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 [4], [14] 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 [4]) 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. In comparison with the original algorithm in [4], we show that the rate of convergence of the original algorithm in [4] 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 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 [4] considered a general framework for distributed algorithms to achieve desynchronization needed in TDMA. In their framework, nodes are modelled by *pulse-coupled oscillators* in [16], [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 [4] 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 [4] 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.

Pagliari, Hong and Scaglione [14] considered an important extension of the fair resource scheduling scheme to the generalized processor sharing (GPS) scheme [15], 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 näive 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 [14], 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 [14] 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 [4] and the extension of the GPS scheme in [14] 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 [14]. 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

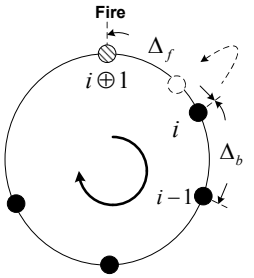

Fig. 1. Illustration of the phase adjustment of node i immediately after node $i \oplus 1$ fires (the white node indicates the new phase position of node i)

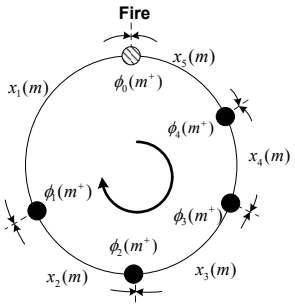

Fig. 2. The state of the system with $n = 5$ at time m^+ (the anchored node is marked with stripes)

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 [4]) 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 [4]. (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 [14]. (iii) In comparison with the DESYNC-STALE algorithm in [4], we show that the rate of convergence of the DESYNC-STALE algorithm in [4] is not always better than ours and it is only better in the asymptotic regime.

Due to space limitation, all the mathematical proofs are omitted here and we refer the readers to our full technical report [9].

II. Anchored desynchronization framework

As in [4], 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, $i = 0, 1, \ldots, n-1$. Upon reaching $\phi_i(t) = 1$, 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

$$\phi_i(\tau^+) = f_i(\phi_i(\tau), \Delta_b, \Delta_f), \tag{1}$$

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:

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

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} time 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, $m = 1, 2, \ldots$. In order to make sure that every node fires according to the desired order, i.e., node 0, node 1, node 2, ..., 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 $i = 1, 2, \ldots, n-1$. Then, node i will adjust its phase such that its phase satisfies the following inequality:

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

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

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

where

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

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

$$\begin{aligned}\mathbf{x}(0) &= (1-\phi_1(0), \phi_1(0)-\phi_2(0), \ldots, \\ &\quad \phi_{n-2}(0)-\phi_{n-1}(0), \phi_{n-1}(0))^T\end{aligned}$$

and the state vector indeed contains nonzero elements. Let $\tau_i(m)$ be the time that node i fires for the m^{th} time. Then, the governing dynamics of the system can be derived as follows.

Lemma 1: For some $i = 1, 2, \ldots, n-1$, 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. Under Algorithm 1, the following holds:

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

$$\phi_i(\tau_{i\oplus 1}(m+1)) = x_{i+1}(m). \quad (5)$$

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

$$\begin{aligned}&\phi_i(\tau_{i\oplus 1}(m+1)^+) \\ &= f_i(x_{i+1}(m), x_i(m), x_{i+1}(m)).\end{aligned} \quad (6)$$

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

$$\begin{aligned}&\phi_{i-1}(\tau_{i\oplus 1}(m+1)) = \phi_0(\tau_2(m+1)) \\ &= x_1(m) + x_2(m).\end{aligned} \quad (7)$$

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

$$\begin{aligned}&\phi_{i-1}(\tau_{i\oplus 1}(m+1)) \\ &= f_{i-1}(x_i(m), x_{i-1}(m), x_i(m)) + x_{i+1}(m).\end{aligned} \quad (8)$$

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 $i = 1, 2, \ldots, n-1$, 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 $i = 1$

$$x_1(m) + x_2(m) > f_1(x_2(m), x_1(m), x_2(m)) > 0, \quad (9)$$

and for $i > 1$

$$\begin{aligned}&f_{i-1}(x_i(m), x_{i-1}(m), x_i(m)) + x_{i+1}(m) \\ &> f_i(x_{i+1}(m), x_i(m), x_{i+1}(m)) > 0.\end{aligned} \quad (10)$$

III. Generic anchored desynchronization

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

$$f_i(\phi_i(\tau), \Delta_b, \Delta_f) = \gamma\phi_i(\tau) + (1-\gamma)\frac{\Delta_b + \Delta_f}{2}, \quad (11)$$

for $0 \le \gamma < 1$ and $i = 1, 2, \ldots, n-1$. Note that the phase adjustment rule in (11) is exactly the same as that in the DESYNC-STALE algorithm in [4]. 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 [4]. Thus, the only difference between the DESYNC-STALE algorithm in [4] and ours is the anchored node. Though it was shown in [4] 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.

For $0 \le \gamma < 1$, the governing dynamics of the phases can be derived as shown in our technical report [9] (Lemma 3), and the dynamics of phase difference $\mathbf{x}$ can be thus represented in the matrix form as

$$\mathbf{x}(m+1) = \mathbf{W}\mathbf{x}(m), \quad (12)$$

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

$$W_{ij} = \begin{cases} \frac{1+\gamma}{2} & \text{for } i=j=1 \text{ or } i=j=n, \\ \gamma & \text{for } i=j=2,\ldots,n-1, \\ \frac{1-\gamma}{2} & \text{for } i=j-1=1,2,\ldots,n-1, \\ \frac{1-\gamma}{2} & \text{for } i=j+1=2,\ldots,n, \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

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

$$\sum_{i=1}^{n} W_{ij} = 1, \quad j = 1, 2, \ldots, n, \quad (14)$$

and

$$\sum_{j=1}^{n} W_{ij} = 1, \quad i = 1, 2, \ldots, n. \quad (15)$$

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 (12), 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 in 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 3: The n eigenvalues of the $n \times n$ matrix $\mathbf{W}$ are $\gamma + (1-\gamma)\cos(\frac{i\pi}{n})$, $i = 0, 1, \ldots, n-1$. Thus, the largest eigenvalue of $\mathbf{W}$ is 1. Moreover, for $0 \le \gamma \le 1$, 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 $\gamma + (1-\gamma)\cos(\frac{\pi}{n})$.

Analogous to the definition of desynchronization accuracy in [5], 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 firings if

$$\sum_{i=1}^{n} \left| x_i(m) - \frac{1}{n} \right| \leq \epsilon. \tag{16}$$

Theorem 4: 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 (11), achieves ϵ-desynchrony in $O(n^2 \ln(\frac{n}{\epsilon})/(1-\gamma))$ rounds of firings.

IV. Generalized processor sharing

An important extension of the fair resource scheduling scheme is the generalized processor sharing (GPS) scheme addressed in [15], [14]. 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, $i = 0, 1, \ldots, n-1$. The objective of this section is to propose an anchored desynchronization algorithm so that for all i

$$\lim_{m\to\infty} \left| x_i(m) - \frac{\alpha_{i-1}}{\sum_{j=0}^{n-1} \alpha_j} \right| = 0. \tag{17}$$

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

$$f_i(\phi_i(\tau), \Delta_b, \Delta_f) = \gamma\phi_i(\tau) + (1-\gamma)\beta_i(\Delta_b + \Delta_f), \tag{18}$$

where

$$0 < \beta_i = \alpha_i/(\alpha_{i-1} + \alpha_i) < 1. \tag{19}$$

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 [14], 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 $\alpha_0, \delta, \alpha_1, \delta, \ldots, \alpha_{n-1}, \delta$. 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 [14], the constant δ needs to be relatively small to ensure "proportional fairness."

Analogous to the derivation for the recursive equation for the state vector in (12), the governing dynamics of the system with generalized processor sharing can be represented as

$$\mathbf{x}(m+1) = \tilde{\mathbf{W}}\mathbf{x}(m), \tag{20}$$

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

$$\tilde{W}_{ij} = \begin{cases} 1-(1-\gamma)\beta_1, & \text{for } j = i = 1, \\ (1-\gamma)(1-\beta_i), & \text{for } j = i+1, \\ \gamma + (1-\gamma)(\beta_{i-1} - \beta_i), & \text{for } j = i \neq 1 \text{ or } n, \\ (1-\gamma)\beta_{i-1}, & \text{for } j = i-1, \\ 1-(1-\gamma)(1-\beta_{n-1}) & \text{for } j = i = n, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Then, the convergence of the system with generalized processor sharing can be guaranteed as follows.

Theorem 5: Suppose that the functions f_i, $i = 1, 2, \ldots, n-1$, are chosen in (18) with $1/2 \leq \gamma < 1$. Under Algorithm 1, the state vector $\mathbf{x}$ of the system of n nodes converge to $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, .i.e.,

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

V. 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 [4]. 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. In Figure 3, 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 (12), i.e., $\gamma + (1-\gamma)\cos(\frac{\pi}{n})$ in Proposition 3, and the SLEM of the matrix in (10) of [4] 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 [4] by $1-\gamma$, and the characteristic polynomial in (10) of [4] can be rewritten as

$$\lambda^{n+1} - \frac{1-\gamma}{2}\lambda^2 - \gamma\lambda - \frac{1-\gamma}{2} = 0. \tag{22}$$

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 (22) 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 [4].

To better understand the rate of convergence of the DESYNC-STALE algorithm in [4], let λ_2 be one of the roots in (22) that correspond to the SLEM of the matrix in (10) of [4]. As shown in [4], the polynomial in (22) 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 (22) by their polar coordinates. Specifically, we let $\lambda_2 = R_2 e^{\mathbf{i}\theta_2}$, where $R \geq 0$, $0 \leq \theta < 2\pi$ and $\mathbf{i} = \sqrt{-1}$. Since λ_2 corresponds to the SLEM of the characteristic polynomial in (22), it seems plausible to make the following approximation

$$R_2 \approx 1. \tag{23}$$

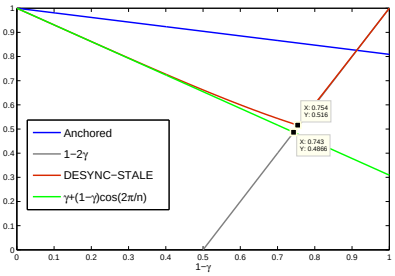

Fig. 3. Numerical results for the SLEM of the anchored desynchronization algorithm with $n = 5$ (marked in blue) and that of the corresponding DESYNC-STALE algorithm in [4] (marked in red) with respect to various jump size $1-\gamma$ in $(0, 1)$

Then, one can verify that

$$|\lambda_2|^n = R_2^n \approx \max\left[\gamma + (1-\gamma)\cos\left(\frac{2\pi}{n}\right), 1-2\gamma\right] \quad (24)$$

for an odd n, and

$$\begin{aligned} &|\lambda_2|^n = R_2^n \\ &\approx \max\left[\gamma + (1-\gamma)\cos\left(\frac{2\pi}{n}\right), -\gamma + (1-\gamma)\cos\left(\frac{\pi}{n}\right)\right] \end{aligned} \quad (25)$$

for an even n. As clearly shown in Figure 3, the approximations in (24) matches very well to the true numerical values for $n = 5$. Through extensive numerical computations, we find that the approximations in (24) and (25) are also extremely good for other values of n. For more discussions, especially for the connections to the distributed averaging problem [18], please see the full report [9].

VI. Conclusion

In this paper, we considered the desynchronization problem in a system based on pulse-coupled oscillators. Unlike the schemes in [4], [14], 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 [4]. Though the only difference between our anchored desynchronization algorithm and the DESYNC-STALE algorithm in [4] is the anchored node, we are able to rigourously 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 [14]. In terms of the rate of convergence, the DESYNC-STALE algorithm in [4] is not always better than ours. For this, we derived an approximation for the SLEM of the matrix in (10) of [4] for n firings.

There are two 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 [18] to the randomized gossip algorithms in [2], 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., [6], [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 [17] and the dynamic frame sizing algorithm in [8].

References

[1] S. Boyd, P. Diaconis, L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, Vol. 46, No. 4, pp. 667-689, 2004.

[2] 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.

[3] P. Diconis and D. Stroock, "Geometric bounds for eigenvalues of markov chains, *The Annals of Applied Probability*, Vo. 1, pp. 36–61, 1991.

[4] J. Degesys, 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.

[5] J. Degesys, 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.

[6] J. Degesys and R. Nagpal, "Towards Desynchronization of Multi-hop Topologies," *Second International Conference on Self-Adaptive and Self-Organizing Systems*, 2008. SASO '08.

[7] D. A. Levin, Y.Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Socieity, 2009.

[8] 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*.

[9] C.-M. Lien, S.-H. Chang, C.-S. Chang, and D.-S. Lee, "Anchored desynchronization," *Techincal Report*, 2011. Available from http://www.ee.nthu.edu.tw/cschang/AnchoredTechnicalReport.pdf.

[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. Motskin, 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] 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.

[15] 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.

[16] C. S. Peskin. *Mathematical Aspects of Heart Physiology.* Courant Institute of Mathematical Sciences, New York University, New York, 1975.

[17] L. Tassiulas 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.

[18] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, Vol. 53, No. 1, pp. 65-78, Sep. 2004.