Synchronization of Strongly Pulse-Coupled Oscillators with Refractory Periods and Random Medium Access

Julius Degesys
BBN Technologies
10 Moulton Street
Cambridge, MA
jdegesys@bbn.com

Prithwish Basu
BBN Technologies
10 Moulton Street
Cambridge, MA
pbasu@bbn.com

Jason Redi
BBN Technologies
10 Moulton Street
Cambridge, MA
redi@bbn.com

ABSTRACT

The weakly pulse-coupled oscillator framework has proven to be a valuable resource for the development of peer-to-peer synchronization algorithms [9]. But leveraging it in a practical implementation (e.g. in wireless ad hoc/sensor networks) is problematic due to the difficulty in achieving precise coordination of broadcast messages. We found that a pseudo-random medium access control (MAC) protocol produces a *super-linear* increase in the number of messages required per node with increasing network size, which would normally discourage its use. However, introducing a "refractory period" experimentally reduces this growth to *linear* with a small constant. Furthermore, the refractory period allows for an increase in the coupling constant, effectively making the network "strongly" pulse-coupled. We show that the combination of the refractory period, strong coupling, and probabilistic medium access results in a significant decrease in the average number of messages required per node in several practical network topologies (and as much as $\sim 90\%$ over the original idealistic mechanism in line topologies).

Categories and Subject Descriptors: C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms: Algorithms, Design, Performance

Keywords: Synchronization, self-organizing, sensor networks, pulse-coupled oscillators

1. INTRODUCTION

Wireless networks often require nodes to coordinate periodic actions. For instance, wireless sensor networks (WSNs) are often required to gather data from all sensors periodically for global monitoring of the environment. Highly duty-cycled networks (for energy efficiency) might wake up only for short periods of time in between long periods of inactivity. In order to communicate in such scenarios, nodes must synchronize their wake-up times effectively.

Achieving event synchronization[1] in WSNs is challenging due their potentially large scale and ad hoc deployment. In addition, achieving synchronization once is not sufficient. Clock manufacturing defects, noisy channels, and changing environmental conditions imply that synchronization must be maintained continually for the life-span of the network. Furthermore, WSN devices usually have limited energy supplies, thus obviating the need for energy efficient sync algorithms.

Centralized mechanisms achieve synchronization by distributing a single consistent value, thereby eliminating any ambiguity. Global Positioning System (GPS) chips, for example, are able to provide time with a 40-nanosecond precision [14]. However, these devices have high power consumption when compared to the radios and processors that typically comprise sensor networks [13]. Furthermore, GPS neither works indoors nor underground, making its applicability limited.

Distributed leader-election and tree-formation algorithms disseminate a single node's information as ground truth [3]. But this can leave a network in a vulnerable state, as the failure of a critical node could have to be reported to the entire network before a recovery can occur, resulting in many messages being transmitted. Since failures are expected to be the norm in large WSNs, global repair mechanisms should be generally avoided. Ideally, the failure of a node should only elicit a local recovery mechanism.

An attractive model that robustly achieves event synchronization in a peer-to-peer fashion happens to be a biologically-inspired one—one that mimics the synchronization of the cardiac pacemaker, neurons, and even fireflies. Originally inspired from Peskin's lecture notes on cardiac-cell synchronization, Mirollo and Strogatz wrote a seminal paper that described how a fully-connected system of pulse-coupled oscillators would provably converge to synchrony [9]. Lucarelli and Wang, extended this work to show convergence for time-varying, multi-hop topologies [7]. In summary, nodes periodically fire, i.e., broadcast a message, and upon receiving

Prepared through collaborative participation in the Collaborative Technology Alliance for Communications & Networks sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0011.

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

[1]It should be noted that a distinction exists between *event* synchronization and *time* synchronization. While the latter can trivially achieve the former, time synchronization is bound by algorithms that must maintain causality. This implies that clocks can only be set forward in time. Whereas event synchronization algorithms can move events either backwards or forwards in time. This is discussed thoroughly in Lamport's classic paper [6].

these messages, nodes adjust their local timings via a simple update equation. As a result, the entire network is *eventually* driven to synchrony.

A further extension of the biological analogy would consider a *refractory period* which is a period immediately after firing during which a node ignores all received firings. Biological systems primarily exhibit this as a means of preventing detrimental oscillations. Remarkably, even in this regard, to the authors' knowledge, it has received little attention [5]. It has been stated [4] that including these realistic effects "would result in a pulse-coupled model that would be as complex and cumbersome as the Hodgkin-Huxley[2] neurons." However, this paper presents novel experimental results indicating that the inclusion of a refractory period is instrumental in improving the convergence rate.

Due to its distributed and localized nature, this algorithm is inherently robust to node failure and topology changes. Regardless, the fundamental assumption of messages being sent once a *firing period* effectively eliminates the feasibility of using this algorithm for applications with a small period, e.g., slot synchronization for time-division multiple access (TDMA) MAC protocols. Time is divided into tiny slots, and each node's perception of when a slot begins must be synchronized for accurate functioning of a TDMA-style MAC. Here it is clearly not possible for each node to send a message at the beginning of each slot because each time slot is normally dedicated only to a single node. Additionally, the slot size may be picked to match the size of a single message, making it likely that a node will need to fire again before it has even sent its first message. This can result in either an overflow of the outgoing message queue or dropped messages.

Inspired by a pseudo-random TDMA MAC protocol [2, 11], this paper considers the case where firing messages are sent probabilistically (this is further explained in section 3.3). For a sufficiently low broadcast transmit probability, message collisions can effectively be eliminated. Without refractory periods, this modification greatly increases the total number of messages required by the original algorithm to achieve convergence (compare, for example, figures 1(d) and (e)). With refractory periods, it is possible to drastically reduce the number of messages required to *below* the value required when *all messages* were being transmitted with no refractory period.

Furthermore, with randomness it becomes possible to increase the coupling between oscillators from weakly coupled to what could be considered *strongly coupled*. Using the optimal value for the coupling (independent of the topology and its size), the algorithm is effectively reduced to triviality, consisting of a single comparison and the possible copy of a variable.

In summary, the contributions of this paper are as follows: numerical analysis reveals that the refractory period significantly improves the convergence rate and the total number of number of messages required to achieve synchronization. We show that a refractory period of half a period (slot) is ideal in all topologies considered and this also makes the algorithm extremely robust to probabilistic firings. Furthermore, in this setting, strong coupling of the oscillators proves to be beneficial, thus eliminating the arbitrariness in selecting the coupling constant for actual deployments. Effectively, we observe that synchronization algorithms can benefit by appropriately utilizing *less information*. For instance, in a 20-node line topology, approximately *15 times fewer messages per node* are required on average by the strongly pulse-coupled algorithm that utilizes a refractory period and probabilistic firings than the weakly pulse-coupled, linear algorithm utilizing *all messages* and no refractory period.

The rest of the paper is organized as follows: section 2 presents assumptions that our model makes, section 3 presents our extensions to the weakly pulse-coupled model proposed in [9], and section 4 presents simulation results and their interpretation. Section 5 concludes the paper.

[2]Nobel prize-winning work in 1963 that models how action potentials in neurons are initiated and propagated.

2. ASSUMPTIONS

This paper makes the following simplifying assumptions:

No Propagation Delay: The propagation delay in WSNs is bounded by the speed of light in air. In the deployments considered in this paper, this is assumed to be in the submicrosecond range and hence negligible.

No Transmission Delays: We assume precise transmission timing of wireless messages since it is possible to maintain one-to-one correspondence between symbol-level timing in the PHY and local clock ticks and this can be exploited in the MAC layer of the network stack if running on a real-time operating system [1]. While popular sensor network operating systems such as TinyOS on Mica2 motes suffer non-deterministic transmission delays in *milliseconds*, it is possible to overcome that by modeling clock drift with linear regression and performing MAC-level time-stamping [8].

No Clock Drift: Manufacturing imprecision causes clocks to run with varying degrees of accuracy. However, for stable external conditions and time scales on the order of minutes, a crystal oscillator can be accurately modeled as running at an unknown, bounded, *fixed* frequency. For larger time scales, environmental conditions can vary greatly, likely necessitating an online linear regression algorithm or a more complex model. But for time scales on which the synchronization algorithm is running, linear regression permits clock drift to be safely assumed to be negligible.

Bidirectional Communication: We assume symmetric communication between nodes i and j which is achievable if nodes have calibrated, homogeneous transmission powers.

This paper primarily focuses on *how to increase the scalability of a biologically-inspired peer-to-peer synchronization algorithm by reducing the number of total messages required to achieve synchronization*, thereby implying a reduction in total energy consumed.

3. SYNCHRONIZATION FRAMEWORK

In this section a general framework is presented for exploring pulse-coupled, peer-to-peer algorithms.[3] Let $G = (V, E)$ denote a graph where V is an n-dimensional vertex set and $E \subseteq V \times V$. Since bidirectional communication is assumed, all edges are undirected. In other words, $e = (v_i, v_j) \in E \Rightarrow (v_j, v_i) \in E$. The set of neighbors of vertex i is denoted by $N_i = \{j : (v_i, v_j) \in E\}$ (i.e., the set of vertices that have an incoming edge from vertex i).

3.1 Pulse-coupled Oscillators

Each vertex in G has an associated oscillator with a natural frequency of ω. The oscillator represents the local tim-

[3]See Izhikevich's tutorial on the topic for further details [4].

ing of a periodic task. The state of oscillator i is scaled such that $x_i \in \mathbb{S}^1 = [0, 1]$ with 0 and 1 equated. When oscillator i reaches $x_i = 1$, it *fires*, emitting a pulse to its neighbors[4] and resetting itself to 0. When oscillator j receives this firing, it updates its local state via the following update equation:

$$x'_j = \begin{cases} x_j + \epsilon x_j, & \text{if } x_j + \epsilon x_j < 1 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Thus, all oscillators utilize the same update equation, jumping by ϵx_j when any oscillator $i \in N_j$ fires. The synchronized state, or synchrony, is the state in which all oscillators fire at the same time (i.e., $\vec{x}^* = [1, 1, \ldots, 1]^T$ at the point of firing).

It is important to note that if a jump would cause x_j to exceed the firing threshold, x_j is set to 0 instead of 1. If this were not the case, very large ϵ could immediately drive the system to synchrony the instant the first oscillator fired. This solution is not robust, however; once any propagation delay is introduced into the system, nodes would continuously fire, as each firing would trigger more firings.

A salient feature of this algorithm is that once oscillator i fires, its neighbors require only local information (i.e. their own states) to compute their update equations.[5]

3.2 Refractory Period

The update performed in equation (1) may seem counterintuitive. Consider an oscillator that has just fired. If a firing were to occur shortly thereafter, it would cause the states of these two oscillators to be driven further apart (see, for example, the firing oscillator and oscillator i in figure 1(b)). Intuitively, they should be driven closer together with a nearby firing, or at the very least, maintain the current separation. When viewed in this light, it seems natural that a refractory period (a period in which a node ignores all incoming firings) might be beneficial to increasing the rate of convergence. The update equation including a refractory period, $x_r \in [0, 1]$, can be written as:

$$x'_j = \begin{cases} x_j, & \text{if } x_j < x_r \\ x_j + \epsilon x_j, & \text{if } x_j \geq x_r \text{ and } x_j + \epsilon x_j < 1 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Figures 1(b & c) visually show a response to a firing.

3.3 Pseudo-Random Medium Access

Event synchronization at nodes also results in the synchronization of message transmissions. Despite the fact that only a single bit is required by the receiving node to perform its update, more information (such as preamble) needs to be sent to detect this bit in wireless media. If two matching symbols from different sources are not exactly aligned, then they will interfere with one another, making it difficult to detect either transmission. Due to the limited precision of clocks and radios' transmission frequencies, exactly aligning neighbors' preambles is practically impossible. Thus, nodes need to spread out their transmissions in time to reduce collisions. In channel-sensing protocols, if the sampled average power on the media is below a threshold, the channel is assumed available and the node begins transmission. Otherwise, the channel is assumed busy, and the node waits. As long as the messages are eventually sent, and the MAC delay is accurately contained in the message, a reach-back algorithm can still achieve synchronization [12, 15]. However, channel sensing is inherently noisy and difficult to perform accurately, and it is still likely that messages will collide thus requiring more messages to be retransmitted.

Rather than deliberately congesting the channel, an alternative method is to utilize pseudo-randomness to compensate for message collisions [2, 11]. This mechanism is effectively equivalent to oscillators only firing with probability p_f. If p_f is low enough, collisions can be effectively eliminated. Although this increases the time required to transmit the same number of messages, if correspondingly fewer messages need to be sent for achieving synchrony, then there is no penalty. The main benefit of this mechanism is that reach-back mechanisms do not need to be performed since messages will almost never collide. The rest of the paper will use a 3-tuple, $\langle \epsilon, x_r, p_f \rangle$, to define the relevant parameters.

4. SIMULATION RESULTS

In this section, we present numerical simulations to illustrate the various properties of equation (2). Note that these simulations did *not* simulate collisions. Where necessary, we assumed that a sufficiently small p_f is selected to achieve this. Figure 1(d) shows a representative trial for $\langle \epsilon = 0.1,\ x_r = 0,\ p_f = 1 \rangle$. Figure 1(e) shows the effects of introducing $p_f = 0.2$ as a medium access control mechanism. As can be seen, this greatly increases the number of messages required.

The cumulative distribution functions (CDFs) for varying 20-node topologies are shown in figures 2(a-c). The random graph was constructed by drawing 20 two-dimensional coordinates from a uniform distribution. All nodes within 0.2 units of one another were joined with an edge. This process was repeated until a connected graph was achieved. Figure 2(a) shows $\langle \epsilon = 0.1,\ x_r = 0,\ p_f = 1.0 \rangle$. Here, the ring topology proved to be the most difficult to synchronize. Introducing $p_f = 0.2$ in 2(b) greatly increases all topologies, and made the line graph the worst. Similarities between the line and the random graph are due to the random graph having a very linear structure with only a few cycles. Lastly, by adding a refractory period of $x_r = 0.5$ in 2(c), the convergence time for all topologies is greatly reduced, beyond even that where all messages are received. Due to paucity of space, the line topology was selected for the experiments in figure 3. However, the conclusions qualitatively hold for the other considered topologies.

Figures 3(a) and (b) show the effect that the refractory period has on a 20-node line topology with $\epsilon = 0.1$ and $p_f = 0.2$. Notice that simply by introducing a refractory period, the number of messages required to reach convergence is greatly reduced from the super-linear growth of $\langle \epsilon = 0.1,\ x_r = 0,\ p_f = 0.2 \rangle$. A minimum is reached with $\langle \epsilon = 0.1,\ x_r = 0.5,\ p_f = 0.2 \rangle$. For $x_r \geq 0.5$, the system will no longer converge for all initial conditions (two oscillators separated by 0.5, i.e. $|x_i - x_j| = 0.5$, would never move). Note that for $p_f = 1$, the same qualitative results also hold but are excluded for paucity of space.

In effect, a refractory period corresponds to an oscillator selectively ignoring information; an oscillator has heard a

[4] This is tantamount to broadcasting a wireless message.

[5] It should also be noted that equation (1) is only one of many such algorithms that leads to synchrony. The linear example was selected due to its simplicity of computation and its nice convergence rate properties.

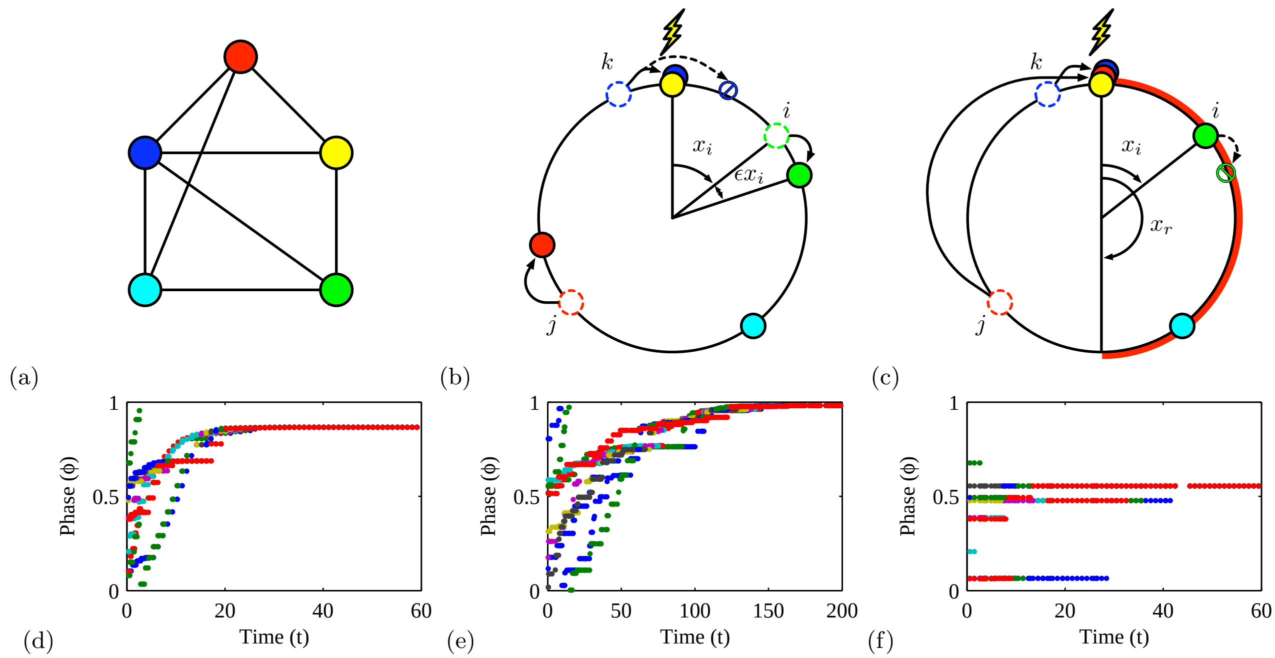

Figure 1: (a) 5-node topology (b) a corresponding diagram showing eqn. 1 with a firing (c) eqn. 2 (d-f) Example time-evolutions of x for 10 oscillators in a line topology. Each point represents the state of an oscillator relative to its initial starting point when a firing occurred. (d) $\langle \epsilon = 0.1,\ x_r = 0,\ p_f = 1\rangle$ (e) $\langle \epsilon = 0.1,\ x_r = 0,\ p_f = 0.2\rangle$ (f) $\langle \epsilon = 1,\ x_r = 0.5,\ p_f = 0.2\rangle$ Total number of messages required to achieve exact synchronization: (d) 633 (e) 1773 (f) 89

neighbor's firing, but is choosing not to act. Moreau also observed the same phenomenon with time-varying communication topologies; namely, more communication does not necessarily lead to faster convergence [10]. Determining when to avoid information and when to utilize it is still an open question. However, it is observed here that the setting of a refractory period to 0.5 breaks a particular symmetry that might be crucial to speeding convergence.

It has been widely assumed that oscillators must be weakly coupled to converge to synchrony. In fact, the proofs guaranteeing convergence require a finitely small coupling constant, ϵ, to argue a that a linear approximation holds for the nonlinear dynamical system described in equation (2) [7]. Increasing the coupling constant increases the rate to synchrony, but once a critical value, ϵ_0 (which depends on the network topology and the number of nodes), is reached, convergence is no longer guaranteed.[6] Since this algorithm is most desired for variable-sized, ad-hoc networks, selecting ϵ becomes very difficult to select. In practice, it is either numerically tuned or arbitrarily selected.

However, with the combination of randomness and a refractory period of 0.5, ϵ can be increased to 1 with no apparent harmful effects. Figure 3(c) depicts the results of increasing ϵ from 0.2 to 1. Values larger than 1 do not have any additional effect since $x_r = 0.5$; for $\epsilon \geq 1$, x_i is always reset to $x_i = 0$. The best performance is achieved at the strongest coupling, $\epsilon = 1$. Thus, a previously difficult to determine parameter has been reduced to a constant.

Thus, $\langle \epsilon = 1,\ x_r = 0.5,\ p_f = 0.2\rangle$ is the best out of the considered algorithms. Investigating the optimal p_f if message collisions are neglected is left to future work. Intuitively, oscillators running this algorithm have two options: (1) ignore a firing if $x_i < x_r$ (2) otherwise, reset x_i to 0. Figure 1(f) depicts a typical run for a 10-node line topology. Note that in the absence of clock drift, the only states at which nodes ever exist when a node is firing are a subset of the initial starting states, which implies that this algorithm may have additional uses as a general-purpose consensus algorithm. Comparing figures 2(b) and (c) shows that this algorithm has similar performance gains for other topologies. Additionally, figure 2(c) also confirms the intuition that line graphs are the worst-case for this algorithm. Note that there is slight tail for the ring topology near the 95th percentile. Once the topology size reaches hundreds of nodes the ring topology begins to become more difficult than the line.

5. CONCLUSION

This paper shows how the weakly pulse-coupled oscillator framework suffers from unsatisfactory scalability properties. This is exacerbated when randomness is introduced with $p_f < 1$. By combining randomness with a refractory period of 0.5, however, it is possible to strongly couple the oscillators, and drive the entire network to synchrony very quickly.

A recurring theme is how the selective reduction of transmitted information can not only save on energy, but also improve results for distributed consensus algorithms. If each node has access to a local clock, it can access x_i to send its information to neighboring nodes when $x_i = 1$. Receiving nodes then compare x_j to x_r to determine which information to accept and which to decline. Preliminary results

[6] While the results are shown for $x_r = 1$, it is conjectured that ϵ is still bounded by a small, but finite ϵ_0 with $x_r > 0$

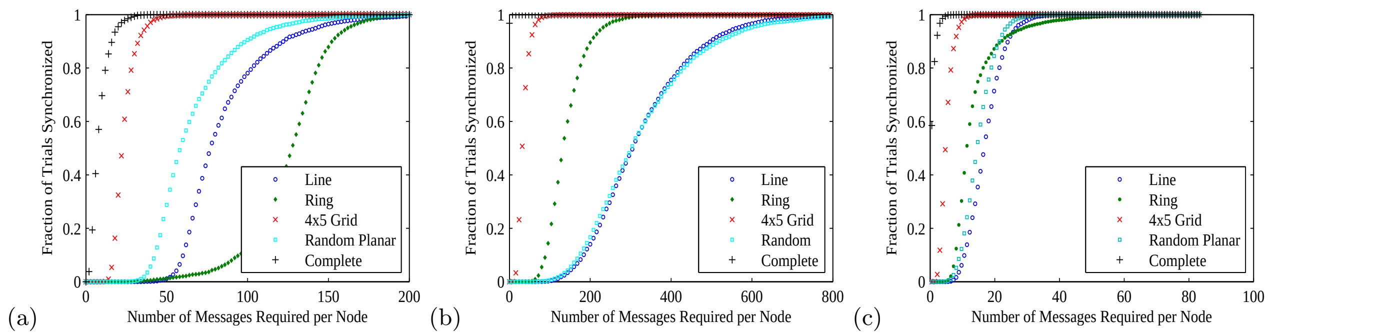

Figure 2: The cumulative distribution functions for the number of messages required per node (Note the axis scales). 5000 trials were run on the following 20-node topologies: line, ring, 4x5 grid, a random graph, and a complete graph. (a) $\langle \epsilon = 0.1,\ x_r = 0,\ p_f = 1\rangle$ **(b)** $\langle \epsilon = 0.1,\ x_r = 0,\ p_f = 0.2\rangle$ **(c)** $\langle \epsilon = 1,\ x_r = 0.5,\ p_f = 0.2\rangle$

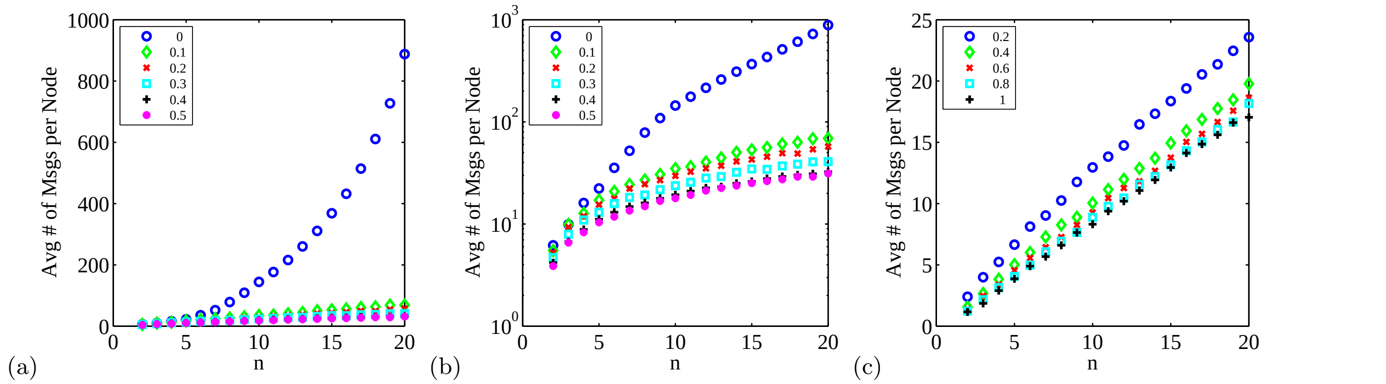

Figure 3: The average number of messages required per node. Each point is determined by 500 trials that were run on variable-sized line graphs. (a & b) $\langle \epsilon = 0.1,\ x_r = \{0, 0.1, 0.2, 0.3, 0.4, 0.5\},\ p_f = 0.2\rangle$ **on linear and log scales, respectively (c)** $\langle \epsilon = \{0.2, 0.4, 0.6, 0.8, 1\},\ x_r = 0.5,\ p_f = 0.2\rangle$

also indicate that if nodes keep track of where their current information was received from, a tree can be formed by running the strongly coupled algorithm. This suggests that this may be a distributed algorithm for directly computing a spanning tree of a network. Future work will investigate these preliminary results and possibly use them to construct a convergence proof for $\langle \epsilon = 1,\ x_r = 0.5,\ p_f \in [0, 1)\rangle$.

6. REFERENCES

[1] Green Hills Software. `http://www.ghs.com`.

[2] I. Chlamtac, C. Petrioli, and J. Redi. Energy-conserving access protocols for identification networks. *IEEE/ACM Transactions on Networking*, 7(1):51–59, 1999.

[3] L. Dai, P. Basu, and J. Redi. An accurate and energy efficient scheme for slot synchronization in wireless sensor networks. In *Proceedings of IEEE/CreateNet BROADNETS Conference*, 2006.

[4] E. M. Izhikevich. Weakly pulse-coupled oscillators, FM interactions, synchronization, and oscillatory associative memory. *IEEE Transactions on Neural Networks*, 10(3):508–26, 1999.

[5] Y. Kuramoto. Collective synchronization of pulse-coupled oscillators and excitable units. *Physica D*, 50:15–30, 1991.

[6] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–65, 1978.

[7] D. Lucarelli and I. Wang. Decentralized synchronization protocols with nearest neighbor communication. In *Proceedings of SenSys*, 2004.

[8] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of SenSys*, 2004.

[9] R. Mirollo and S. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal of Applied Math*, 50(6):1645–62, 1990.

[10] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005.

[11] J. Redi. *Energy-Conserving Protocols for Wireless Data Networks*. PhD thesis, Boston University, 1998.

[12] A. Scaglione and Y. Hong. Time synchronization and reach-back communications with pulse-coupled oscillators for UWB wireless ad hoc networks. In *the IEEE Conference on Ultra Wideband Systems and Technologies*, 2003.

[13] SiRF Technologies. SiRFstarIII GSC3e/LP product insert. `http://www.sirf.com`.

[14] U.S. Department of Defense. Global positioning system standard positioning service performance standard. `http://www.navcen.uscg.gov`, 2001.

[15] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly-inspired sensor network synchronicity with realistic radio effects. In *Proceedings of SenSys*, 2005.