Desynchronization with an Artificial Force Field for Wireless Networks

Supasate Choochaisri, Kittipat Apicharttrisorn, Kittiporn Korprasertthaworn, Pongpakdi Taechalertpaisarn, and Chalermek Intanagonwiwat [1]
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand
{supasate.c, kittipat.ap, kittiporn.k, pongpakdi.t}@student.chula.ac.th, chalermek.i@chula.ac.th

ABSTRACT

Desynchronization is useful for scheduling nodes to perform tasks at different time. This property is desirable for resource sharing, TDMA scheduling, and collision avoiding. Inspired by robotic circular formation, we propose *DWARF* (Desynchronization With an ARtificial Force field), a novel technique for desynchronization in wireless networks. Each neighboring node has artificial forces to repel other nodes to perform tasks at different time phases. Nodes with closer time phases have stronger forces to repel each other in the time domain. Each node adjusts its time phase proportionally to its received forces. Once the received forces are balanced, nodes are desynchronized. We evaluate our implementation of DWARF on TOSSIM, a simulator for wireless sensor networks. The simulation results indicate that DWARF incurs significantly lower desynchronization error and scales much better than existing approaches.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed Networks*

General Terms

Algorithms, Design, Performance

Keywords

desynchronization, self-organizing, sensor networks, wireless networks

1. INTRODUCTION

Networked, distributed systems usually cooperate to have a common notion of time in order to accomplish tasks with consistent results through time synchronization protocols (*e.g.*, TPSN[5], FTSP[9], GTSP[16], and EGTSP[1]). However, some systems simply require nodes to work at the same time (*e.g.*, Firefly Synchronicity [17]). In such systems, a global notion of time may not be necessary. Conversely, some systems require nodes *not* to work at the same time (*i.e.*, to desynchronize). Desynchronization organizes all accesses to a shared resource to be collision-free and even equitable. A concrete example is a system using a Time Division Multiple Access (TDMA) protocol. Nodes access the shared media only in their time slots to send messages with no collision. Other potential applications are techniques to increase a sampling rate in multiple analog-to-digital converters, to schedule resource in multi-core processors, and to control traffic at intersections.

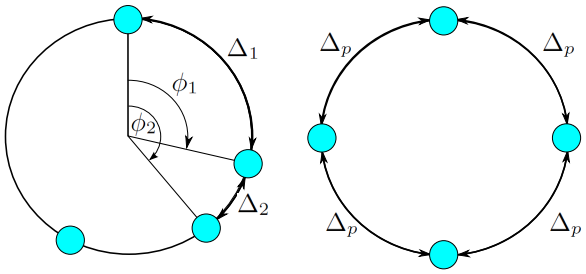


Figure 1: Desynchronization Framework. (a) Nodes are placed and labeled on the time circle. Node i has the time phase ϕ_i and the phase difference Δ_i with its previous phase neighbor. Δ_{i+1} is the phase difference between node i and its next phase neighbor. (b) The positions of nodes in the perfect desynchrony state.

DESYNC [4] proposes a simple desynchronization framework as depicted in Figure 1. The perimeter of a time circle represents a configurable firing period T of nodes' oscillators. The time position or phase of each node represents its turn to access a shared resource. The system is desynchronized when all nodes are equally separated in the time circle. Desynchronization can be categorized into two groups:

1. Weak desynchronization (*e.g.*, INVERSE-MS [13]) distributes nodes equally in the time circle but distorts the time period (*i.e.*, the period is extended). Therefore, it is not suitable for systems that require an exact time period to access resources.

2. Strong desynchronization (*e.g.*, DESYNC [4], M-DESYNC [8]) is more desirable than weak desynchronization because of no period distortion.

Similar to DESYNC, we focus only on strong desynchronization due to their mentioned desirable property. The DESYNC algorithm is simple and effective. The phase of each node is an average phase of its two phase neighbors on the time circle. However, DESYNC's error is quite high even after convergence because a phase error of a node can

[1]Corresponding Author

propagate to its phase neighbors and indefinitely circulate in the network (see Section 6).

In this paper, we present *DWARF* (Desynchronization With an ARtificial Force field), a novel desychronization protocol that incurs low desynchronization error even in a highly dense network. Our work is inspired by the robot circular formation with an electromagnetic-like force field [2]. In that work, robots of the same type have artificial forces to repel each other whereas robots of the different types have attracting forces. For a certain coefficient of forces, a circle of heterogeneous robots can be formed (see Figure 2). Similarly, if we think of nodes on a time circle as the robots of the same type, the nodes will repel each other and keep time intervals from their neighbors as far as possible. Once all received forces are balanced, nodes are equally spread out in the time circle (*i.e.*, desynchronized).

DWARF have the following contributions:

- DWARF is a distributed desynchronization algorithm using time phases of all neighbors to achieve the desyncrhony state.
- DWARF does not require time synchronization, does not assume already slotted time, and does not incur any control message overhead.
- DWARF is simple due to low complexity in terms of computation and memory. Therefore, it is suitable for resource-constraint networks, such as wireless sensor networks. Additionally, message complexity is low because the algorithm relies on the timing of the message, not information inside the message.
- We have implemented and evaluated DWARF on TOSSIM, a simulator for wireless sensor networks. Our results indicate that DWARF scales well with network size and outperforms DESYNC significantly by achieving 10 - 63% reduction in desynchronization error.

The rest of the paper is organized as follows. Section 2 covers the related works about desynchronization in wireless networks. In Section 3, we overview the robot circular formation that motivates our work. We describe our desynchronization protocol in Section 4. Section 5 analyses the convergence property of the protocol. Then, we evaluate the performance in Section 6. Finally, Section 7 concludes the paper and discusses limitations as well as future works.

2. RELATED WORK

To the best of our knowledge, DESYNC [4] is the first to introduce the desynchronization problem. In DESYNC, a node simply attempts to stay in the middle between its previous and next phase neighbors. By repeating this simple algorithm, all nodes will eventually be spread out. However, the error is also propagated to the phase neighbors and indefinitely circulated inside a network. Therefore, the DESYNC's error is quite high even after convergence. Using phases of all neighbors, DWARF can achieve significantly lower desynchronization error than that of DESYNC.

Designed to converge faster than DESYNC, INVERSE-MS [13] is an inverse algorithm of the synchronicity work by Morollo and Strogatz [10]. At a steady state, INVERSE-MS maintains a dynamic equilibrium (*i.e.*, nodes keep changing time phases while maintaining desynchronization). However, the time period is distorted (*i.e.*, weak desynchronizaiton).

M-DESYNC [8] proposes a localized multi-hop desynchronization protocol that works on a granularity of time slots. M-DESYNC is strong desynchronization protocol that is designed to work on a multi-hop network. The protocol estimates the required number of time slots with a two-hop maximum degree. This estimation helps M-DESYNC converge very fast. However, M-DESYNC requires that all nodes have a global notion of time in order to share the common perception of time slots. Additionally, M-DESYNC also requires additional message overhead for claiming the slots. Conversely, DWARF does not require a global notion of time or additional message overhead. Neither does DESYNC.

Motskin et al. [11] proposes a simple and lightweight strong desynchronization algorithm that is based on a graph coloring model. Lightweight coloring algorithm works on general graph networks and does not need global time. To ensure that the selected time slot does not overlap with others', a node needs to listen to the media for a full time period before claiming the slot. Without a common notion of time, the starting time of each slot is quite random. As a result, several time gaps are too small to be used as time slots. This external fragmentation problem reduces resource utilization of the system. Additionally, to converge faster, their algorithm overestimates the number of time slots. Hence, several large time gaps are also left unused and the resource utilization is low. In contrast, DESYNC and DWARF fully utilize resources even without a global notion of time.

Recently, Degesys and Nagpal [3] extend DESYNC to support multi-hop networks. We also plan to similarly extend DWARF for multi-hop networks in our future work as well.

Other works that are similar to desynchronization protocols are distributed TDMA protocols. Most of distributed TDMA protocols (e.g., TRAMA [14], Parthasarathy [12], ED-TDMA [6], and Herman [7]) assume time is already slotted or all nodes are synchronized to achieve the same global clock. Some distributed TDMA protocols do not require time synchronization. However, they require more states and incur control message overhead. For example, DRAND [15] requires the control overhead for sending request, reject, release, and grant messages. In our work, again, we do not require time synchronization, do not assume already slotted time, and do not incur any control message overhead.

3. MOTIVATION

We observe that the desynchronization framework is similar to the circular formation of heterogeneous robots [2]. In robotics, some tasks are too difficult for a single robot to accomplish by itself. Some even require multiple robots with different types. For such tasks, heterogeneous robots are distributedly grouped into teams that are equally spread out to cover the monitored area. Each robot has no global knowledge of others' absolute positions but can detect relative positions of the others with respect to itself as well as the type of the others. To form a circle, an artificial force is used as an abstraction for velocity adaptation of a robot. Robots of different types have attracting forces to each other. Conversely, robots of the same type have repelling forces. As a result, the circle of heterogeneous robots will be formed and robots are nicely spaced on the circle (see Figure 2).

In our work, a node (like a robot) does not have a global

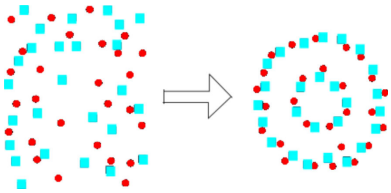


Figure 2: Results of Robotic Circular Formation in [2]. Robots with two different types form the circle.

notion of time but each node can measure relative time differences with other nodes. The circle of robots is similar to our circle of time period. The distribution of robots can be mapped to the distribution of nodes on the time circle. The work inspires us to design a novel desynchronization protocol for wireless networks based on an artificial force field. In this paper, the circle is temporal rather than spatial and only repelling forces are required to space nodes on the time circle.

4. DESYNCHRONIZATION PROTOCOL

We first describe the concept of an artificial force field in Section 4.1 and explain our algorithm in Section 4.2.

Like DESYNC [4] and INVERSE-MS [13], we assume a one-hop network in this paper. Similar to DESYNC, our protocol can also be extended for multi-hop networks. We discuss this assumption in Section 7.

4.1 Artificial Force Field

An artificial force field is an analogy to the circle of a time period. Nodes are in the same force field if they can communicate with each other.

If node i and node j are on the same force field, they have repelling forces to push one another away. A closer pair of nodes has a higher magnitude of force than a farther pair does. The time interval between two nodes is derived from the phase difference between them. If two nodes have a small phase difference, they have a high magnitude of force and vice versa. In other word, a repelling force is an inverse of a phase difference between two nodes:

$$f_{ij} = \frac{1}{\Delta\phi_{ij}/T}, \Delta\phi_{ij} \in (-\frac{T}{2}, \frac{T}{2}), \quad (1)$$

where f_{ij} is the repelling force from node j to node i on a time period T and $\Delta\phi_{ij}$ is the phase difference between node i and j. We note that $\Delta\phi_{ij}$ is not equal to 0 because if two nodes fire at the same time, their firings collide and two nodes do not record other's firing. Additionally, at $\frac{T}{2}$ or $-\frac{T}{2}$, a node does not repel an opposite node because they are balanced.

A repelling force can be positive (clockwise repelling) or negative (counterclockwise repelling). A positive force is created by a node on the left half of the circle relative to the considered node whereas a negative force is created by a node on the right half. Figure 3 represents a field of repelling forces on node 1.

Each node in the force field moves to a new time position or phase proportional to the total received force. Given n nodes in a force field, the total force on a node i is the

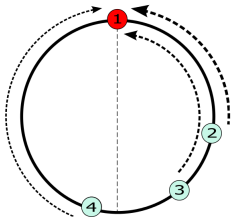


Figure 3: Artificial Force Field. Arrow lines represent repelling forces from node 2, 3, and 4 to node 1. A shorter and thicker line is a stronger force. A force from node 4 is a positive force and two forces from node 2 and 3 are negative forces.

following:

$$\mathcal{F}_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} f_{ij}. \quad (2)$$

Eventually, nodes reach an equilibrium state whereby the total force of the system is close to zero and each pair of phase neighboring nodes has the same time interval. This equilibrium state also indicates the desynchrony state because all nodes are equally spaced on the time circle.

4.2 Algorithm

We assume that, initially, nodes are not desynchronized. Each node sets a timer to fire in T time unit. After setting the timer, each node listens to all neighbors until its timer expires.

When receiving a firing message from its neighbor, the (positive or negative) repelling force from that neighbor is calculated based on the phase difference. When the timer expires, a node broadcasts a firing message to neighbors. Then, the node calculates a new time phase to move on the circle based on the summation of forces from all neighbors and sets a new timer according to the new time phase.

Reasonably, one may wonder how far a node should move or adjust its phase. In our work, given the total received force $\mathcal{F}_i$, the node i adjusts to a new time phase $\phi_i^{'}$,

$$\phi_i^{'} = (\phi_i + K\mathcal{F}_i) \mod T, \quad (3)$$

where ϕ_i is the current phase of the node i.

Undoubtedly, the proper value of the coefficient K leads to the proper new phase. The value of K is similar to a step size which is used in artificial intelligence techniques. Therefore, if the value of K is too small, the system takes much time to converge. On the other hand, if the value of K is too large, the system may overshoot the optimal value and does not converge. We observe that, given the same time period, fewer nodes in the system result in bigger phase difference between two phase neighbors. To be desynchronized, nodes in sparse networks must make a bigger adjustment to their time phases than nodes in dense networks must. Therefore, the same total received force should have a bigger impact on a node in sparse networks than on a node in dense networks. To reflect this observation, the coefficient K is inversely proportional to a power of a number of nodes n,

$$K = c_1 \times n^{-c_2}, c_1, c_2 \geq 0. \quad (4)$$

Therefore, we have conducted an experiment to find the

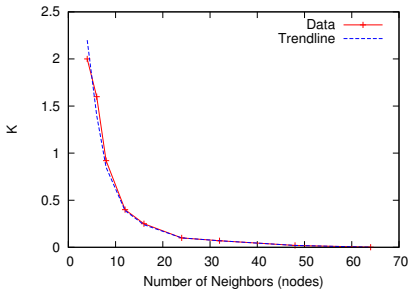


Figure 4: Relation of the coefficient K with a number of nodes n

proper value of c_1 and c_2. We set the time period T to 1000. Additionally, we have varied a number of nodes and have searched for the K values that led to low errors (see Figure 4). Using power regression on the result, we have deduced a relation of K and n (the trendline). We have also found the proper value of c_1 and c_2 as follows:

$$K = 38.597 \times n^{-1.874}.$$

However, this K value is derived by setting T equal to 1000. Therefore, for arbitrary T,

$$K = 38.597 \times n^{-1.874} \times \frac{T}{1000}. \quad (5)$$

We explain other details of the experiment and proof of Equation 5 in Appendix A.

All nodes in the artificial force field (in the period circle) iteratively run the same algorithm until the force is balanced (*i.e.*, all nodes are in the desynchrony state). The pseudocode of this algorithm is shown in Appendix B.

5. STABILITY ANALYSIS

To mathematically analyze the stability of the algorithm (*i.e.*, the algorithm converges to the steady desynchronized state) is rather complicated. The stability analysis of standard dynamical systems does not suffice because the force function is non-linear and the transformation matrix to find the eigenvalues cannot be formed. Therefore, we have tested the stability by simulation. In each simulation scenario, nodes randomly wake up. Our simulation result shows that the system always converge except when there are too many nodes within a short period (*i.e.*, the time gap between each firing may be shorter than the message delay). In other words, the system is over-saturated. The result is shown in Section 6.3. We believe that the non-linear dynamic systems analysis based on the Lyapunov stability theory could prove our conjecture.

However, one of the reasons that the system converges (nodes are nicely spread) is because our objective function (*i.e.*, the summation of received forces at all nodes) is convex. Therefore, it contains only one global minima and no local minima. The proof of convexity is shown in Appendix D. Our algorithm attempts to reduce the value of the objective function overtime and eventually reach a value near the global minima. However, the system does not always converge. To converge, the system must meet two criteria. First, the value of K must be proper (see Section 4.2). Second, a number of nodes within a time period must not be too high. Otherwise, the system may be over-saturated.

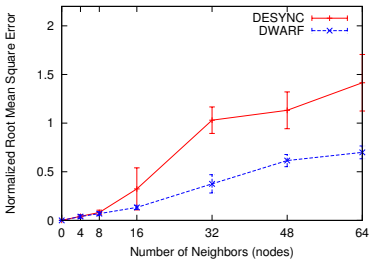


Figure 5: Root mean square error normalized by perfect phase difference after 300 time periods

6. EVALUATION

In this section, we evaluate the performance of our proposed mechanism and compare with DESYNC [4] because DESYNC and our mechanism shares the same goal and requirement. Particularly, they do not require time synchronization, do not assume already slotted time, do not need to look into the packet content, and do not incur control packets but still achieves equivalent time spaces. Other protocols (e.g., M-DESYNC [8], Lightweight [11]) assume different requirements. Therefore, we only discuss our differences with such protocols in Section 2.

The performance metrics in this evaluation are desynchronization error and convergence time. The former indicates how close the current state is to the perfect desynchrony state. The latter indicates how fast the algorithm converges.

6.1 Evaluation Environment

We implement DWARF on TinyOS, an operating system for wireless sensor networks and evaluate the protocol on TOSSIM, a TinyOS simulator. We vary the one-hop network size from 4 to 64 nodes. Each node periodically fires a message that contains only application data with no extra control overhead. This zero overhead is the advantage of both DWARF and DESYNC because, to avoid collisions, a node only needs to know the timing of the firing rather than the control information inside a packet. In our simulation, for both DWARF and DESYNC, we use a 2-byte node ID and a 2-byte counter as the data. However, we do use the regular 11-byte CC2420 header for TOSSIM. Therefore, we do not measure the overhead in our evaluation. We set the time period to 500 milliseconds and compare our result with that of DESYNC. The step size (α) of DESYNC is set to 0.95 (the same value used in [4]). Initially, the phase of each node is random in a range of 0 to 500.

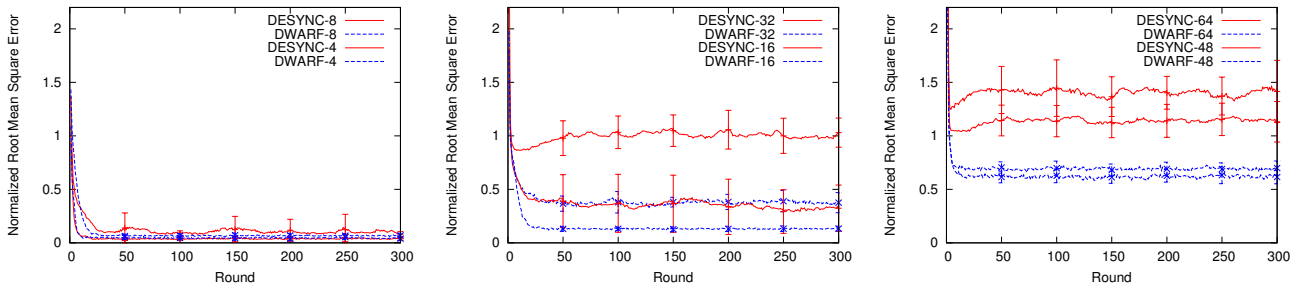
6.2 Desynchronization Error

To measure the desynchronization error, we run the simulation for 300 time periods. In each network size, we run the simulation for 30 times. Then, we measure the average root mean square error (RMSE). The error (ERR) is the measured phase difference minus the perfect phase difference:

$$ERR_i = \Delta\phi_{ij} - T/n,$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} ERR_i^2}{n}},$$

where node j is the next phase neighbor of node i. $\Delta\phi_{ij}$ is the phase difference between node i and node j on the time



(a) Sparse (b) Dense (c) Extremely Dense

Figure 6: Convergence time and root mean square error normalized by expected phase difference

period T. Given that n is a total number of nodes, T/n is the perfect phase difference.

However, a smaller absolute error in a dense network is not necessarily better than a bigger error in a sparse network because the perfect phase difference in a dense network is also smaller than that in a sparse network. Thus, for a comparable view of each network size, we also measure a normalized root mean square error (NRMSE) that is a ratio of the root mean square error and the perfect phase difference of each network size. Figure 5 illustrates the result of the normalized desynchronization error in each network size after 300 time periods. (see Figure C.1 in Appendix C for the absolute root mean square errors).

The result indicates that, in all network sizes (4 - 64 nodes), DWARF achieves significantly better desynchrony states than DESYNC does. Understandably, using information from all neighbors (as in DWARF) leads to lower errors than using information from only two phase neighbors (as in DESYNC) does. Furthermore, DESYNC's mechanism allows a phase error of a node to propagate to its phase neighbors. A part of this error will propagate back and forth between two phase neighbors as well as circulate inside the network. As a result, DESYNC's error after convergence is still quite large. In contrast, DWARF is robust to this error propagation. Even though the error propagation may still occur, the impact is not significant. Given that DWARF uses the sum of forces from all neighbors, an error from one neighbor does not overwhelm the system.

We note that we show RMSE and NRMSE after 300 periods because, by that time, the errors of all simulation scenarios seem to be stable. The actual convergence time in most scenarios is much lower than 300 rounds (see the next section).

6.3 Convergence Time

In desynchronization error evaluation, we only measure the performance after 300 time periods. However, the previous result does not indicate whether the protocols have converged or not. Neither does it indicate how fast the protocols have converged. Hence, we also measure the absolute root mean square error and normalized root mean square error for each time period (absolute root mean square errors are shown in Appendix C).

In sparse networks (Figure 6a), both protocols converge with similar small error but DWARF converges slightly faster. In dense networks (Figure 6b) and extremely dense networks (Figure 6c), DESYNC converges but errors highly fluctuate whereas DWARF converges faster and errors are more stable. Furthermore, in dense and extremely dense networks, the normalized root mean square errors of DESYNC after convergence is higher or equal to 1. This means that the error is very large compared to the perfect phase difference. Conversely, the normalized error of DWARF is lower than 1 even in the extremely dense networks. Therefore, DWARF scales well with the network density whereas DESYNC does not.

In a denser network, the errors of DESYNC and DWARF are higher because the probability of message collisions increases (see the next section).

6.4 Correlation of Packet Loss and Desynchronization Error

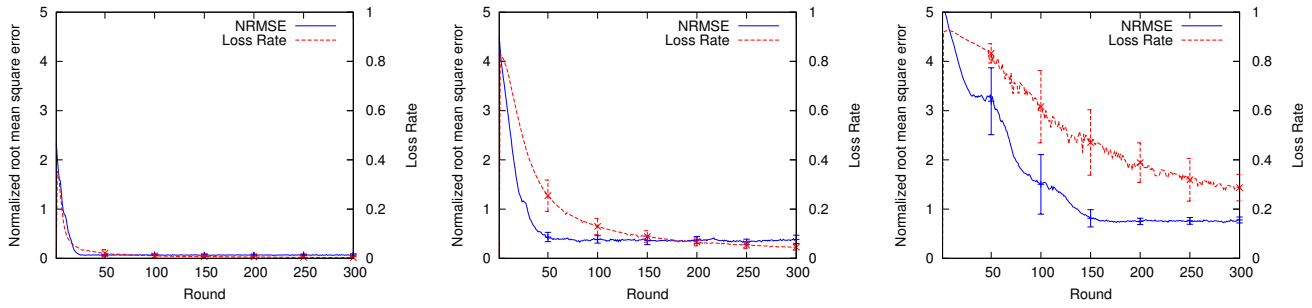
In this section, we investigate the correlation of packet loss and desynchronization error by setting all nodes to wake up simultaneously. Due to space limitation, we only include the results of DWARF in networks of 8, 32, and 64 nodes for 300 time periods (Figure 7). However, in networks of 4, 16, and 48 nodes, the results are similar (not shown).

At the beginning, the network is far from the desynchrony state and messages from different nodes are simultaneously fired. This results in lost packets and errors. However, over time, nodes gradually adapt their time phases. Consequently, a number of lost packets and the error also gradually drop.

7. CONCLUSION AND DISCUSSION

In this paper, we present DWARF, a novel strong desynchronization protocol that enables nodes in a system to perform tasks at different time. To the best of our knowledge, DWARF is the first desynchronization protocol that is based on the concept of electromagnetic fields, a foundation of physics.

Our protocol is completely distributed and localized with no reliance on a centralized node or a global notion of time. Due to low complexity in terms of computation, memory, and message overhead, DWARF is suitable for traditional wireless networks as well as resource-constraint wireless sensor networks. Our preliminary evaluation indicates that DWARF can significantly outperform DESYNC by reducing 10 - 63% of the desynchronization error. In addition, DWARF scales well with network size given that the normalized error is lower than 1 even in extremely dense networks.



(a) 8 nodes (b) 32 nodes (c) 64 nodes

Figure 7: Correlation of packet loss and desynchronization error

Given that this work is still in an early stage, there is room to improve and to extend the idea before we can realize the full potential of DWARF. Recently, Degesys [3] has extended the DESYNC concept to support multi-hop topologies. The extension is also applicable to DWARF. Therefore, we believe that DWARF can be similarly extended for multi-hop topologies as well. In addition, the force function of DWARF is not limited to the sum of rational functions. Other different force functions (*e.g.* squared, cubed) are possible and viable for further investigation. We plan to evaluate DWARF on real wireless devices and to explore other extensions in our future work.

8. ACKNOWLEDGEMENT

We would like to acknowledge and thank three anonymous reviewers for their valuable suggestions to revise and improve this work.

9. REFERENCES

[1] Apicharttrisorn, K., Choochaisri, S., and Intanagonwiwat, C. Energy-Efficient Gradient Time Synchronization for Wireless Sensor Networks. In *Second International Conference on Computational Intelligence, Communication Systems and Networks*, 2010.

[2] Boonpinon, N., and Sudsang, A. Heterogeneity Driven Circular Formation. In *IEEE International Conference on Robotics and Biomimetics*, 2006.

[3] Degesys, J., and Nagpal, R. Towards Desynchronization of Multi-hop Topologies. In *IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 2008.

[4] Degesys, J., Rose, I., Patel, A., and Nagpal, R. DESYNC: Self-Organizing Desynchronization and TDMA on Wireless Sensor Networks. In *IPSN*, 2007.

[5] Ganeriwal, S., Kumar, R., and Srivastava, M. B. Timing-sync protocol for sensor networks. In *SenSys*, 2003.

[6] Gong, H., Liu, M., Chen, G., and Zhang, X. A study on event-driven tdma protocol for wireless sensor networks. *EURASIP J. Wirel. Commun. Netw. 2010* (February 2010), 9:1–9:12.

[7] Herman, T., and Tixeuil, S. A distributed tdma slot assignment algorithm for wireless sensor networks. In *Algorithmic Aspects of Wireless Sensor Networks*, S. Nikoletseas and J. Rolim, Eds., vol. 3121 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 45–58.

[8] Kang, H., and Wong, J. A localized multi-hop desynchronization algorithm for wireless sensor networks. In *INFOCOM*, 2009.

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

[10] Mirollo, R. E., and Strogatz, S. H. Synchronization of pulse-coupled biological oscillators. *SIAM J. Appl. Math. 50* (November 1990), 1645–1662.

[11] Motskin, A., Roughgarden, T., Skraba, P., and Guibas, L. Lightweight Coloring and Desynchronization for Networks. In *INFOCOM*, 2009.

[12] Parthasarathy, S., and Gandhi, R. Distributed algorithms for coloring and domination in wireless adhoc networks. In *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science*, K. Lodaya and M. Mahajan, Eds., vol. 3328 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 447–459.

[13] Patel, A., Degesys, J., and Nagpal, R. Desynchronization: The Theory of Self-Organizing Algorithms for Round-Robin Scheduling. In *International Conference on Self-Adaptive and Self-Organizing Systems*, 2007.

[14] Rajendran, V., Obraczka, K., and Garcia-Luna-Aceves, J. J. Energy-efficient, collision-free medium access control for wireless sensor networks. *Wirel. Netw. 12* (February 2006), 63–78.

[15] Rhee, I., Warrier, A., Min, J., and Xu, L. Drand: Distributed randomized tdma scheduling for wireless ad hoc networks. *IEEE Transactions on Mobile Computing, 8*, 10 (oct. 2009), 1384 –1396.

[16] Sommer, P., and Wattenhofer, R. Gradient clock synchronization in wireless sensor networks. In *IPSN*, 2009.

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

APPENDIX

A. FINDING K-N RELATION

In this section, we explain in details how we find the proper values of c_1 and c_2 for the relation between K and n values.

We set a time period T to 1000 and vary a number of nodes. In a specific number of node, we first simulate to see the trend of the value K that leads to small errors. Then, we select a range of good K values. After that, we simulate 100 times to obtain the average desynchronization error for each K value. In each simulation, we randomly set an initial phase of each node between 0 and T (period value). Finally, we select the K value that results in the lowest error. After getting the proper K value for each number of node, we plot the relation between K and a number of nodes (Figure 4) and use a mathematic tool to calculate the power regression. The obtained relation function between K and n (the trendline in Figure 4) consists of c_1 and c_2 values as follows:

$$K = 38.597 \times n^{-1.874}.$$

However, this K value is derived by setting T equal to 1000. Therefore, for arbitrary T,

$$K = 38.597 \times n^{-1.874} \times \frac{T}{1000}.$$

PROOF. From Equation 2 and 3, the phase of node j is adjusted by $K\mathcal{F}_j = K\sum_{i\neq j}^{n} f_{i,j} = \sum_{i\neq j}^{n} Kf_{i,j}$. Therefore, we can analyze the value of K from only single force $f_{i,j}$.

For a time circle of a period T_p, let θ_p be an angle between two nodes on the circle and Θ_p be an angle between the new adjusted phase and the old phase based on a single force where $\theta_p, \Theta_p \in (0, 2\pi)$. Hence,

$$\frac{\theta_p}{2\pi} = \frac{\Delta\phi_{i,j}}{T_p}, \tag{6}$$

and

$$\frac{\Theta_p}{2\pi} = \frac{Kf_{i,j}}{T_p}. \tag{7}$$

If θ_1 of T_1 equals to θ_2 of T_2, both of them should be adjusted with the same angle amount $\Theta_1 = \Theta_2$. Thus, from Equation 7,

$$\Theta_1 = \Theta_2$$
$$\frac{K_1 f_{i,j(1)}}{T_1} = \frac{K_2 f_{i,j(2)}}{T_2}.$$

From Equation 1 and 6, $f_{i,j} = \frac{1}{\Delta\phi_{i,j}/T_p} = \frac{2\pi}{\theta_p}$, consequently,

$$\frac{K_1 2\pi}{T_1\theta_1} = \frac{K_2 2\pi}{T_2\theta_2}$$
$$\frac{K_1}{T_1} = \frac{K_2}{T_2}$$
$$K_2 = K_1\frac{T_2}{T_1}. \tag{8}$$

At $T = 1000$, we get $K = 38.597 \times n^{-1.874}$. Therefore, from Equation 8, for arbitrary T,

$$K = 38.597 \times n^{-1.874} \times \frac{T}{1000}. \quad \square$$

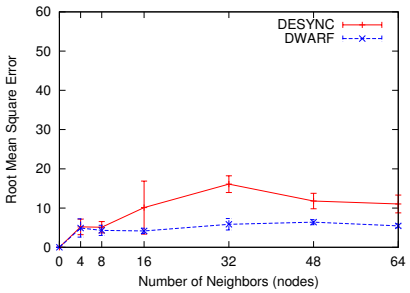


Figure C.1: Absolute root mean square error after 300 time periods

B. PSEUDOCODE

Initialization
$T = TimePeriod$ {Configurable Time Period}
$n = 1$ {Number of receiving messages including itself}
$\mathcal{F} = 0$ {Force Summation}
$lastFiringTime = localTime$
$currentPhase = localTime$ modulo T
Set a firing timer to be T unit time

Upon timer firing
Broadcast a firing message to neighbors
$lastFiringTime = localTime$
$currentPhase = localTime$ modulo T
$K = 38.597 \times n^{-1.874} \times \frac{T}{1000}$
$newPhase = currentPhase + (K \times \mathcal{F})$
if $newPhase < 0$ **then**
  $newPhase = T + newPhase$
end if
Set a firing timer to be fired at $newPhase$
$\mathcal{F} = 0$
$n = 1$

Upon receiving a firing message
$n = n + 1$
$phaseDiff = localTime - lastFiringTime$
if $phaseDiff == 0.5T$ **then**
  $\mathcal{F} = \mathcal{F} + 0$ {Balanced force}
else if $phaseDiff < 0.5T$ **then**
  $\mathcal{F} = \mathcal{F} + |\frac{1}{phaseDiff/T}|$ {Positive force}
else
  $\mathcal{F} = \mathcal{F} - |\frac{1}{(T-phaseDiff)/T}|$ {Negative force}
end if

C. ABSOLUTE ROOT MEAN SQUARE ERROR

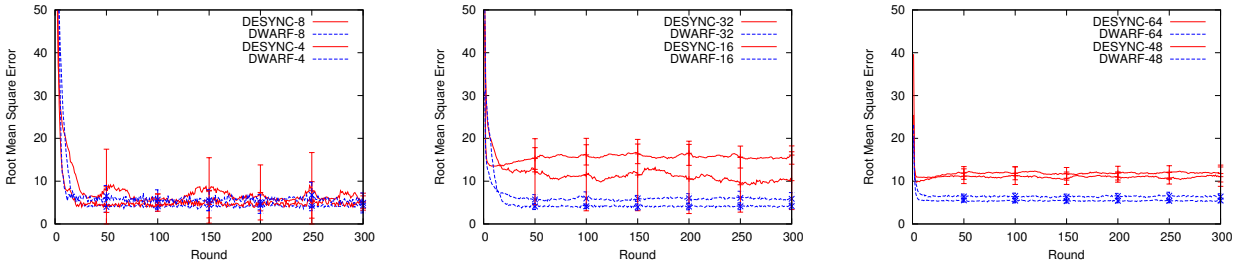
In this section, we show the simulation results of average root mean square errors without normalization.

Figure C.1 indicates the absolute root mean square errors after 300 rounds that we evaluate in Section 6.2.

Figure C.2 indicates the convergence time and the absolute root mean square errors that we evaluate in Section 6.3.

D. PROOF OF CONVEXITY

THEOREM 1. *The system force summation function has*



(a) Sparse (b) Dense (c) Extremely Dense

Figure C.2: Convergence time and absolute root mean square error

one global minima and no local minima.

To prove Theorem 1, let F_i be the force summation at node i and $\Delta_{i,j} \in (-\frac{T}{2}, \frac{T}{2})$ is an interval between node i and j. If $\Delta_{i,j} > 0$, the node j repels the node i in a positive direction. In contrast, if $\Delta_{i,j} < 0$, the node j repels the node i in a negative direction.

Therefore, for n nodes, F_i can be formulated as the following equation,

$$F_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\Delta_{i,j}}.$$

The objective function E of the system is the summation of absolute received forces at all nodes,

$$E = \sum_{i=1}^{n} |F_i| = \sum_{i=1}^{n} \left| \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\Delta_{i,j}} \right| \tag{9}$$

To prove the function E is a convex function, we must prove that two following conditions are satisfied; 1) the set of all $\Delta_{i,j}$ is a convex set and 2) the Hessian matrix of E is positive semidefinite.

PROPOSITION 1. *A set of all possible interval $\Delta_{i,j}$ is a convex set.*

PROOF. A set of $(-\frac{T}{2}, \frac{T}{2})$ is a line connecting between $-\frac{T}{2}$ and $\frac{T}{2}$. Therefore, any Δ_{i_1,j_1}, $\Delta_{i_2,j_2} \in (-\frac{T}{2}, \frac{T}{2})$ and $\alpha \in \mathbb{R}$ with $0 \leq \alpha \leq 1$,

$$\alpha\Delta_{i_1,j_1} + (1-\alpha)\Delta_{i_2,j_2} \in (-\frac{T}{2}, \frac{T}{2}). \quad \square$$

PROPOSITION 2. *The Hessian of the function E is positive semidefinite.*

PROOF. Let H be the Hessian matrix of the objective function E,

$$H = \begin{pmatrix} \frac{\partial^2 E}{\partial \Delta_{1,2}^2} & \cdots & \frac{\partial^2 E}{\partial \Delta_{1,2} \partial \Delta_{n,n-1}} \\ \frac{\partial^2 E}{\partial \Delta_{1,3} \partial \Delta_{1,2}} & \cdots & \frac{\partial^2 E}{\partial \Delta_{1,3} \partial \Delta_{n,n-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial \Delta_{n,n-1} \partial \Delta_{1,2}} & \cdots & \frac{\partial^2 E}{\partial \Delta_{n,n-1}^2} \end{pmatrix}.$$

For any $\Delta_{x,y}$, we derive the first-order derivative as follows,

$$\frac{\partial E}{\partial \Delta_{x,y}} = \frac{\sum_{\substack{j=1 \\ j \neq x}}^{n} \frac{1}{\Delta_{x,j}}}{\left| \sum_{\substack{j=1 \\ j \neq x}}^{n} \frac{1}{\Delta_{x,j}} \right|} \left(-\frac{1}{\Delta_{x,y}^2} \right).$$

For the second-order partial derivatives, if we differentiate with the same $\Delta_{x,y}$,

$$\frac{\partial^2 E}{\partial \Delta_{x,y}^2} = \frac{\sum_{\substack{j=1 \\ j \neq x}}^{n} \frac{1}{\Delta_{x,j}}}{\left| \sum_{\substack{j=1 \\ j \neq x}}^{n} \frac{1}{\Delta_{x,j}} \right|} \left(\frac{2}{\Delta_{x,y}^3} \right).$$

In the other hand, if we differentiate with other $\Delta_{u,v}$, where $u \neq x$ or $v \neq y$, $\frac{\partial^2}{\partial \Delta_{u,v} \Delta_{x,y}} E = 0$.

Therefore, the Hessian matrix of the function E is

$$H = \begin{pmatrix} a_{1,2} & 0 & \cdots & 0 \\ 0 & a_{1,3} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n-1} \end{pmatrix},$$

where $a_{i,j} = \frac{u_i}{|u_i|} \left(\frac{2}{\Delta_{i,j}^3} \right)$ and $u_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\Delta_{i,j}}$.

To show that the Hessian matrix of the function E is positive semidefinite, we show that $\vec{\Delta}^T H \vec{\Delta} \geq 0$ for all $\vec{\Delta}$ when $\vec{\Delta} \neq 0$.

$$\begin{aligned} \vec{\Delta}^T H \vec{\Delta} &= \begin{pmatrix} \Delta_{1,2} & \Delta_{1,3} & \cdots & \Delta_{n,n-1} \end{pmatrix} H \begin{pmatrix} \Delta_{1,2} \\ \Delta_{1,3} \\ \vdots \\ \Delta_{n,n-1} \end{pmatrix} \\ &= \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{u_i}{|u_i|} \left(\frac{2}{\Delta_{i,j}} \right) \\ &= 2 \sum_{i=1}^{n} \frac{u_i}{|u_i|} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\Delta_{i,j}} \\ &= 2 \sum_{i=1}^{n} \frac{u_i}{|u_i|} u_i = 2 \sum_{i=1}^{n} |u_i| \\ &\geq 0 \end{aligned}$$

Therefore, we can conclude that $\vec{\Delta}^T H \vec{\Delta} \geq 0$ and the Hessian matrix of E is positive semidefinite. $\square$

From Proposition 1 and 2, we derive the following lemma.

LEMMA 1. *the function E is a convex function.*

From Lemma 1, we consequently prove Theorem 1 that the system force summation function has one global minima and no local minima. $\square$