Distributed TDMA Scheduling Using Topological Ordering in Wireless Sensor Networks

Thanh-Tung Nguyen, Linh-An Phan, Taejoon Kim, Taehong Kim*
School of Information and Communication Engineering
Chungbuk National University, Republic of Korea
{tungnt, linhan, ktjcc, taehongkim}@cbnu.ac.kr

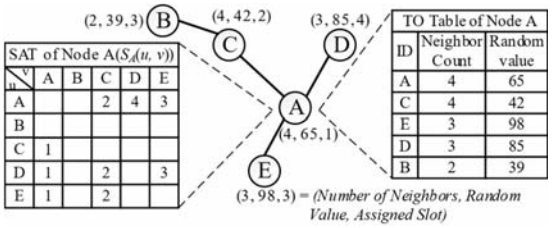
JaeSeang Lee+, JaeHyun Ham+
The 2nd R&D Institute
Agency for Defense Development+
{jslee15, mjhham}@add.re.kr+

***Abstract*—TDMA protocols can provide a reliable, collision-free data-transferring mechanism for wireless sensor networks. However, it requires an effective scheduling (time slot assignment) algorithm, which is a challenging issue, especially in wireless multi-hop networks due to random-based competition. In this paper, we propose DSTO, a distributed TDMA scheduling algorithm using Topological Ordering (TO). DSTO aims to reduce conflict of scheduling time among neighbor nodes by creating a Topological Order with local neighborhood size as the main priority factor. We implemented DSTO on OPNET Network Simulator and proved its effectiveness compared to DRAND in terms of running time and message overheads.**

Keywords—distributed, scheduling algorithm, DSTO, topological ordering, wireless sensor networks

I. INTRODUCTION

In contrast to CSMA protocols' inefficient channel utilization at low loads, TDMA protocols are popularly used in wireless sensor networks due to its collision-free and reliable data transfer. However, TDMA protocol requires time synchronization and transmission scheduling to guarantee reliable data transfer, which are challenging issues especially in wireless multi-hop networks. Among diverse suggestions, DRAND [1] and its variants [2], [3] are one of the representatives for TDMA scheduling algorithms due to the fully distributed design. In DRAND, nodes work in rounds. In each round (or trial), if a node wins the lottery, it can broadcast a REQUEST to all of its one-hop neighbors. If all the neighbors respond with a GRANT, the node can pick a time slot. On the other hand, if any of the neighbors replies with a REJECT, the round is considered failed and the node has to start again from the beginning in the next round. In other words, despite having received GRANTs from almost all neighbors, a trial is still considered unsuccessful once the node receives even just only one REJECT. Therefore, this is the downside of DRAND which leads to wasted resources. Thus, to overcome the disadvantages of slot assignment using random-based competition, in this paper, we propose a new algorithm in which each node waits until its turn, which has been decided based on topology information, to reserve a slot. This ensures that in each two-hop neighborhood, there can be only one node claiming a time slot at any time, which reduces collisions and conflicts between neighbor nodes. Therefore, nodes do not have to have to go through several trials and failures. They collect information and act only after having collected enough of it.



SAT of Node A($S_A(u, v)$)

v / u	A	B	C	D	E
A			2	4	3
B					
C	1				
D	1		2		3
E	1		2		

TO Table of Node A

ID	Neighbor Count	Random value
A	4	65
C	4	42
E	3	98
D	3	85
B	2	39

Fig. 1. Example of Slot Awareness Table and Topological Ordering Table

II. SCHEDULING USING TOPOLOGICAL ORDERING

A. Topological Ordering and Slot-awareness Table

We assume that each node obtains Topological Ordering (TO) information from neighbors within two-hop range, which consists of their number of neighbors (NB) and random value (RV), through exchanging of HELLO messages during the Neighbor Discovery Phase (NDP). Each node then determines a scheduling order for itself and nodes inside its two-hop neighborhood based on the collected information. Among neighboring nodes, one will perform slot assignment first if it has 1) the highest NB and 2) the highest RV in the case where there are nodes with the same NB. During the Scheduling Phase (SP), to follow the topological order, a node A collects and stores its neighborhood's slot information into a Slot-awareness Table (SAT), denoted S_A, in which $S_A(u, v) = a$ ($\forall\ a > 0$) that a node u is aware that another node v has taken time slot a. Fig. 1 shows an example of applying TO to a network of 5 nodes. Node A's TO Table is created after the NDP ends and this table's order is also the scheduling order in which A is the first and followed by C. Furthermore, A's SAT is constantly updated during SP so A can be aware of neighbors' slot as well as their awareness. For example, in A's SAT, $S_A(D, E) = 3$ as A knows that D is aware that E has selected slot 3.

B. Scheduling Algorithm using Topological Ordering

As stated earlier in the previous subsection, after finishing NDP, each node possesses a TO Table and will act accordingly to this table during the SP. Fig. 2 shows the scheduling algorithm of each node in the network. At the beginning of this stage, node A is at the top of its TO Table and automatically assigns its time slot and broadcasts one-hop *release* messages which carry the information about its slot. Information including 1) the list of one-hop neighbors that have responded to A, denoted R_A, and 2) the list of one-hop neighbors of A that have assigned their slot will also be piggy-backed inside the message as well if they exist. Then it waits for all of the one-hop neighbors' responses before returning to *idle-listening* state. On the other hand, in the case where A is not currently at the top of the TO Table, it enters the *idle-listening* state awaiting for messages sent from its one-hop neighbors. Upon receipt of a *release* message from a neighbor node B, node A first updates its SAT according to the information contained in the message and checks if it has been popped up to the top of the list (as one of its neighbor had finished slot assignment earlier). If it has, it performs exactly as

978-1-7281-1856-7/19/$31.00 ©2019 IEEE

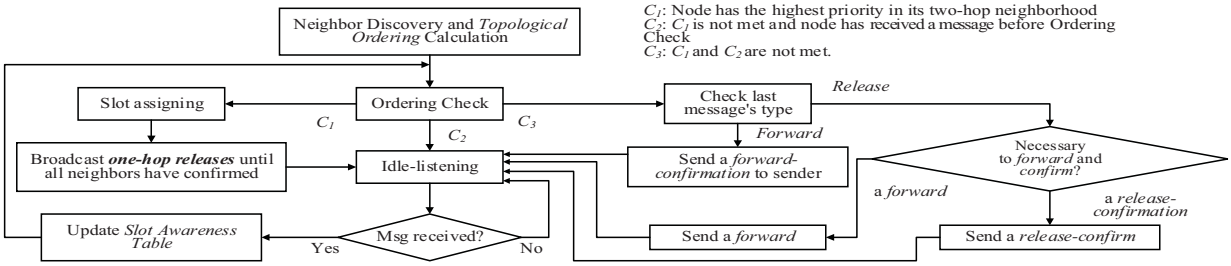


Fig. 2. Scheduling Algorithm's flowchart

above-mentioned steps to assign a time slot. Otherwise, node A checks if it needs to confirm to node B on the receipt of this message and forward information contained inside the message to one of its one-hop neighbors, denoted C. Depending on the result of the check, A will send either a *release-confirmation* or a *forward,* which serves as a confirmation to B as well. Node A only sends a *forward* if the latter or both of the following conditions is satisfied: 1) $A \notin R_B$ (in this case, A's *forward* message can also serve as a *release-confirmation* for B); 2) $S_A(A, x) > 0$ and $S_A(C, x) = 0$ ($\forall x \in$ *{A's one-hop neighbors}* & $x \notin$ *{C's one-hop neighbors}*). Once these conditions are both not met, A will stops sending *forward* messages to node C. Lastly, in another case where node A receives a *forward* message from neighbor node B with itself as a designated destination, it updates the SAT according to the information carried by that message and sends one *forward-confirmation* back to B.

III. Performance Evaluation

To study the performance of DSTO over DRAND [1], we use the OPNET Network Simulator. The number of nodes in a network ranges from 50 to 250 and all nodes are deployed randomly in an area of 300 x 300 m^2, which makes two-hop neighborhood sizes' range from 8 to 60. Fig. 3 shows that DSTO's average running time performance is 30% to 70% better than that of DRAND. In terms of message transmissions required, our algorithm generates less messages than DRAND by 20% to 40%, which is shown in Fig. 4. This is because in DRAND, nodes must compete in several rounds before being able to claim a slot, which increases running time and generates more messages than necessary. Whereas, in our algorithm, nodes collect information and claim a slot only at its turn,

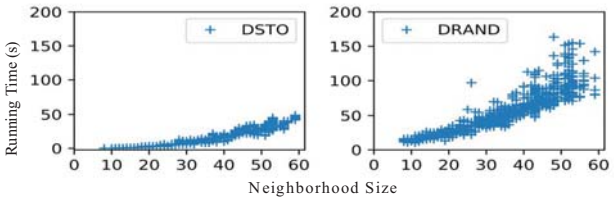


Fig. 3. Average running time per node for time slot assignment

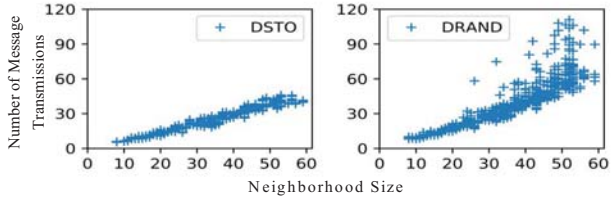


Fig. 4. Average message transmissions needed for time slot assignment

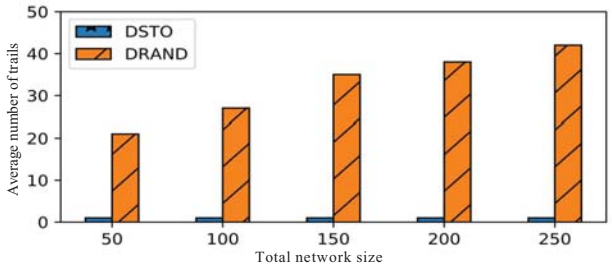


Fig. 5. Average numbers of trials (rounds) needed for slot assignment

reducing collisions with their neighbors. Furthermore, because of DRAND's random nature, there are significant output variations as the neighbor density increases. This is not the case for our algorithm. Because DSTO mainly employs Topological Ordering, so it yields similar or only slightly different outcomes in most cases. Additionally, Fig. 5 shows that in DSTO, nodes wait only try once to pick a time slot at its turn. On the other hand, DRAND nodes try and fail several times before they can secure a slot, which leads to longer running time and more energy wasted.

IV. Conclusion

This paper proposes the Topological Ordering based TDMA scheduling algorithm to decide the scheduling order of each node in a fully distributed manner as well as to reduce the running time and message overheads required to successfully allocate the time slots. The evaluations prove that the DSTO provides better performance than DRAND in respect of scheduling overhead and efficiency. In the future work, we plan to verify the effectiveness of the proposed algorithm through diverse performance comparison and mathematical analysis.

Acknowledgement

This work has been supported by the Small-scale Mobile Ad-hoc Network with Bio-networking Technology project of Agency for Defense Development (UD170094ED)

References

[1] I. Rhee, A. Warrier, J. Min and L. Xu, "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing,* vol. 8, no. 10, pp. 1384-1396, Oct. 2009.

[2] Y. Li, X. Zhang, T. Qiu, J. Zeng and P. Hu, "A Distributed TDMA Scheduling Algorithm Based on Exponential Backoff Rule and Energy-Topology Factor in Internet of Things," *IEEE Access,* vol. 5, pp. 20866 - 20879, Sep. 2017.

[3] K. Sato and S. Sakata, "A Power-Efficient Distributed TDMA Scheduling Algorithm with Distance-Measurement for Wireless Sensor Networks," *Wireless Personal Communications,* vol. 75, no. 2, p. 1511–1528, Mar. 2014.