

Anchored Desynchronization

Ching-Min Lien, Shu-Hao Chang, Cheng-Shang Chang, and Duan-Shin Lee

Institute of Communications Engineering

National Tsing Hua University

Hsinchu 30013, Taiwan, R.O.C.

Email: keiichi@gibbs.ee.nthu.edu.tw; s9864549@m98.nthu.edu.tw;

cschang@ee.nthu.edu.tw;lds@cs.nthu.edu.tw

Abstract—Distributed algorithms based on pulse-coupled oscillators have been recently proposed in [5], [15] for achieving desynchronization of a system of *identical* nodes. Though these algorithms are shown to work properly by various computer simulations, they are still lack of rigorous theoretical proofs for both the convergence of the algorithms and the rates of convergence for these algorithms. On the other hand, all the nodes are not likely to be identical in many practical applications. In particular, there might be a node that needs to interact with the “outside” world and thus may not have the freedom to adjust its local clock. Motivated by all these, in this paper we consider the desynchronization problem in a system where there exists an anchored node that never adjusts the phase of its oscillator. For such a system, we propose a generic anchored desynchronization algorithm that achieves ϵ -desynchrony (defined in [5]) in $O(n^2 \ln(\frac{n}{\epsilon}))$ rounds of firings. We also prove that our algorithm converges even for the generalized processor sharing (GPS) scheme, where every node is assigned a weight and the amount of resource received by a node is proportional to its weight. Finally, when the information of the total number of nodes in the system is available to all the nodes, we propose a set of algorithms that achieve perfect desynchrony in $3n - 4$ rounds of firings. In comparison with the original algorithm in [5], we show that the rate of convergence of the original algorithm in [5] is not always better than ours and it is only better in the asymptotic regime.

I. INTRODUCTION

Desynchronization has many applications in resource scheduling in wireless networks. For example, if there are n nodes sharing a common wireless channel, a *fair* resource scheduling scheme is to perform a simple *round-robin* schedule. In such a schedule, time is divided into frames with n equal time slots in each frame, and every node is assigned exactly one time slot to transmit in each frame. Such a protocol is known as Time Division Multiple Access (TDMA) that can provide collision-free packet transmission and fully utilizes the channel in heavy load. In order for TDMA to work, it requires that every node to know the exact time to transmit in the time slot assigned to the node. This is generally done by a *centralized* coordinator (mostly a base station) that notifies every node in a wireless network.

Instead of using a centralized coordinator, Degesys, Rose, Patel, and Nagpal [5] considered a general framework for distributed algorithms to achieve desynchronization needed in TDMA. In their framework, nodes are modelled by *pulse-coupled oscillators* in [17], [11] that were designed for cardiac/firefly synchronization. They assume that (i) all the n

nodes can communicate with each other, (ii) each node is modelled by an oscillator with the same fundamental frequency, and (iii) there is no clock drift in every oscillator. Thus, the state of a node can be represented by the phase of its oscillator. Without loss of generality, it is convenient to assume that the fundamental frequency is 1 and the phase is in $[0, 1]$.

Their DESYNC-STALE algorithm in [5] works as follows. When a node reaches the end of its cycle, i.e., its phase reaches 1, it fires and resets its phase back to 0. The firing also notifies all the other nodes that it begins a new cycle. Then it waits for the next node to fire and jumps to a new phase according to a certain function. Its jumping function only uses the firing information of the node fires *before* it and the node fires *after* it. It was shown in [5] that the DESYNC-STALE algorithm achieves desynchronization, i.e., the phases of the n nodes are spaced as evenly as possible, if the new phase of each jump in a node is moved toward to an “estimated” midpoint of the phases of two neighboring nodes. However, the rate of convergence of the DESYNC-STALE algorithm is only conjectured to be $O(n^2)$ from various computer simulations.

The original objective of the desynchronization algorithm in [5] is to adjust the phase of each node so that the phases are spaced *evenly*. By so doing, each node can receive the same amount of bandwidth in a wireless network with a common channel. Pagliari, Hong and Scaglione [15] considered an important extension of the fair resource scheduling scheme to the generalized processor sharing (GPS) scheme [16], where every node is assigned a weight and the amount of bandwidth received by a node is proportional to its weight. If the weights are rational numbers, a naïve implementation of the GPS scheme is simply to have each node in the DESYNC-STALE algorithm to maintain an integer number of nodes that is proportional to its weight. As addressed in [15], such an approach is obviously inefficient as it increases the number of firings for each node and thus increases the hardware complexity of each node and the convergence time. Instead, Pagliari, Hong and Scaglione [15] proposed a genuine algorithm with two oscillators in each node and showed that their algorithm indeed converges in the *ideal* case where the up-to-date phase information is known. However, the convergence of the *stale* case was only verified by computer simulations.

Though both the DESYNC-STALE algorithm in [5] and the extension of the GPS scheme in [15] are shown to work properly by various computer simulations, they are still lack

of rigorous theoretical proofs in many aspects, including the rate of convergence of the DESYNC-STALE algorithm and the convergence of the stale GPS scheme in [15]. On the other hand, all the nodes are not likely to be identical in many practical applications. In particular, there might be a node that needs to interact with the “outside” world and thus may not have the freedom to adjust its local clock, e.g., the master node in Bluetooth, the collector node in a wireless sensor network, and the master clock in parallel analog-to-digital converters. Instead of assuming that all the nodes are identical, in this paper we consider the desynchronization problem with an *anchored* node that never adjusts its phase. Except the anchored node, all the other nodes are identical and they do not know which node the anchored node is.

For the anchored desynchronization problem, our contributions consist of three parts: (i) We are able to formally prove that our generic anchored desynchronization algorithm achieves ϵ -desynchrony (defined in [5]) in $O(n^2 \ln(\frac{n}{\epsilon}))$ rounds of firings. This partially solves the conjecture for the rate of convergence for the DESYNC-STALE algorithm in [5]. (ii) For the generalized processor sharing problem, we show that our anchored desynchronization algorithm indeed converges even in the *stale* case. This provides additional theoretical support for the convergence problem in [15]. (iii) As previously discussed in [15], one way to speed up the rate of convergence is to use the information of the total number of nodes in the system. If such information is available to each node, we propose a set of algorithms that achieve perfect desynchrony in $3n - 4$ rounds of firings. In comparison with the DESYNC-STALE algorithm in [5], we show that the rate of convergence of the DESYNC-STALE algorithm in [5] is not always better than ours and it is only better in the asymptotic regime.

The rest of the paper is organized as follows. In Section II, we propose the general framework for our anchored desynchronization algorithms. Based on the framework, we derive the system dynamics of our algorithms. In Section III, we consider the generic anchored desynchronization algorithms and obtain the formal mathematical analysis for the rate of convergence of such algorithms. We then extend the setting to the generalized processor sharing scheme in Section IV, where we prove the convergence of our anchored desynchronization algorithms. In Section V, we propose algorithms that use the information of the total number of nodes in the system and show that they achieve perfect desynchrony in $3n - 4$ rounds of firings. In Section VI, we compare the rate of convergence of the DESYNC-STALE algorithm in [5] to that of our anchored desynchronization algorithm. In Section VII, we conclude this paper by addressing some further extensions.

II. ANCHORED DESYNCHRONIZATION FRAMEWORK

As in [5], we consider the desynchronization problem in a complete graph with n nodes, i.e., all the n nodes can communicate with each other. Each node is modelled by an oscillator with frequency 1 and there is no clock drift in every oscillator. Let $\phi_i(t) \in [0, 1]$ be the phase of node i at time t , $i = 0, 1, \dots, n - 1$. Upon reaching $\phi_i(t) = 1$,

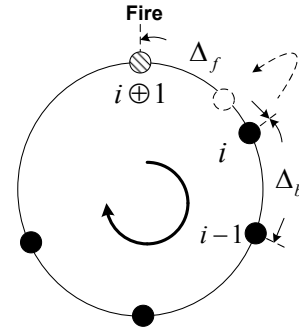


Fig. 1. Illustration of the phase adjustment of node i immediately after node $i \oplus 1$ fires (the white node indicates the new phase position of node i)

node i fires (or pulses) indicating the termination of its cycle to the other nodes. Upon firing, the node resets its phase to $\phi_i(t^+) = \lim_{\epsilon \downarrow 0} \phi_i(t + \epsilon) = 0$.

The objective of our anchored desynchronization algorithm is to adjust the phase of each node in a distributed manner so that the phases of the n nodes can be spaced as evenly as possible (or as close as possible to the targeted positions). We outline our anchored desynchronization algorithm as follows.

Algorithm 1. (General framework for anchored desynchronization)

- (i) Anchored node: node 0 is the anchored node and it never adjusts its phase when other nodes fire.
- (ii) Phase adjustment: except the anchored node, every node keeps track of three events: the firing time immediately *before* it fires, its firing time, and the firing time immediately *after* it fires. Call the node that fires immediately *after* it fires its *next* node. Let Δ_f (resp. Δ_b) be the absolute value of the difference between the firing time immediately *after* (resp. *before*) it fires and its firing time (see Figure 1). Suppose that the next node of node i fires at some time τ . Then node i adjusts its phase by setting

$$\phi_i(\tau^+) = f_i(\phi_i(\tau), \Delta_b, \Delta_f), \quad (1)$$

where $f_i(\cdot, \cdot, \cdot)$ is a deterministic function available to node i .

Without loss of generality, we assume that the phases of the n nodes are initially ordered as follows:

$$1 = \phi_0(0) > \phi_1(0) > \phi_2(0) > \dots > \phi_{n-1}(0) > 0. \quad (2)$$

Thus, node 0 is the first one to fire at time 0 and the phase of node 0 is reset to 0, i.e., $\phi_0(0^+) = 0$. To ease our presentation, the initial firing of node 0 at time 0 is counted as the 0^{th} firing of node 0. Also, we define $i \oplus 1$ as $(i + 1) \bmod n$. As there is no clock drift, node 0 will fire for the m^{th} time at time m , $m = 1, 2, \dots$. In order to make sure that every node fires according to the desired order, i.e., node 0, node 1, node 2, \dots , node $n - 1$, and then node 0, we need the following non-overtaking condition.

(Non-overtaking condition) Suppose that node $i \oplus 1$ fires at some time τ for some $i = 1, 2, \dots, n - 1$. Then, node i

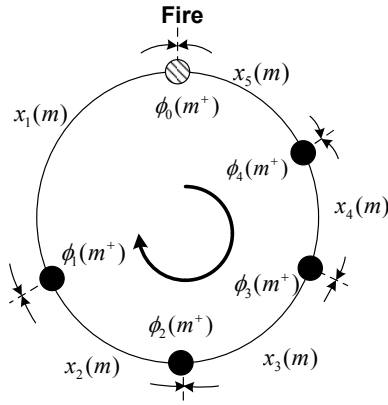


Fig. 2. The state of the system with $n = 5$ at time m^+ (the anchored node is marked with stripes)

will adjust its phase such that its phase satisfies the following inequality:

$$\phi_{i-1}(\tau) > \phi_i(\tau^+) > \phi_{i\oplus 1}(\tau^+) = 0. \quad (3)$$

Note that there is no need to check the non-overtaking condition for node 0 as node 0 is the anchored node and it never adjusts its phase when other nodes fire. The condition in (3) ensures that the order of the phases is preserved after the adjustment of the phase of every node. By so doing, every node fires exactly once in every unit of time. Also, except the anchored node, every node adjusts its phase exactly once in every unit of time when its next node fires.

Suppose that the non-overtaking condition is satisfied (which we will verify later for our algorithm). Then we can take a snap shot of the phases immediately after node 0 fires at time m . Let

$$\mathbf{x}(m) = (x_1(m), x_2(m), \dots, x_n(m))^T,$$

where

$$x_i(m) = \begin{cases} 1 - \phi_1(m^+) & \text{for } i = 1 \\ \phi_{i-1}(m^+) - \phi_i(m^+) & \text{for } i = 2, 3, \dots, n-1 \\ \phi_{n-1}(m^+) & \text{for } i = n \end{cases}. \quad (4)$$

As shown in Figure 2, $x_i(m)$, $i = 1, 2, \dots, n-1$, is the phase difference between node $i-1$ and node i immediately after node 0 fires at time m , and $x_n(m)$ is the phase difference between node $n-1$ and node 0 immediately after node 0 fires at time m . Clearly,

$$\sum_{i=1}^n x_i(m) = 1. \quad (5)$$

Moreover, if the non-overtaking condition is satisfied, then we have for all m

$$x_i(m) > 0, \quad i = 1, 2, \dots, n. \quad (6)$$

In this paper, we will use $\mathbf{x}(m)$ as the state vector of our algorithm. In particular, we have from the initial condition in

(2) that

$$\mathbf{x}(0) = (1 - \phi_1(0), \phi_1(0) - \phi_2(0), \dots, \phi_{n-2}(0) - \phi_{n-1}(0), \phi_{n-1}(0))^T$$

and the state vector indeed contains nonzero elements.

Let $\tau_i(m)$ be the time that node i fires for the m^{th} time. Clearly, we have

$$\tau_0(m) = m, \quad (7)$$

and for $i = 1, 2, \dots, n-1$,

$$\tau_i(m+1) = m+1 - \phi_i(m^+). \quad (8)$$

Thus, for $i = 1, 2, \dots, n-2$,

$$\begin{aligned} & \tau_{i+1}(m+1) - \tau_i(m+1) \\ &= m+1 - \phi_{i+1}(m^+) - (m+1 - \phi_i(m^+)) \\ &= x_i(m), \end{aligned} \quad (9)$$

and

$$\begin{aligned} & \tau_0(m+1) - \tau_{n-1}(m+1) \\ &= m+1 - (m+1 - \phi_{n-1}(m^+)) = x_n(m). \end{aligned} \quad (10)$$

In conjunction with (9) and (10), we have for $i = 1, 2, \dots, n-1$,

$$\tau_{i\oplus 1}(m+1) - \tau_i(m+1) = x_{i+1}(m). \quad (11)$$

In the following lemma, we first derive the governing dynamics of the system.

Lemma 1: For some $i = 1, 2, \dots, n-1$, suppose that the non-overtaking condition in (3) is satisfied up to $\tau_i(m+1)^+$, i.e., the time immediately after node i fires for the $(m+1)^{\text{th}}$ time. Under Algorithm 1, the following holds:

- (i) The phase of node i when (or immediately before) node $i \oplus 1$ fires for the $(m+1)^{\text{th}}$ time is

$$\phi_i(\tau_{i\oplus 1}(m+1)) = x_{i+1}(m). \quad (12)$$

- (ii) The phase of node i immediately after node $i \oplus 1$ fires for the $(m+1)^{\text{th}}$ time is

$$\begin{aligned} & \phi_i(\tau_{i\oplus 1}(m+1)^+) \\ &= f_i(x_{i+1}(m), x_i(m), x_{i+1}(m)). \end{aligned} \quad (13)$$

- (iii) For $i = 1$, the phase of node $i-1$ immediately after node $i \oplus 1$ fires for the $(m+1)^{\text{th}}$ time is

$$\begin{aligned} & \phi_{i-1}(\tau_{i\oplus 1}(m+1)) = \phi_0(\tau_2(m+1)) \\ &= x_1(m) + x_2(m). \end{aligned} \quad (14)$$

- (iv) For $i > 1$, the phase of node $i-1$ immediately after node $i \oplus 1$ fires for the $(m+1)^{\text{th}}$ time is

$$\begin{aligned} & \phi_{i-1}(\tau_{i\oplus 1}(m+1)) \\ &= f_{i-1}(x_i(m), x_{i-1}(m), x_i(m)) + x_{i+1}(m). \end{aligned} \quad (15)$$

Proof. (i) Since node i fires for the $(m+1)^{\text{th}}$ time at time $\tau_i(m+1)$ and its phase is reset back to 0, we have $\phi_i(\tau_i(m+1)$

$1)^+ = 0$. Its phase then increases linearly before $\tau_{i \oplus 1}(m+1)$. In view of (11), it follows that

$$\begin{aligned} & \phi_i(\tau_{i \oplus 1}(m+1)) \\ &= \tau_{i \oplus 1}(m+1) - \tau_i(m+1) = x_{i+1}(m). \end{aligned}$$

(ii) Since the non-overtaking condition in (3) is satisfied up to $\tau_i(m+1)^+$, we have from (11) that

$$\Delta_f = \tau_{i \oplus 1}(m+1) - \tau_i(m+1) = x_{i+1}(m). \quad (16)$$

Similarly, for $i > 1$,

$$\Delta_b = \tau_i(m+1) - \tau_{i-1}(m+1) = x_i(m). \quad (17)$$

For $i = 1$, we have from (8), (7) and (4) that

$$\begin{aligned} \Delta_b &= \tau_1(m+1) - \tau_0(m) \\ &= m+1 - \phi_1(m^+) - m = x_1(m). \end{aligned} \quad (18)$$

Thus, for all $i = 1, 2, \dots, n-1$,

$$\Delta_b = x_i(m). \quad (19)$$

Using (12), (16) and (19) in the phase adjustment rule in (1) yields

$$\begin{aligned} \phi_i(\tau_{i \oplus 1}(m+1)^+) &= f_i(\phi_i(\tau_{i \oplus 1}(m+1)), \Delta_b, \Delta_f) \\ &= f_i(x_{i+1}(m), x_i(m), x_{i+1}(m)). \end{aligned}$$

(iii) For the case that $i = 1$,

$$\phi_{i-1}(\tau_{i \oplus 1}(m+1)) = \phi_0(\tau_2(m+1)).$$

Since node 0 never adjusts its phase, we have

$$\begin{aligned} \phi_0(\tau_2(m+1)) &= \tau_2(m+1) - m \\ &= 1 - \phi_2(m) = x_1(m) + x_2(m), \end{aligned}$$

where we use (8) and (4) in the last two identities.

(iv) For the case that $i > 1$, we note that node i fired at time $\tau_i(m+1)$ and node $i-1$ adjusted its phase immediately after node i fired. Thus, we have from (13) and (11) that

$$\begin{aligned} & \phi_{i-1}(\tau_{i \oplus 1}(m+1)) \\ &= \phi_{i-1}(\tau_i(m+1)^+) + \tau_{i \oplus 1}(m+1) - \tau_i(m+1) \\ &= f_{i-1}(x_i(m), x_{i-1}(m), x_i(m)) + x_{i+1}(m). \end{aligned} \quad (20)$$

■

As a direct consequence of Lemma 1 (ii), (iii) and (iv), the non-overtaking condition in (3) can be easily checked by the conditions stated in the following corollary.

Corollary 2: For some $i = 1, 2, \dots, n-1$, suppose that the non-overtaking condition in (3) is satisfied up to $\tau_i(m+1)^+$, i.e., the time immediately after node i fires for the $(m+1)^{th}$ time. Then the non-overtaking condition in (3) is satisfied up to $\tau_{i \oplus 1}(m+1)^+$, i.e., the time immediately after node $i \oplus 1$ fires for the $(m+1)^{th}$ time if for $i = 1$

$$x_1(m) + x_2(m) > f_1(x_2(m), x_1(m), x_2(m)) > 0, \quad (21)$$

and for $i > 1$

$$\begin{aligned} & f_{i-1}(x_i(m), x_{i-1}(m), x_i(m)) + x_{i+1}(m) \\ & > f_i(x_{i+1}(m), x_i(m), x_{i+1}(m)) > 0. \end{aligned} \quad (22)$$

III. GENERIC ANCHORED DESYNCHRONIZATION

In this section, we propose our generic anchored desynchronization algorithm by choosing

$$f_i(\phi_i(\tau), \Delta_b, \Delta_f) = \gamma \phi_i(\tau) + (1-\gamma) \frac{\Delta_b + \Delta_f}{2}, \quad (23)$$

for $0 \leq \gamma < 1$ and $i = 1, 2, \dots, n-1$. Note that the phase adjustment rule in (23) is exactly the same as that in the DESYNC-STALE algorithm in [5]. For $\gamma = 0$, the rule simply adjusts the phase to the stale midpoint of two neighboring nodes. As such, the parameter $1-\gamma$ is called the jump size parameter in [5]. Thus, the only difference between the DESYNC-STALE algorithm in [5] and ours is the anchored node. Though it was shown in [5] that the DESYNC-STALE algorithm indeed achieves desynchronization, it is still not clear what the rate of convergence is. With the insertion of the anchored node, we are able to obtain a formal theoretical result for the rate of convergence.

In the following lemma, we first derive the governing dynamics of the system under (23).

Lemma 3: Suppose that the functions $f_i, i = 1, 2, \dots, n-1$, are chosen in (23) with $0 \leq \gamma < 1$. Under Algorithm 1 and initial condition in (2), the following holds:

- (i) The phase of node $i, i = 1, 2, \dots, n-1$, immediately after node $i \oplus 1$ fires for the $(m+1)^{th}$ time is

$$\begin{aligned} & \phi_i(\tau_{i \oplus 1}(m+1)^+) \\ &= \frac{(1+\gamma)}{2} x_{i+1}(m) + \frac{(1-\gamma)}{2} x_i(m). \end{aligned} \quad (24)$$

- (ii) For $i > 1$, the phase of node $i-1$ immediately after node $i \oplus 1$ fires for the $(m+1)^{th}$ time is

$$\begin{aligned} & \phi_{i-1}(\tau_{i \oplus 1}(m+1)) = x_{i+1}(m) \\ & + \frac{(1+\gamma)}{2} x_i(m) + \frac{(1-\gamma)}{2} x_{i-1}(m). \end{aligned} \quad (25)$$

- (iii) The non-overtaking condition is satisfied and thus $x_i(m) > 0$ for all i and m .

Proof. For (i) and (ii) of this lemma, we simply replace f_i with (23) in (13) and (15) of Lemma 1. To check the non-overtaking condition in (iii), we simply replace f_i with (23) in (21) and (22) of Corollary 2. It is easy to see that for $0 \leq \gamma < 1$ that the inequalities in (21) and (22) are satisfied with the induction hypothesis $x_i(m) > 0$ for all i . From the initial condition in (2), we have $x_i(0) > 0$ for all i and that verifies the induction hypothesis for $m = 0$. ■

Since the non-overtaking condition is satisfied, every node fires exactly once in every unit of time. Thus, node $i, i = 1, 2, \dots, n-1$, adjusts its phase exactly once in every unit of time immediately after node $i \oplus 1$ fires, and it will not adjust its phase again before the next firing of the anchor node.

From (14) and (24), it then follows that

$$\begin{aligned}
x_1(m+1) &= 1 - \phi_1((m+1)^+) \\
&= \phi_0(\tau_2(m+1)) - \phi_1(\tau_2(m+1)^+) \\
&= x_1(m) + x_2(m) - \frac{(1+\gamma)}{2}x_2(m) - \frac{(1-\gamma)}{2}x_1(m) \\
&= \frac{1+\gamma}{2}x_1(m) + \frac{1-\gamma}{2}x_2(m). \tag{26}
\end{aligned}$$

Also, we have from (24) and (25) that for $i = 2, \dots, n-1$,

$$\begin{aligned}
x_i(m+1) &= \phi_{i-1}((m+1)^+) - \phi_i((m+1)^+) \\
&= \phi_{i-1}(\tau_{i\oplus 1}(m+1)) - \phi_i(\tau_{i\oplus 1}(m+1)^+) \\
&= \frac{1-\gamma}{2}x_{i-1}(m) + \gamma x_i(m) + \frac{1-\gamma}{2}x_{i+1}(m). \tag{27}
\end{aligned}$$

Finally, we have from (26) and (27) that

$$\begin{aligned}
&\sum_{i=1}^{n-1} x_i(m+1) \\
&= \sum_{i=1}^{n-2} x_i(m) + \frac{1+\gamma}{2}x_{n-1}(m) + \frac{1-\gamma}{2}x_n(m). \tag{28}
\end{aligned}$$

Thus, it follows from (5) that

$$\begin{aligned}
x_n(m+1) &= 1 - \sum_{i=1}^{n-1} x_i(m+1) \\
&= \frac{1-\gamma}{2}x_{n-1}(m) + \frac{1+\gamma}{2}x_n(m). \tag{29}
\end{aligned}$$

Then we can write (26), (27) and (29) in the matrix form

$$\mathbf{x}(m+1) = \mathbf{W}\mathbf{x}(m), \tag{30}$$

where $\mathbf{x}(m) = (x_1(m), x_2(m), \dots, x_n(m))^T$ is the state vector and $\mathbf{W} = (W_{ij})$ is the $n \times n$ tridiagonal matrix with

$$W_{ij} = \begin{cases} \frac{1+\gamma}{2} & \text{for } i = j = 1 \text{ or } i = j = n, \\ \gamma & \text{for } i = j = 2, \dots, n-1, \\ \frac{1-\gamma}{2} & \text{for } i = j-1 = 1, 2, \dots, n-1, \\ \frac{1-\gamma}{2} & \text{for } i = j+1 = 2, \dots, n, \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

In particular, if $n = 5$, then

$$\mathbf{W} = \begin{pmatrix} \frac{1+\gamma}{2} & \frac{1-\gamma}{2} & 0 & 0 & 0 \\ \frac{1-\gamma}{2} & \gamma & \frac{1-\gamma}{2} & 0 & 0 \\ 0 & \frac{1-\gamma}{2} & \gamma & \frac{1-\gamma}{2} & 0 \\ 0 & 0 & \frac{1-\gamma}{2} & \gamma & \frac{1-\gamma}{2} \\ 0 & 0 & 0 & \frac{1-\gamma}{2} & \frac{1+\gamma}{2} \end{pmatrix}.$$

For $0 \leq \gamma < 1$, we have $W_{ij} \geq 0$. Moreover, both the row sums and the column sums of \mathbf{W} are equal to 1, i.e.,

$$\sum_{i=1}^n W_{ij} = 1, \quad j = 1, 2, \dots, n, \tag{32}$$

and

$$\sum_{j=1}^n W_{ij} = 1, \quad i = 1, 2, \dots, n. \tag{33}$$

Such a matrix is known as a doubly stochastic matrix (see e.g., the book by Marshall and Olkin [10]). In view of the recursion for the state vectors in (30), there is a partial ordering, known as the majorization ordering ([10], p 20, Theorem 2.A.4), among the sequence of the state vectors $\mathbf{x}(m)$. As a direct consequence of the majorization ordering, we know that $\sum_{i=1}^n g(x_i(m))$ is decreasing in m for any convex function g ([10], p 108, Proposition 4.B.1). In particular, $\sum_{i=1}^n |x_i(m) - \frac{1}{n}|$ is decreasing m . To further understand the rate of convergence, we need to identify the eigenvalues of the matrix \mathbf{W} and this is done in the following proposition.

Proposition 4: The n eigenvalues of the $n \times n$ matrix \mathbf{W} are $\gamma + (1-\gamma)\cos(\frac{i\pi}{n})$, $i = 0, 1, \dots, n-1$. Thus, the largest eigenvalue of \mathbf{W} is 1. Moreover, for $0 \leq \gamma < 1$, the absolute values of the other eigenvalues of \mathbf{W} are bounded above by $\gamma + (1-\gamma)\cos(\frac{\pi}{n})$. Thus, the second largest eigenvalue modulus (SLEM) of \mathbf{W} , defined as the maximum of the absolute values of the other eigenvalue, is $\gamma + (1-\gamma)\cos(\frac{\pi}{n})$.

Proof. Write

$$\mathbf{W} = \mathbf{I} - \frac{1-\gamma}{2}\mathbf{L}, \tag{34}$$

where \mathbf{I} is the $n \times n$ identity matrix and $\mathbf{L} = (L_{ij})$ is the $n \times n$ tridiagonal matrix with

$$L_{ij} = \begin{cases} 1 & \text{for } i = j = 1 \text{ or } i = j = n, \\ 2 & \text{for } i = j = 2, \dots, n-1, \\ -1 & \text{for } i = j-1 = 1, 2, \dots, n-1, \\ -1 & \text{for } i = j+1 = 2, \dots, n, \\ 0 & \text{otherwise.} \end{cases} \tag{35}$$

In particular, if $n = 5$, then

$$\mathbf{L} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}.$$

For such a tridiagonal matrix \mathbf{L} , it is well-known (see e.g., [20]) that its eigenvalues are $2 - 2\cos(\frac{i\pi}{n})$, $i = 1, 2, \dots, n$. In view of (34), if λ is an eigenvalue of \mathbf{L} with the corresponding eigenvector \mathbf{u} , then $1 - \frac{1-\gamma}{2}\lambda$ is also an eigenvalue of \mathbf{W} with the corresponding eigenvector \mathbf{u} . Thus, the n eigenvalues of the $n \times n$ matrix \mathbf{W} are $\gamma + (1-\gamma)\cos(\frac{i\pi}{n})$, $i = 0, 1, \dots, n-1$.

When $i = 0$, we know that \mathbf{W} has the eigenvalue 1. For $i = 1, 2, \dots, n-1$ and $0 \leq \gamma < 1$, we also have

$$\begin{aligned}
|\gamma + (1-\gamma)\cos(\frac{i\pi}{n})| &\leq \gamma + (1-\gamma)|\cos(\frac{i\pi}{n})| \\
&\leq \gamma + (1-\gamma)\cos(\frac{\pi}{n}).
\end{aligned}$$

■

Analogous to the definition of desynchronization accuracy in [6], we define the desynchronization accuracy as the sum of the absolute deviations from perfect desynchrony. We say

that the system is ϵ -desynchronized after m rounds of firing if

$$\sum_{i=1}^n |x_i(m) - \frac{1}{n}| \leq \epsilon. \quad (36)$$

Theorem 5: For $0 \leq \gamma < 1$, a system of n nodes whose dynamics are governed by the generic anchored desynchronization algorithm, i.e., Algorithm 1 with f_i in (23), achieves ϵ -desynchrony in $O(n^2 \ln(\frac{n}{\epsilon}) / (1 - \gamma))$ rounds of firings.

Proof. We note that we can associate the system to a specific birth-death process by viewing \mathbf{W} as the transition probability matrix of the birth-death process (see e.g., [13]). It is well known that such a stochastic process is a reversible discrete-time Markov chain. As \mathbf{W} is a doubly stochastic matrix, the steady state probability distribution of the Markov chain is the *uniform* distribution. Thus, we have

$$\lim_{m \rightarrow \infty} \sum_{i=1}^n |x_i(m) - \frac{1}{n}| = 0.$$

Diconis and Stroock ([4], Proposition 3) derived a total variation bound for the mixing time of a reversible Markov chain. When the bound is applied for the uniform distribution, it can be written as follows (see [2], p. 669):

$$\sum_{i=1}^n |x_i(m) - \frac{1}{n}| \leq \sqrt{n} \mu^m, \quad (37)$$

where μ is the second largest eigenvalue modulus (SLEM) of the transition probability matrix \mathbf{W} . Since we have shown in Proposition 4 that the SLEM of \mathbf{W} is $\gamma + (1 - \gamma) \cos(\frac{\pi}{n})$. It then follows from (37) that

$$\sum_{i=1}^n |x_i(m) - \frac{1}{n}| \leq \sqrt{n} \left(\gamma + (1 - \gamma) \cos\left(\frac{\pi}{n}\right) \right)^m. \quad (38)$$

For large n , we have

$$\cos\left(\frac{\pi}{n}\right) \approx 1 - \frac{1}{2} \left(\frac{\pi}{n}\right)^2,$$

and

$$\ln\left(1 - (1 - \gamma) \frac{\pi^2}{2n^2}\right) \approx -(1 - \gamma) \frac{\pi^2}{2n^2}.$$

Using these in (38) yields

$$\begin{aligned} & \ln\left(\sqrt{n} \left(\gamma + (1 - \gamma) \cos\left(\frac{\pi}{n}\right)\right)^m\right) \\ & \approx \frac{1}{2} \ln n - m(1 - \gamma) \frac{\pi^2}{2n^2}. \end{aligned} \quad (39)$$

In view of (38) and (39), we conclude that the system achieves ϵ -desynchrony in $O(n^2 \ln(\frac{n}{\epsilon}) / (1 - \gamma))$ rounds of firings.

We note that an alternative proof of this theorem is to use the upper bound for the mixing time of a reversible Markov chain by the relaxation time in Theorem 12.3 of the book [8].

■

IV. GENERALIZED PROCESSOR SHARING

The original objective of the desynchronization algorithm is to adjust the phase of each node so that the phases are spaced *evenly*. By so doing, each node can receive the same amount of bandwidth in a wireless network with a common channel. An important extension of the fair resource scheduling scheme is the generalized processor sharing (GPS) scheme addressed in [16], [15]. In the GPS scheme, every node is assigned a weight and the amount of bandwidth assigned to a node should be proportional to its weight. For this purpose, we will extend the generic anchored desynchronization algorithm so that the phase differences are proportional to the weights. Specifically, let $\alpha_i > 0$ be the weight assigned to node i , $i = 0, 1, \dots, n - 1$. The objective of this section is to propose an anchored desynchronization algorithm so that for all i

$$\lim_{m \rightarrow \infty} |x_i(m) - \frac{\alpha_{i-1}}{\sum_{j=0}^{n-1} \alpha_j}| = 0. \quad (40)$$

For this objective, we choose for $i = 1, 2, \dots, n - 1$,

$$f_i(\phi_i(\tau), \Delta_b, \Delta_f) = \gamma \phi_i(\tau) + (1 - \gamma) \beta_i (\Delta_b + \Delta_f), \quad (41)$$

where

$$0 < \beta_i = \alpha_i / (\alpha_{i-1} + \alpha_i) < 1. \quad (42)$$

Note that for this algorithm to work, node i needs to have the information of α_i and α_{i-1} . If every node knows its own weight at the beginning, every node still needs to pass its own weight to its next node. This information might be embedded when a node fires. Another trick, as proposed in [15], is for every node to have two oscillators and a universal weight δ that is known to every node. In such a setting, every node behaves as if it has two virtual nodes and the system can thus be viewed as a system of $2n$ nodes with weights $\alpha_0, \delta, \alpha_1, \delta, \dots, \alpha_{n-1}, \delta$. As δ is known to each node, each node now has the information of the weight of the node fired before it. For such a system, the amount of bandwidth received by node i is then proportional to $\alpha_i + \delta$. As discussed in [15], the constant δ needs to be relatively small to ensure “proportional fairness.”

In the following lemma, we first derive the governing dynamics of the system with generalized processor sharing.

Lemma 6: Suppose that the functions f_i , $i = 1, 2, \dots, n - 1$, are chosen in (41) with $1/2 \leq \gamma < 1$. Under Algorithm 1, the following holds:

- (i) The phase of node i , $i = 1, 2, \dots, n - 1$, immediately after node $i \oplus 1$ fires for the $(m + 1)^{th}$ time is

$$\begin{aligned} \phi_i(\tau_{i \oplus 1}(m + 1)^+) &= (\gamma + (1 - \gamma) \beta_i) x_{i+1}(m) \\ &+ (1 - \gamma) \beta_i x_i(m). \end{aligned} \quad (43)$$

- (ii) The phase of node $i - 1$, $i = 2, \dots, n - 1$, immediately after node $i \oplus 1$ fires for the $(m + 1)^{th}$ time is

$$\begin{aligned} \phi_{i-1}(\tau_{i \oplus 1}(m + 1)) &= (\gamma + (1 - \gamma) \beta_{i-1}) x_i(m) \\ &+ (1 - \gamma) \beta_{i-1} x_{i-1}(m) + x_{i+1}(m). \end{aligned} \quad (44)$$

(iii) The non-overtaking condition is satisfied and thus $x_i(m) > 0$ for all i and m .

Proof. The proof for (i) and (ii) of this lemma is the same as that in Lemma 3. Analogous to the proof for the non-overtaking condition in Lemma 3, we use the induction hypothesis that $x_i(m) > 0$ for all i . From the induction hypothesis and (43), it is clear that $\phi_i(\tau_{i\oplus 1}(m+1)^+) > 0$ for $1/2 \leq \gamma < 1$. On the other hand, we note from (44) and (43) that

$$\begin{aligned} & \phi_{i-1}(\tau_{i\oplus 1}(m+1)) - \phi_i(\tau_{i\oplus 1}(m+1)^+) \\ &= (1 - \beta_i)(1 - \gamma)x_{i+1}(m) + (1 - \gamma)\beta_{i-1}x_{i-1}(m) \\ & \quad + (\gamma + (1 - \gamma)(\beta_{i-1} - \beta_i))x_i(m) \end{aligned}$$

Since $0 < \beta_i < 1$ for all i , we have for $1/2 \leq \gamma < 1$

$$(\gamma + (1 - \gamma)(\beta_{i-1} - \beta_i)) > 2\gamma - 1 \geq 0.$$

Thus, $\phi_{i-1}(\tau_{i\oplus 1}(m+1)) > \phi_i(\tau_{i\oplus 1}(m+1)^+)$ and the non-overtaking condition is satisfied. ■

Analogous to the derivation for the recursive equation for the state vector in (30), one can show that

$$\mathbf{x}(m+1) = \tilde{\mathbf{W}}\mathbf{x}(m), \quad (45)$$

where $\tilde{\mathbf{W}} = (\tilde{W}_{ij})$ is the $n \times n$ tridiagonal matrix with

$$\tilde{W}_{ij} = \begin{cases} 1 - (1 - \gamma)\beta_1, & \text{for } j = i = 1, \\ (1 - \gamma)(1 - \beta_i), & \text{for } j = i + 1, \\ \gamma + (1 - \gamma)(\beta_{i-1} - \beta_i), & \text{for } j = i \neq 1 \text{ or } n, \\ (1 - \gamma)\beta_{i-1}, & \text{for } j = i - 1, \\ 1 - (1 - \gamma)(1 - \beta_{n-1}) & \text{for } j = i = n, \\ 0, & \text{otherwise.} \end{cases} \quad (46)$$

The governing dynamics in Lemma 6 leads to the following theorem.

Theorem 7: Suppose that the functions $f_i, i = 1, 2, \dots, n-1$, are chosen in (41) with $1/2 \leq \gamma < 1$. Under Algorithm 1, the state vector \mathbf{x} of the system of n nodes converge to $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_{n-1})$, i.e.,

$$\lim_{m \rightarrow \infty} |x_i(m) - \frac{\alpha_{i-1}}{\sum_{j=0}^{n-1} \alpha_j}| = 0, \quad i = 1, 2, \dots, n.$$

Proof. It is straightforward to check that the sum of each column in matrix $\tilde{\mathbf{W}}$ is one, namely, $\sum_{i=1}^n \tilde{W}_{ij} = 1$ for all $j = 1, 2, \dots, n$. Analogous to the proof of Theorem 5, we can associate the system to a discrete-time birth-death process with the transition probability matrix $\tilde{\mathbf{W}}^T$ and the steady state of the system, denoted by $\mathbf{x} = (x_1, x_2, \dots, x_n)$, is simply the steady state probability vector of the birth-death process that can be solved by the following detailed balance equations (see e.g., [13]):

$$x_i(1 - \gamma)\beta_i = x_{i+1}(1 - \gamma)(1 - \beta_i), \quad i = 1, 2, \dots, n-1, \quad (47)$$

and

$$\sum_{i=1}^n x_i = 1. \quad (48)$$

Since $\beta_i = \alpha_i / (\alpha_{i-1} + \alpha_i)$, we have from (47) that

$$\frac{x_{i+1}}{\alpha_i} = \frac{x_i}{\alpha_{i-1}}, \quad i = 1, 2, \dots, n-1. \quad (49)$$

In conjunction with (48), we then have $x_i = \frac{\alpha_{i-1}}{\sum_{j=0}^{n-1} \alpha_j}$, $i = 1, 2, \dots, n$. ■

V. ANCHORED DESYNCHRONIZATION WITH THE INFORMATION OF THE TOTAL NUMBER OF NODES

In Theorem 5, we show that a system of n nodes whose dynamics are governed by the generic anchored desynchronization algorithm achieves ϵ -desynchrony in $O(n^2 \ln(\frac{n}{\epsilon}) / (1 - \gamma))$ rounds of firings. One question is whether we can beat such a rate of convergence by offering additional information. One piece of information that might be relatively easy to collect is the total number of nodes in the system. This could be done by adding a counter in each node that counts the number of firings between two successive firings of a node. In this section, we assume that the information of the total number of nodes is available to all the nodes. With this additional information, we will show how the rate of convergence could be improved.

Anchored desynchronization with the information of the total number of nodes would be rather easy to achieve if all the nodes knew which one the anchored node was. Specifically, suppose that there are n nodes in the system and every node knows that node 0 is the anchored node. Then node i knows that its relative phase difference to node 0 should be i/n . After the first round of firing, every node knows its relative order in the system and it can then adjust its phase to the targeted position.

Motivated by this, our idea to speed up the rate of convergence is for our anchored desynchronization to “learn” which node the anchored node is (without adding additional hardware complexity). As the anchored node never adjusts its phase, one simple way to recognize the anchored node is to observe whether the firing times of a node are spaced exactly one unit apart. As we have already stored the firing time of the next node in the previous round, we can compare that with the firing time in the current round. If the difference is exactly one unit of time, then the node treats its next node as the anchored node. Clearly, after two rounds of firings, node $n-1$ will be the first node to identify the anchored node and it can then adjust its relative phase to node 0 to $1/n$. After that, node $n-1$ has been set to the targeted position and it will not adjust its phase further. Such a node will be called a *locked* node (as its relative phase to the anchored node is locked). A locked node behaves just like another anchored node. After another two rounds of firings, node $n-2$ will treat node $n-1$ as the “anchored” node and adjust its relative phase to node 1 to $1/n$. Then node $n-2$ becomes a locked node. Repeating the process, one can see that node i adjusts its phase to its targeted position after treating node $i \oplus 1$ as the anchored node and node i will not adjust its phase from that point on. Thus, the convergence of the above algorithm works like a ripple propagating from node $n-1$ to node 1.

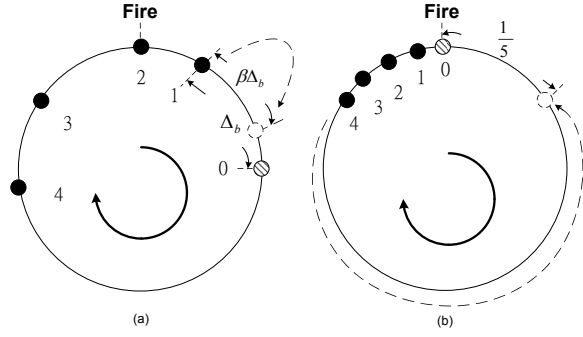


Fig. 3. (a) Illustration of the non-overtaking condition for the phase adjustment of node i with the choice of functions in (52). (b) Nodes are clustered near the anchored node.

There is one catch for the above argument. In order for the algorithm to work properly, we need to make sure the non-overtaking condition in (3) is satisfied so that the relative order of firings remains unchanged. Thus, even when a node treats its next node as the anchored node, it may not be able to adjust its phase difference to $1/n$ as the non-overtaking condition might be violated. For this, we have to put a restriction on the jump size and such a restriction slows down the rate of convergence. As such, our second idea to speed up the rate of convergence is to reduce the restriction of the jump size by moving nodes away from their next nodes. This is outlined in the following algorithm.

Algorithm 2. (Anchored desynchronization by learning locked nodes) The algorithm runs Algorithm 1 with the following additional twist. Suppose that the next node of node i fires at some time τ . If the difference of the previous firing time of node $i \oplus 1$ and the current firing time is exactly 1, then node i treats node $i \oplus 1$ as a *locked* node and adjusts its phase by setting

$$\phi_i(\tau^+) = \min \left[\frac{1}{n}, f_i(\phi_i(\tau), \Delta_b, \Delta_f) \right]. \quad (50)$$

Otherwise, node i adjusts its phase as in Algorithm 1, i.e.,

$$\phi_i(\tau^+) = f_i(\phi_i(\tau), \Delta_b, \Delta_f). \quad (51)$$

As discussed before, the order for the $n-1$ nodes to become locked is node $n-1$, node $n-2$, ..., and then node 1. Even when node i has learned that node $i \oplus 1$ is a locked node, it still follows the same phase adjustment rule as in Algorithm 1 until the relative phase difference between node i and node $i \oplus 1$ is not smaller than $1/n$. Then it is locked and it will not adjust its phase further. Thus, the non-overtaking condition is satisfied in Algorithm 2 if the non-overtaking condition in Corollary 2 is satisfied in Algorithm 1.

Now consider the choice of the functions

$$f_i(\phi_i(\tau), \Delta_b, \Delta_f) = \phi_i(\tau) + \beta\Delta_b, \quad i = 1, 2, \dots, n-1, \quad (52)$$

where β is a constant in $(0, 1)$. It is rather straightforward to see that the two inequalities in (21) and (22) of Corollary 2 are satisfied and thus Algorithm 2 with such a choice of functions is non-overtaking.

To see the intuition behind such a choice of functions, we note that the phase of a node in each update is always increased by $\beta\Delta_b$ before it becomes a locked node. If we choose β near 1, we note that node 1 will be moved to a position that is very close to the anchored node after the first round of firings. Then node 2 will also be moved to a position that is very close to the anchored node after the second round of firings. After $n-1$ rounds of firing, nodes $1, \dots, n-1$, will be clustered near the anchored node as shown in Figure 3(b). As they are all clustered near the anchored node, they will not be limited by the jump size once they identify their next nodes as locked nodes. Thus, after n rounds of firing, node $n-1$ will be in its targeted position, i.e., its relative phase difference to node 0 is $1/n$. Similarly, node $n-i$ will be in its targeted position after $n+2(i-1)$ rounds of firings (as it takes two rounds to identify a locked node). What happens here is that each node is moved toward the anchored node first and then back to its targeted position like a boomerang. With the intuition in mind, we prove in the following theorem that Algorithm 2 indeed converges in $O(n)$ rounds of firings if β is close to 1.

Theorem 8: Suppose that the functions $f_i, i = 1, 2, \dots, n-1$, are chosen in (52) with $\beta \geq 1 - \frac{2}{n^2}$. Then Algorithm 2 achieves perfect desynchrony in $3n-4$ rounds of firings.

For the proof of Theorem 8, we need the following result in Lemma 9.

Lemma 9: Suppose that node i has not been locked after node 0 fires at time m_0 . Then for all $j = 1, \dots, i$ and $j \leq m \leq m_0$,

$$x_j(m) \leq j(1-\beta). \quad (53)$$

Proof. Since node i has not been locked after node 0 fires at time m_0 , the phase of node i is changed during each update and thus node $i-1$ will not identify node i as a locked node. As such, we know that nodes $j, j = 1, 2, \dots, i$ have not been locked after node 0 fires at time m_0 and they follow the phase adjustment rule in (51).

We first show that $x_1(m) \leq (1-\beta)$ for all $1 \leq m \leq m_0$. At time $\tau_2(m+1)$, node 2 fires for the $(m+1)^{th}$ time. From (13) and (52), we know that

$$\phi_1(\tau_2(m+1)^+) = x_2(m) + \beta x_1(m).$$

Since $\phi_0(\tau_2(m+1)) = x_2(m) + x_1(m)$ in (14) and the phase of node 1 will not be adjusted before time $m+1$, it follows that

$$\begin{aligned} x_1(m+1) &= 1 - \phi_1((m+1)^+) \\ &= \phi_0(\tau_2(m+1)) - \phi_1(\tau_2(m+1)^+) \\ &= (1-\beta)x_1(m) \leq 1-\beta. \end{aligned}$$

Now assume that for all $j = 1, 2, \dots, j_0-1$ ($2 \leq j_0 \leq i$) and $j \leq m \leq m_0$,

$$x_j(m) \leq j(1-\beta) \quad (54)$$

as the induction hypothesis. At time $\tau_{j_0 \oplus 1}(m+1)$, node $j_0 \oplus 1$ fires for the $(m+1)^{th}$ time. According to (52) and (13), node j_0 adjusts its phase to

$$\phi_{j_0}(\tau_{j_0 \oplus 1}(m+1)^+) = x_{j_0+1}(m) + \beta x_{j_0}(m).$$

On the other hand, we have from (52) and (15) that

$$\begin{aligned} & \phi_{j_0-1}(\tau_{j_0 \oplus 1}(m+1)) \\ &= x_{j_0}(m) + \beta x_{j_0-1}(m) + x_{j_0+1}(m). \end{aligned}$$

Thus, for $j_0 - 1 \leq m \leq m_0$,

$$\begin{aligned} x_{j_0}(m+1) &= \phi_{j_0-1}((m+1)^+) - \phi_{j_0}((m+1)^+) \\ &= \phi_{j_0-1}(\tau_{j_0 \oplus 1}(m+1)) - \phi_{j_0}(\tau_{j_0 \oplus 1}(m+1)^+) \\ &= (1-\beta)x_{j_0}(m) + \beta x_{j_0-1}(m) \\ &\leq (1-\beta) + (j_0-1)(1-\beta) = j_0(1-\beta), \end{aligned}$$

where we use the fact that $x_{j_0}(m) \leq 1$ and the induction hypothesis in the inequality. \blacksquare

Proof. (Theorem 8) We first prove that node $n-1$ will be locked after n rounds of firings. Suppose that node $n-1$ has not been locked after $n-1$ rounds of firings. Then we have from Lemma 9 that for all $i = 1, \dots, n-1$,

$$x_i(n-1) \leq i(1-\beta). \quad (55)$$

Since $\sum_{i=1}^n x_i(n-1) = 1$,

$$\begin{aligned} x_n(n-1) &= 1 - \sum_{i=1}^{n-1} x_i(n-1) \geq 1 - \frac{n(n-1)}{2}(1-\beta) \\ &\geq \frac{1}{n}, \end{aligned}$$

where we use the assumption that $\beta \geq 1 - \frac{2}{n^2}$ in the last inequality. Note that node $n-1$ learns that node 0 is a locked node (in fact the anchored node) after the first two rounds of firings. According to the phase adjustment rule in (50) and (12) of Lemma 1,

$$\begin{aligned} & \phi_{n-1}(n^+) \\ &= \min \left[\frac{1}{n}, \phi_{n-1}(n) + \beta x_{n-1}(n) \right] \\ &= \min \left[\frac{1}{n}, x_n(n-1) + \beta x_{n-1}(n) \right] \\ &= \frac{1}{n}. \end{aligned} \quad (56)$$

Thus, $x_n(n) = 1/n$ and node $n-1$ will be locked after n rounds of firings.

Now we assume that node $n-i$, $i \leq i_0$ (for some $i_0 \geq 1$) are all locked after $n+2(i_0-1)$ rounds of firings as the induction hypothesis. Clearly, node $n-(i_0+1)$ learns that node $n-i_0$ is locked after $n-i_0$ fires for the $(n+2i_0)^{th}$ times. Suppose that node $n-(i_0+1)$ has not been locked after $n+2i_0-1$ rounds of firings. Then we have from Lemma 9 that for all $i = 1, \dots, n-(i_0+1)$,

$$x_i(n+2i_0-1) \leq i(1-\beta). \quad (57)$$

Since $\sum_{i=1}^n x_i(n+2i_0-1) = 1$ and $x_i(n+2i_0-1) = 1/n$ for $i = n-i_0+1, \dots, n$,

$$\begin{aligned} x_{n-i_0}(n+2i_0-1) &= 1 - \sum_{i=1}^{n-i_0-1} x_i(n+2i_0-1) \\ &\quad - \sum_{i=n-i_0+1}^n x_i(n+2i_0-1) \\ &\geq 1 - \frac{(n-i_0)(n-i_0-1)}{2}(1-\beta) - \frac{i_0-1}{n} \\ &\geq \frac{1}{n}, \end{aligned}$$

where we use the assumption that $\beta \geq 1 - \frac{2}{n^2}$ in the last inequality. According to the phase adjustment rule in (50) and (12) of Lemma 1,

$$\begin{aligned} & \phi_{n-(i_0+1)}(\tau_{n-i_0}(n+2i_0)^+) \\ &= \min \left[\frac{1}{n}, \phi_{n-(i_0+1)}(\tau_{n-i_0}(n+2i_0)) + \beta x_{n-(i_0+1)}(n) \right] \\ &= \min \left[\frac{1}{n}, x_{n-i_0}(n+2i_0-1) + \beta x_{n-(i_0+1)}(n+2i_0-1) \right] \\ &= \frac{1}{n}. \end{aligned} \quad (58)$$

Thus, node $n-(i_0+1)$ will be locked after $n+2i_0$ rounds of firings and the system achieves perfect desynchrony in $3n-4$ rounds of firings. \blacksquare

VI. COMPARISON WITH THE DESYNC-STALE ALGORITHM

In this section, we compare the rate of convergence of our anchored desynchronization and that of the DESYNC-STALE algorithm in [5]. As we mentioned before, the only difference between these two algorithms is that there is an anchored node in ours that never adjusts its phase. As such, one might expect that the rate of convergence of our algorithm would be slower than that of the DESYNC-STALE algorithm. However, to our surprise, we find out this is not always true from our numerical results and it is only true in the asymptotic regime. In Figure 4, we consider the case for five nodes, i.e., $n = 5$, and plot the second largest eigenvalue modulus (SLEM) of the matrix \mathbf{W} in (30), i.e., $\gamma + (1-\gamma) \cos(\frac{\pi}{n})$ in Proposition 4, and the SLEM of the matrix in (10) of [5] for n firings. From this figure, it is clear that the DESYNC-STALE algorithm is not always better. To explain this, we replace the jump size α in (12) of [5] by $1-\gamma$, and the characteristic polynomial in (10) of [5] can be rewritten as

$$\lambda^{n+1} - \frac{1-\gamma}{2}\lambda^2 - \gamma\lambda - \frac{1-\gamma}{2} = 0. \quad (59)$$

Note that the DESYNC-STALE algorithm may not converge if the jump size is chosen to be 1 (the maximum jump size). This is because there is an eigenvalue -1 in (59) if the jump size $1-\gamma$ is set to be 1 and n is an odd number. As such, when the jump size is close to 1 and n is an odd number, the rate of convergence of our algorithm is better than that of the DESYNC-STALE algorithm in [5].

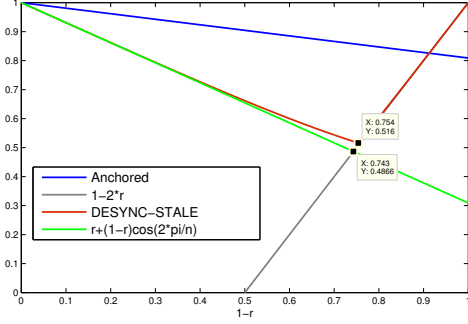


Fig. 4. Numerical results for the SLEM of the anchored desynchronization algorithm with $n = 5$ (marked in blue) and that of the corresponding DESYNC-STALE algorithm in [5] (marked in red) with respect to various jump size $1 - \gamma$ in $(0, 1)$

To better understand the rate of convergence of the DESYNC-STALE algorithm in [5], let λ_2 be one of the roots in (59) that corresponds to the SLEM of the matrix in (10) of [5]. As shown in [5], the polynomial in (59) is a stable polynomial and thus $|\lambda_2| < 1$ for $0 < \gamma < 1$. In the following, we derive an approximation for $|\lambda_2|^n$.

Our approach is to represent the roots in (59) by their polar coordinates. Specifically, we let $\lambda = Re^{i\theta}$, where $R \geq 0$, $0 \leq \theta < 2\pi$ and $\mathbf{i} = \sqrt{-1}$. Note that R must be larger than 0 as zero is not a root of (59) for $0 \leq \gamma < 1$. Moreover, except the trivial root $\lambda = 1$, it was shown in [5] that $|\lambda| < 1$ and thus $0 < R < 1$ for $0 < \gamma < 1$. Now we can rewrite (59) as follows:

$$\lambda^n = \gamma + (1 - \gamma)\left(\lambda + \frac{1}{\lambda}\right). \quad (60)$$

It then follows from (60) that

$$R^n \cos(n\theta) = \gamma + \frac{1 - \gamma}{2}\left(R + \frac{1}{R}\right) \cos(\theta), \quad (61)$$

and

$$R^n \sin(n\theta) = \frac{1 - \gamma}{2}\left(R - \frac{1}{R}\right) \sin(\theta). \quad (62)$$

In particular, if $\theta = 0$, then we have from (61) that

$$R^n = \gamma + \frac{1 - \gamma}{2}\left(R + \frac{1}{R}\right) \geq 1, \quad (63)$$

where we use the fact that $(R + \frac{1}{R})/2 \geq 1$. Thus, we must have $R = 1$ for $\theta = 0$ and this corresponds to the trivial root $\lambda = 1$. As such, we only need to consider $0 < \theta < 2\pi$ for the other roots.

Let $\lambda_2 = R_2 e^{i\theta_2}$. Since λ_2 corresponds to the SLEM of the characteristic polynomial in (59), it seems plausible to make the following approximation

$$R_2 \approx 1. \quad (64)$$

Thus, we have from (61) and (62) that

$$R_2^n \cos(n\theta_2) \approx \gamma + (1 - \gamma) \cos(\theta_2), \quad (65)$$

and

$$R_2^n \sin(n\theta_2) \approx 0. \quad (66)$$

In view of (66), we know that $\theta_2 \approx \frac{k\pi}{n}$ for some integer k in $[1, 2n - 1]$ and thus $\cos(n\theta_2) \approx (-1)^k$. This leads to

$$R_2^n \approx (-1)^k \left(\gamma + (1 - \gamma) \cos\left(\frac{k\pi}{n}\right)\right). \quad (67)$$

If n is an odd number, the right hand side of (67) is maximized when $k = 2$ or $k = n$. This leads to

$$|\lambda_2|^n = R_2^n \approx \max\left[\gamma + (1 - \gamma) \cos\left(\frac{2\pi}{n}\right), 1 - 2\gamma\right] \quad (68)$$

for an odd n . On other hand, if n is an even number, the right hand side of (67) is maximized when $k = 2$ or $k = n + 1$.

$$\begin{aligned} |\lambda_2|^n &= R_2^n \\ &\approx \max\left[\gamma + (1 - \gamma) \cos\left(\frac{2\pi}{n}\right), -\gamma + (1 - \gamma) \cos\left(\frac{\pi}{n}\right)\right] \end{aligned} \quad (69)$$

for an even n . As clearly shown in Figure 4, the approximations in (68) matches very well to the true numerical values for $n = 5$. Through extensive numerical computations, we find that the approximations in (68) and (69) are also extremely good for other values of n .

Note from the approximations in (68) and (69) that for *very large* n and $0 < \gamma < 1$

$$|\lambda_2|^n \approx \gamma + (1 - \gamma) \cos\left(\frac{2\pi}{n}\right). \quad (70)$$

It is of some interest to compare (70) with the SLEM for \mathbf{W} in Proposition 4, i.e., $\gamma + (1 - \gamma) \cos(\frac{\pi}{n})$. This shows that the rate of convergence for the DESYNC-STALE algorithm is faster than that of our anchored desynchronization algorithm in the asymptotic regime. The intuition behind this might be explained by considering the near perfect desynchrony scenario. When the state of the system is near perfect desynchrony, the stale estimate is almost the same as its true value. If this is the case, the $n \times n$ governing matrix \mathbf{W}^D for the DESYNC-STALE algorithm could be modified from \mathbf{W} by ‘‘adjusting’’ the phase of the anchored node in \mathbf{W} as well. Thus, we have

$$W_{ij}^D = \begin{cases} \frac{1+\gamma}{2} & \text{for } i = 1, j = n \text{ or } i = n, j = 1, \\ \gamma & \text{for } i = j = 1, 2, \dots, n, \\ \frac{1-\gamma}{2} & \text{for } i = j - 1 = 1, 2, \dots, n - 1, \\ \frac{1-\gamma}{2} & \text{for } i = j + 1 = 2, \dots, n, \\ 0 & \text{otherwise.} \end{cases} \quad (71)$$

In particular, if $n = 5$, then

$$\mathbf{W}^D = \begin{pmatrix} \gamma & \frac{1-\gamma}{2} & 0 & 0 & \frac{1-\gamma}{2} \\ \frac{1-\gamma}{2} & \gamma & \frac{1-\gamma}{2} & 0 & 0 \\ 0 & \frac{1-\gamma}{2} & \gamma & \frac{1-\gamma}{2} & 0 \\ 0 & 0 & \frac{1-\gamma}{2} & \gamma & \frac{1-\gamma}{2} \\ \frac{1-\gamma}{2} & 0 & 0 & \frac{1-\gamma}{2} & \gamma \end{pmatrix}.$$

In conjunction with \mathbf{W} in (31), we note that \mathbf{W} (resp. \mathbf{W}^D) corresponds to the distributed averaging problem in [19] over a *line* (resp. *ring*) graph with n nodes. It is well-known (by using the Birkhoff decomposition [1] for the doubly stochastic matrix in (71)) that the SLEM of \mathbf{W}^D is $\gamma + (1 - \gamma) \cos(\frac{2\pi}{n})$, which is smaller than that of \mathbf{W} .

VII. CONCLUSION

In this paper, we considered the desynchronization problem in a system based on pulse-coupled oscillators. Unlike the schemes in [5], [15], we assume there exists an anchored node that never adjusts the phase of its oscillator. For such a system, we proposed a generic anchored desynchronization algorithm similar to the DESYNC-STALE algorithm in [5]. Though the only difference between our anchored desynchronization algorithm and the DESYNC-STALE algorithm in [5] is the anchored node, we are able to rigorously prove the rate of convergence of our algorithm. Specifically, we show that our algorithm achieves ϵ -desynchrony in $O(n^2 \ln(\frac{n}{\epsilon}))$ rounds of firings. We also proved that our anchored desynchronization algorithm converges even for the generalized processor sharing (GPS) scheme previously studied in [15]. When the information of the total number of nodes in the system is available to all the nodes, we proposed a set of algorithms that achieve perfect desynchrony in $3n - 4$ rounds of firings. In terms of the rate of convergence, the DESYNC-STALE algorithm in [5] is not always better than ours. For this, we derived an approximation for the SLEM of the matrix in (10) of [5] for n firings.

There are three possible extensions.

- (i) Randomized algorithms: here we only assume that f_i 's are deterministic functions. Analogous to the extension of the deterministic distributed averaging algorithms in [19] to the randomized gossip algorithms in [3], we note that it is also possible to extend our analysis to random functions, e.g., with a certain probability a node will not adjust its phase when its next node fires.
- (ii) Multihop setting: here we consider the setting with a complete graph, i.e., every node can hear (and interfere with) every other node. Extension to the setting with a general interfere graph (see e.g., [7], [12]) appears to be much more difficult. Unlike the single hop setting, where perfect desynchrony is capacity achieving, approaches like graph coloring only yield feasible transmission schemes (or matchings) and they are not guaranteed to be capacity achieving as the maximum weighted matching in [18] and the dynamic frame sizing algorithm in [9]
- (iii) Additional memories: here we only need to store the information of the firing times of the two neighboring nodes. Recently, it was shown in [14] that the rate of convergence for the distributed averaging algorithm could be improved by adding a little bit more memories. Research along this line requires further investigation for our anchored desynchronization algorithms.

REFERENCES

- [1] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, Vol. 5, pp. 147-151, 1946.
- [2] S. Boyd, P. Diaconis, L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, Vol. 46, No. 4, pp. 667-689, 2004.
- [3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, Vol. 52, No. 6, pp. 2508-2530, 2006.
- [4] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of markov chains," *The Annals of Applied Probability*, Vol. 1, pp. 36-61, 1991.
- [5] J. Degeysys, I. Rose, A. Patel, and R. Nagpal, "Desync: Self-organizing desynchronization and TDMA on wireless sensor networks," in *International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.
- [6] J. Degeysys, I. Rose, A. Patel, R. Nagpal, "Desynchronization: the theory of self-organizing algorithms for round-robin scheduling," *First International Conference on Self-Adaptive and Self-Organizing Systems*, 2007. SASO '07.
- [7] J. Degeysys and R. Nagpal, "Towards Desynchronization of Multi-hop Topologies," *Second International Conference on Self-Adaptive and Self-Organizing Systems*, 2008. SASO '08.
- [8] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Society, 2009.
- [9] C.-M. Lien, C.-S. Chang, J. Cheng, and D.-S. Lee, "Maximizing throughput in wireless networks with finite internal buffers," *Proc. of IEEE INFOCOM 2011*.
- [10] A.W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*. New York: Academic Press, 1979.
- [11] R. Mirollo and S. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal of Applied Math*, Vol. 50, No. 6, pp. 1645V62, Dec. 1990.
- [12] A. Motzkin, T. Roughgarden, P. Skraba and L. Guibas, "Lightweight coloring and desynchronization for networks," *Proc. of IEEE INFOCOM 2009*.
- [13] R. Nelson, *Probability, Stochastic Processes, and Queueing Theory: the Mathematics of Computer Performance Modeling*. Springer-Verlag: New York, 1995.
- [14] B. N. Oreshkin, M. J. Coates, and M. G. Rabbat, "Optimization and analysis of distributed averaging with short node memory," *IEEE Transactions on Signal Processing*, Vol. 58, No. 5, pp. 2850-2865, 2010.
- [15] R. Pagliari, Y.-W. Hong, and A. Scaglione, "Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling," *IEEE Journal on Selected Areas in Communications: Special Issue on Bio-Inspired Networking*, Vol. 28, No. 4, pp. 564-575, May 2010.
- [16] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: the single-node case," *IEEE/ACM Trans. Networking*, Vol. 1, pp. 344-357, 1993.
- [17] C. S. Peskin. *Mathematical Aspects of Heart Physiology*. Courant Institute of Mathematical Sciences, New York University, New York, 1975.
- [18] L. Tassioulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 31, no. 12, pp. 1936-1948, 1992.
- [19] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, Vol. 53, No. 1, pp. 65V78, Sep. 2004.
- [20] W.-C. Yueh, "Eigenvalues of several tridiagonal matrices," *Applied Mathematics E-Notes*, pp. 66-74, No. 5, 2005. Available free at mirror sites of <http://www.math.nthu.edu.tw/~amen/>