $\tau_i(k+1) \leftarrow \lfloor (1/count) \sum_{j=1}^{T} (j \cdot buffer[j]) \rfloor$;

Algorithm 1: The synchronization algorithm at node i.

Analysis Outline. Note that thanks to the wake up phase, each two neighbors wake up within at most one slot of each other and thus, all nodes beep wake up after at most D slots. We argue that throughout the course of algorithm, for each period k, each two neighboring nodes beep either in the same slot or in two consecutive slots. Then, we use a potential-function to show that the beeping time $\tau_v(k)$ of each node v for period k decreases essentially linearly with period number k, until v gets synchronized with the beeps of the first node (or nodes) that woke up. Thus, all nodes synchronize in at most $O(D)$ periods. We defer the proof to Appendix A.2.

Theorem 3.1 (Synchronization)**.** *For Algorithm 1, when $T \geq 4n$, for each period $k \geq 2D + 1$ and each two nodes v, u, we have $\omega_v + \tau_v(k) = \omega_u + \tau_u(k)$. That is, after the first $2D + 1$ periods, all nodes beep in the same slot of every period.*

The "Adopt-Min" Algorithm. We remark that, while our analysis shows that in this setting the averaging algorithm converges and nodes synchronize, an even simpler algorithm achieves the same result with an almost-trivial proof. (On the other hand, averaging algorithms are still interesting on their own, as research has indicated them to match the behavior of many biological systems, e.g., fireflies, e.g., [9, 10].) Consider the protocol that includes the wake-up phase as described above, but instead of the synchronization, makes each node v set $\tau_v(k+1) = \min\{j | buffer[j] = 1\}$ in each period. That is, v updates its beeping time to the minimum observed beeping time. Clearly, the first node that wakes up will always beep in the middle of its periods (in slot $T/2$ of each period), and all the other nodes will decrease their beeping times gradually until they synchronize with the node that woke up first. Considering the shortest path from the first awaken node to each other node, it is easy to see that global synchronization occurs after at most D periods.

[1] The node discards any beeps in the second slot after its wake-up, since these are wake-up phase beeps for the neighbors that just woke up.

References

[1] Simulator for a clique network with random wake-up times in the first period, available at: http://people.csail.mit.edu/acornejo/p/beepsynch/ (for best results, please use the chrome web browser).

[2] Yehuda Afek, Noga Alon, Ziv Bar-Joseph, Alejandro Cornejo, Bernhard Haeupler, and Fabian Kuhn. Beeping a maximal independent set. In *DISC*, pages 32–50, 2011.

[3] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and distributed computation*. Prentice Hall, 1989.

[4] John Buck. Synchronous rhythmic flashing of fireflies. *Quarterly Review of Biology*, 1988.

[5] Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In *DISC*, pages 148–162, 2010.

[6] Rui Fan and Nancy A. Lynch. Gradient clock synchronization. *Distributed Computing*, 18(4):255–266, 2006.

[7] Christoph Lenzen, Thomas Locher, and Roger Wattenhofer. Tight bounds for clock synchronization. *J. ACM*, 57(2), 2010.

[8] Dennis Lucarelli and I-Jeng Wang. Decentralized synchronization protocols with nearest neighbor communication. In *SenSys*, pages 62–68, 2004.

[9] Renato E. Mirollo, Steven, and H. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM J. Appl. Math*, 50:1645–1662, 1990.

[10] Reza Olfati-Saber, J. Alexander Fax, and Richard M. Murray. Reply to "comments on "consensus and cooperation in networked multi-agent systems"". *Proceedings of the IEEE*, 98(7):1354–1355, 2010.

[11] Charles S. Peskin. *Mathematical aspects of heart physiology*. 1973.

[12] Osvaldo Simeone, Umberto Spagnolini, Yeheskel Bar-Ness, and S Strogatz. Distributed synchronization in wireless networks. *Signal Processing Magazine, IEEE*, 25(5):81–97, 2008.

[13] Hugh M. Smith. Synchronous flashing of fireflies. *Science*, 82(2120):151–152, 1935.

[14] Steven H. Strogatz. *Sync: The Emerging Science of Spontaneous Order*. Hyperion, 1st edition, March 2003.

A Analysis of the Averaging Algorithm

In this section, we give a proof of convergence for the averaging algorithm. We first prove some properties of the round structure, and then proceed to prove the convergence of the algorithm.

A.1 The Round Structure

We start by establishing a few properties of the algorithm's round structure. We start from the simple observation that, given the adversarial wake-up procedure from Section 2, all node wake-up times are between 0 (the earliest wake-up time of a process), and D, the diameter of the graph.

Lemma A.1. *For every node v, $0 \leq \omega_v \leq D$.*

Proof. Recall that the wake-up time for node v, ω_v, is defined as the minimum between the wake-up time for the node decided by the adversary, which we denote by α_v, and the time when one of its neighbors first beeps. On the other hand, the wake-up time for each node v is upper bounded by h_v, the number of hops between the first node to wake up and node v, in the communication graph. Clearly, $h_v \leq D$. □

Next, we prove that the following assumption holds.

Assumption 1. *In every period, every node perceives exactly one message from each of its active neighbors.*

First, notice that, since the period T is at least $4n$, then the maximum difference between two wake-up times is at most $T/4$. We now show that this is a sufficient condition for Assumption 1 to hold throughout an execution of the algorithm.

Lemma A.2. *Given $T \geq 4n$, Assumption 1 holds throughout the synchronization phase.*

Proof. Consider an execution from which we erase the beeps pertaining to the wake-up phase. We prove that Assumption 1 holds throughout the execution by induction on the period number k. The claim is that, for every period $k \geq 0$, every node beeps between (global) slots $kT + T/2$, and $kT + 3T/4$. Notice that this claim implies the Lemma, since this time interval cannot overlap with slots from a previous of subsequent phase, for any node i, given that every node wakes up between time 0 and $T/4$.

For the first period, notice that, by Lemma A.1, every node beeps between global times $T/2$ and $T/2+D$ in period 0 of the synchronization phase. Since $D < n \geq T/4$, we obtain that every node beeps between global times $T/2$ and $3T/4$, i.e. $\tau_i(0) \in [T/2, 3T/4]$, for all nodes i. Notice that this interval cannot overlap with the interval of the second period for any node i, therefore the base case holds.

The induction step follows since, for each node i, the beeping time in period $k+1$ is a convex combination of the beeping times in the previous period k. □

A.2 Proof of Convergence

Theorem A.3. *After $2D + 1$ periods from the time that the first node wakes up, all nodes are synchronized.*

Proof. Let s_0 be the node that wakes up first (or one such node, if more exist). For each node v and each k^{th} period, let $x_v(k) = (\tau_v(k) + \omega_v) - (\tau_{s_0}(k) + \omega_{s_0})$. Note that $x_v(k) \in [0, D]$. We first show that

$$\forall k \in \mathbb{N}, v, u \in V \text{ such that } u \in \mathcal{N}(v), \text{ we have } |x_v(k) - x_u(k)| \leq 1.$$

We prove this by induction on k. The base case $k = 1$ holds because each awake node wakes up all its neighbors which gives $|\omega_v - \omega_u| \leq 1$, and $\tau_v(1) = \tau_u(1) = T/2$. Assume the claim for period $k-1$. From this, we get that $\forall v' \in V, x_{v'}(k) \in [x_{v'}(k-1) - 1, x_{v'}(k-1)]$. Thus, if $x_v(k-1) = x_u(k-1)$, we are done. Otherwise, and without loss of generality, suppose $x_v(k-1) = x_u(k-1) - 1$. Since $\forall v' \in \mathcal{N}(v), x_{v'}(k-1) \geq x_{v'}(k-1) - 1$, and $x_u(k-1) = x_v(k-1) + 1$, we get $x_v(k) = x_v(k-1)$. This along with $x_u(k-1) = x_v(k-1) + 1$ and $x_u(k) \in [x_u(k-1) - 1, x_u(k-1)]$ complete the proof of the claim.

Now define $f_v(k) = (1 + \frac{1}{D}) \cdot x_v(k) - distance(v, s_0) + k - 1$. We claim that

$$\forall k \in \mathbb{N}, v \in V, \textit{ either } x_v(k) = 0 \textit{ or } f_v(k) \leq D + 1.$$

Given this claim, since for any period $k > 2D + 1$ we have $f_v(k) > D + 1$, we get that for each period $k > 2D + 1$ and each node v, we have $x_v(k) = 0$, meaning that all nodes are synchronized.

Proof of the claim is by induction on k. The base case $k = 1$ is trivial. For the inductive step, suppose we have the claim for period $k-1$ and consider an arbitrary node v. If $x_v(k-1) = 0$ or if $f_v(k-1) \leq D$, then we are done because we know $x_v(k-1) \in [x_v(k-1) - 1, x_v(k-1)]$ from the first part of the proof. Suppose $x_v(k-1) \geq 1$ and $f_v(k-1) = D + 1$. We show that $x_v(k) \leq x_v(k-1) - 1$ which gives $f_v(k) \leq D$ thus completing the proof of the claim. To prove $x_v(k) \leq x_v(k-1) - 1$, it suffices to show that

$$\forall u \in \mathcal{N}^+(v), x_u(k-1) \leq x_v(k-1) \quad \& \quad \exists u^* \in \mathcal{N}^+(v),\ x_{u^*}(k-1) < x_v(k-1).$$

The first part is true because if we had $x_u(k-1) > x_v(k-1)$, then since $distance(u, s_0) \leq distance(v, s_0) + 1$, we would get $f_u(k-1) > D + 1 + 1/D$, contradicting the induction hypothesis. For the second part, choose $u^* \in \mathcal{N}(v)$ such that $distance(u^*, s_0) = distance(v, s_0) - 1$. From the induction hypothesis, we know that either $x_{u^*}(k-1) = 0$ or $f_{u^*}(k-1) \leq D+1$, which both show that $x_{u^*}(k-1) < x_v(k-1)$. $\square$