DeVise: a Tool for Visualizing and Validating Desynchronization Protocols for Multi-hop Wireless Sensor Networks

Supasate Choochaisri[1], Siam Aurburananont, Akekanat Saowwapak-adisak, and Chalermek Intanagonwiwat[2]
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand
{supasate.c,siam.a, akekanat.s}@student.chula.ac.th, chalermek.i@chula.ac.th

***Abstract*—In recent years, there are several studies on desynchronization for wireless sensor networks. In order to propose new desynchronization protocols, unavoidably, researchers have to implement previous desynchronization works to compare with their own protocols. However, validating the implementations takes time and effort. In this paper, we propose DeVise, a visualizer for validating desynchronization protocols on multi-hop wireless sensor networks. DeVise reads configuration and trace files, and visualizes the protocol behavior. DeVise provides sufficient and useful information to help researchers reduce time and effort in the protocol validation process. We have evaluated DeVise with several desynchronization protocols. The result shows that DeVise helps researchers validate the implementations and effectively helps explore advantages and drawbacks when comparing different desynchronization protocols.**

I. Introduction

In recent years, there are several studies on desynchronization for wireless sensor networks. Desynchronization is a process to organize wireless sensor nodes *not* to work at the same time without a common notion of time (*i.e.* no time synchronization). Figure 1 illustrates a desynchronization framework which is used in [1]–[4]. The circle represents a time period. A node i fires a message at a *phase* ϕ_i on the time circle. The firing time difference between node $i-1$ and i denotes as Δ_i and the firing time difference between node i and $i+1$ denotes as Δ_{i+1}. Desynchronization is perfect if, in a wireless collision domain, all nodes are Δ_p time-separated equivalently.

Typically, wireless sensor nodes desynchronize themselves distributedly (*i.e.* self-organizing, no global information, and no centralized control). Therefore, several distributed protocols and applications directly gain benefit from desynchronization. For example, desynchronization can be used as an initialization protocol to avoid packet collision for other protocols that rely on exchanging messages at start-up time such as neighbors information. In Time Division Multiple Access (TDMA) protocols, desynchronization helps nodes organize their time slots.

[1]Supported by CU CP Academic Excellence Scholarship from Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University.
[2]Corresponding Author

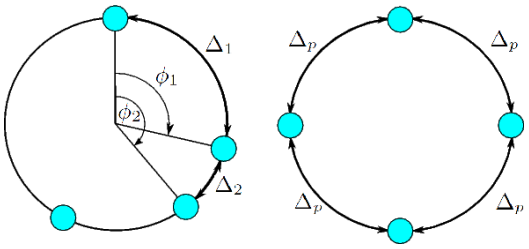


Fig. 1. Desynchronization framework.

In multiple analog-to-digital converters (ADC), desynchronization helps increase the overall sample rate by scheduling multiple ADCs to sample at different time.

In a past few years, several desynchronization protocols have been proposed such as DESYNC [1], Lightweight Coloring [5], M-DESYNC [6], DESYNC-ORT [3], V-DESYNC [4], and DWARF [2]. In order to propose new desynchronization protocols, unavoidably, researchers have to implement such previous works to compare with their own protocols. However, validating the implementations takes time and effort. Traditionally, researchers simulate their protocols on some network simulators (*e.g.* ns-2 [7], ns-3 [8], TOSSIM [9], GloMoSim [10]) and analyze the misbehavior of protocols from log or trace files. Those simulators do not provide a visualization tool that is specifically designed for validating the desynchronization behavior. In addition, when the number of nodes is high, their log or trace files are large and not easy to analyze.

In this paper, we propose DeVise, a visualizer for validating desynchronization protocols on multi-hop wireless sensor networks. DeVise reads configuration and trace files, and visualizes the protocol behavior. DeVise provides sufficient and useful information to help researchers reduce time and effort in the protocol validation process. We have evaluated DeVise with several desynchronization protocols such as DESYNC, and DWARF. The result is that DeVise helps researchers reduce time and effort in validating the implementations and effectively helps researchers explore advantages and drawbacks when comparing such protocols.

NodeID	X	Y
0	2.0	1.0
1	3.0	1.0
2	0.0	2.0
3	0.0	0.0
4	3.0	2.0
5	3.0	0.0

Fig. 2. Example of a topology input file.

Src	Dst	gain
0	1	-60.0
1	0	-60.0
0	2	-60.0
2	0	-60.0
0	3	-60.0
3	0	-60.0
1	4	-60.0
4	1	-60.0
1	5	-60.0
5	1	-60.0

Fig. 3. Example of a link gain input file.

Round	ID 0	ID 1	ID 2	ID 3	ID 4	ID 5
0	25	547	459	28	34	908
1	950	487	403	817	961	861
⋮	⋮	⋮	⋮	⋮	⋮	⋮
300	599	803	291	85	315	91

Fig. 4. Example of a phase input file.

Round	Mean of Δ	RMSE	NRMSE
0	166.66	200.00	1.73
1	166.66	193.13	1.15
⋮	⋮	⋮	⋮
300	166.66	7.45	0.04

Fig. 5. Example of a statistics input file.

The rest of the paper is organized as follows. Section II analyzes the factors to be considered when validating desynchronization protocols. Section III describes our visualization tool called DeVise. Then, we evaluate our tool in Section IV. Finally, Section V concludes the paper.

II. Visualizing Desynchronization Protocols

To visualize desynchronization protocols, there are several factors to be considered. We first focus on visualizing network topology and node connectivity. Normally, network topology and connectivity are defined in configuration files. Figure 2 and 3 are examples of topology and connectivity configuration files used in TOSSIM, a simulator for wireless sensor networks [9]. Instead of reading configuration files by eyes and using imagination, a visualization tool should help researchers easily see the overview of a network and how nodes are connected together.

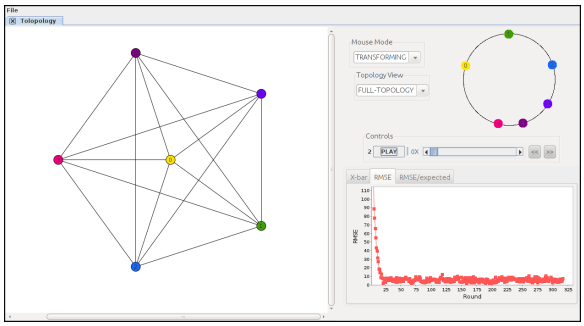


Fig. 6. Overview of DeVise: Left pane visualizes a network topology. Top-right pane visualizes a global phase circle. Bottom-right pane displays statistics graph.

Second, a visualization tool should help validate the protocol behavior. To validate the behavior of a desynchronization protocol, a visualizer should help researchers trace how nodes adapt their phases and see how far each node is separated away from its neighboring nodes. Nodes' phases are extracted from a simulation result which consists of phases of all nodes at every time period. Figure 4 is an example of a log file containing a simulation result of a network with 6 nodes. Apart of visualizing phases, if a visualizer displays a statistical result from the simulation, researchers are able to validate a protocol by comparing the statistical result with the behavior of a protocol at any specific point of simulation time. Figure 5 is an example of a statistics file containing values of mean of Δ ($\bar{\Delta}$), root mean square error (RMSE), and normalized root mean square error (NRMSE) which are used in [1]–[3].

Third, apart from visualizing network-wide connectivity, a visualization tool should provide a local connectivity view for multi-hop networks. Due to the fact that only 2-hop neighbors can interfere a transmission of a source node (*e.g.* the hidden terminal problem), there should be a node's local view that shows only neighboring nodes within an interference region (*i.e.* within 2-hop connectivity). With the local connectivity view, researchers can focus only on nodes that their transmissions affect an inspected node.

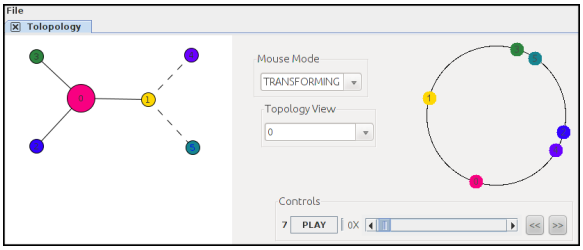
In the next section, we describe our proposed visualization tool that is designed based on these factors.

III. DeVise: A Desynchronization Visualizer

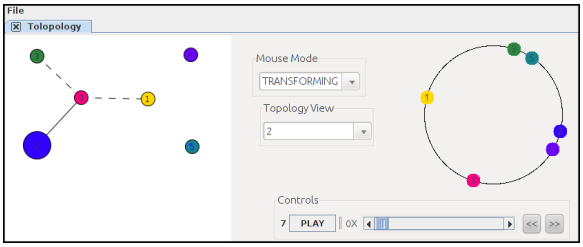
In this section, we propose DeVise, a desynchronization visualizer for multi-hop wireless sensor networks. DeVise helps researchers visualize and validate an implementation of a desynchronization protocol.

A. Network Topology and Local Connectivity for Multi-hop Network

By reading the topology and link gain input files, DeVise visualizes the topology and connectivity between each node pair. The left pane of Figure 6 visualizes a mesh topology whereas Figure 7 visualizes a butterfly topology. Additionally, DeVise is able to visualize local connectivity of each node for multi-hop networks by changing the topology view in



(a) Connectivity of node 0 (pink).



(b) Connectivity of node 2 (blue).

Fig. 7. Multi-hop topology. One-hop neighbors are connected with solid lines and two-hop neighbors are connected with dash lines.

the top-right pane from *FULL-TOPOLOGY* to <*nodeID*>. Figure 7a shows the local connectivity of node 0 and Figure 7b shows that of node 2. The inspected node is enlarged, 1-hop connectivity links are shown in solid lines, and 2-hop connectivity are shown in dash lines. The advantage of local connectivity is that, in desynchronization, only 2-hop neighbors can interfere the source node's transmission (*e.g.* the hidden terminal problem). With the local connectivity, we can easily identify which nodes can affect the transmission.

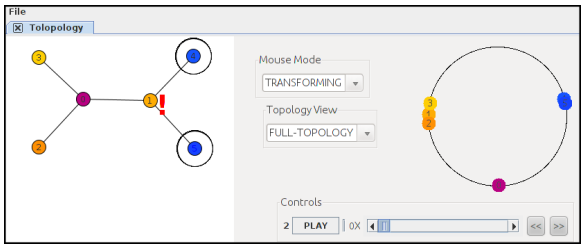


Fig. 8. Collision inspection.

B. Global Phase Circle and Wireless Interference Notification

By reading the phase input files, DeVise visualizes a global phase circle on the top-right pane (Figure 6, 7, and 8). The circle represents a time period which is used in the desynchronization framework as we described in Section I. Under the circle, there is an animation controller. The animation controller provides four functions: play, pause, skip to, and animation speed adjustment. These functions control the animation to visualize how each node adjusts its phase over time. The animation of a global phase circle is useful to validate the behavior of each node in each period. For example, it is easy to see how far each node adjusts its phase over time compared to other nodes.

In addition, DeVise colors each node based on its phase on the circle. If two nodes are at the same phase, they are colored with the same color. DeVise also colors nodes on the topology pane. The coloring helps researchers easily identify which nodes transmit packets at the same time. If two transmitting nodes are within 2-hop connectivity, they interfere each other. If they are far than 2-hop connectivity, they can be colored with the same color and can transmit simultaneously without interference. Figure 8 illustrates a scenario when node 4 and 5 transmit packets simultaneously. Both nodes are colored with the same blue color. In the right pane, both nodes are at the same phase on the circle. In the left pane, DeVise also notifies the packet collision. The two colliding nodes are emphasized with circles. The two transmissions are collided at the receiver; therefore, DeVise notifies with an exclamation mark at the receiver.

C. Statistics View

The metrics to measure the performance of desynchronization protocols are root mean square error (RMSE) over time and normalized root mean square error (NRMSE) over time (see [2] for details). Therefore, DeVise displays such data under the global phase circle (see the bottom-right pane in Figure 6). With this view, researchers can link the RMSE/NRMSE value at a specific point of time with a snapshot of the global phase circle. Additionally, the value of $\bar{\Delta}$ is used for cross-checking the validity of the statistics input (the time period must equal to $\bar{\Delta}$ times the number of nodes).

D. Local View and Phase Graph

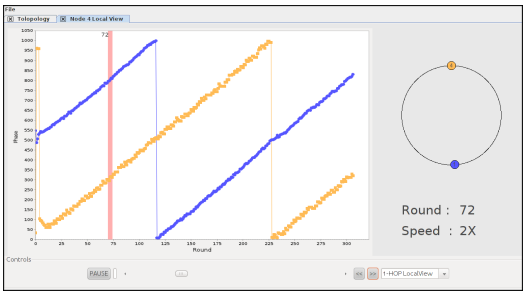
As we mentioned earlier, nodes that are far beyond 2-hop connectivity do not interfere a source node. Therefore, DeVise provides a local view for each node by right-clicking on a node to be inspected. In the local view, only 1-hop and 2-hop neighbors are displayed relatively to the inspected node on the phase circle. Other nodes are filtered out to let researchers focus on only nodes that can cause interference.

Additionally, in the local view, DeVise illustrates the phase graph of nodes in 1-hop or 2-hop connectivity. This phase graph helps researchers identify phases of nodes in quantitative values and see how phases are adapted over time.

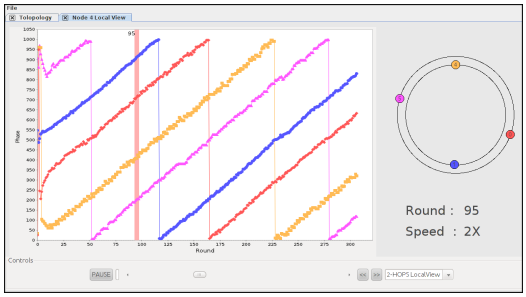
Figure 9 illustrates the local view of node 4 where Figure 9a shows 1-hop connectivity and Figure 9b shows 2-hop connectivity. 2-hop neighbors are displayed on an outer circle.

E. Topology Reformation

In some case, there are the large number of nodes and the topology is too complicated to find 1-hop and 2-hop neighbors. DeVise provides topology reformation by double clicking at a node to be inspected. The topology is reformed to several partial circles. Nodes forming the innermost circle are 1-hop neighbors of a node at the center point. Nodes forming the next outer circle are 2-hop neighbors and so on. Figure 10 illustrates the topology reformation when inspecting node 20



(a) 1-hop neighbors local view of node 4.



(b) 2-hop neighbors local view of node 4.

Fig. 9. Phase graph and local view.

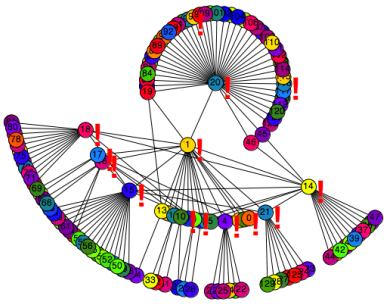


Fig. 10. Topology Reformation.

in a 130-node network. We can easily distinguish which nodes are 1-hop, 2-hop, and 3-hop neigbors of node 20.

IV. Experimental Evaluation

We have evaluated DeVise with two desynchronization protocols: DESYNC [1] and DWARF [2] with multi-hop support. We have implemented DESYNC and DWARF with TinyOS [11] and simulated with TOSSIM [9]. Then, DeVise parsed simulation logs to visualize and animated the behavior of each protocol. Figure 6 is captured from running DeVise with DESYNC whereas Figure 7 - 9 are captured from running DeVise with DWARF. Due to limited space, other screen captures are not shown.

We have found that DeVise usefully helps visualize and inspect the behavior of desynchronization protocols. We have pictorially seen the misbehavior of DESYNC when it worked on a multi-hop network. Additionally, by inspecting the phase graph and the local view, we have seen that even DWARF works very well on a multi-hop network but there are some cases that they do not fully utilize the network channel. Even some nodes are far beyond 2-hop connectivity but they do not use the same phase. The local phase circle shows that, in such cases, forces are well-balanced and nodes do not adjust their phases (see [2] for details of the algorithm) as illustrated in Figure 9b. Additionally, we have found that, for both DESYNC and DWARF, even in the perfect desynchrony state, nodes' phases are gradually increased as shown in Figure 9. This result from the phase graph leads us to further investigate and found that this behavior is caused by transmissoin delay. However, the relative phases are stable which is preferable because the goal of desynchronization is to separate nodes away equivalently in a time domain.

V. Conclusion

This paper has presented DeVise: a visualization tool that helps researchers explore and validate desynchronization protocols for multi-hop wireless sensor networks. DeVise provides several functionalities to visualize network topology and node connectivity, and to reduce time to validate a desynchronization protocol implementation. We believe that DeVise will benefit to researchers who work on developing desynchronization protocols.

References

[1] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: Self-organizing desynchronization and tdma on wireless sensor networks," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, 2007, pp. 11–20.

[2] S. Choochaisri, K. Apicharttrisorn, K. Korprasertthaworn, P. Taechalertpaisarn, and C. Intanagonwiwat, "Desynchronization with an artificial force field for wireless networks." ACM SIGCOMM Comput. Commun. Rev. 42, 2 (April 2012) (to appear).

[3] P. Taechalertpaisarn, S. Choochaisri, and C. Intanagonwiwat, "An orthodontics-inspired desynchronization algorithm for wireless sensor networks," in *IEEE International Conference on Communication Technology*, 2009.

[4] T. Settawatcharawanit, S. Choochaisri, C. Intanagonwiwat, and K. Rojviboonchai, "V-desync: Desynchronization for beacon broadcasting on vehicular networks," in *In the Proceedings of the 75th IEEE Vehicular Technology Conference (IEEE VTC)*, 2012.

[5] A. Motskin, T. Roughgarden, P. Skraba, and L. Guibas, "Lightweight coloring and desynchronization for networks," in *INFOCOM 2009, IEEE*, 2009, pp. 2383–2391.

[6] H. Kang and J. Wong, "A localized multi-hop desynchronization algorithm for wireless sensor networks," in *INFOCOM 2009, IEEE*, 2009, pp. 2906 –2910.

[7] "The Network Simulator ns-2," http://www.isi.edu/nsnam/ns/.

[8] "The Network Simulator ns-3," http://www.nsnam.org/.

[9] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003, pp. 126–137.

[10] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "Glomosim: A scalable network simulation environment," Tech. Rep., 1999.

[11] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *in Ambient Intelligence*. Springer Verlag, 2004.