Bio-inspired Resource Allocation for Multi-hop Networks

Youngjae Kim, Jungryun Lee*, Jiyoung Jung, Eutteum Kong, Uiseong You, Chanjae Lee
Chung-Ang University
Seoul, Republic of KOREA
+82-2-820-5820
{youngj89, jrlee, jiyoung, etkong, uiseong12, cjleeno22}@cau.ac.kr

Hyunho Choi
Han-Kyong University
Gyeonggi-do, Republic of Korea
+82-31-670-5297
hhchoi@hknu.ac.kr

Jaesung Park
Suwon University
Gyeonggi-do, Republic of Korea
+82-31-229-8216
jaesungpark@suwon.ac.kr

Myounghun Han, Bongsoo Roh, Chanyi Park, Mijeong Hoh, Hyungseok Choi
The 2nd R&D institute
ADD, Daejeon, Republic of Korea
+82-42-821-4259
{mengddor, saintroh, chyipark, hmj, chs}@add.re.kr

ABSTRACT

Recently, researches on resource allocation algorithms operating in a distributed way are widely conducted because of the increasing number of network nodes and the rapidly changing the network environment. In this paper, we propose new biologically inspired TDMA-based resource allocation scheme operating in a distributed manner in multi-hop networks. In this paper, we define a frame structure for the proposed algorithm and firing message structure which is a reference for resource allocation and propose the related operating procedures. Through simulation evaluations, it is shown that proposed algorithm works well in multi-hop networks.

Categories and Subject Descriptors

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

General Terms

Algorithm, Design, Performance

Keywords

Bio-inspired, DESYNC, resource allocation, multi-hop network, distributed

1. INTRODUCTION

There have been some proposals for bio-inspired algorithms to solve the various problems of communication networks as shown in Fig. 1 [1][2][3]. Bio-inspired algorithms use mathematical modeling of natural phenomena in order to apply the simple and distributed behavior of natural objects to various engineering fields. For these algorithms, convergent behavior is observed while each autonomous agent just obeys very simple rules repeatedly, based on local information without the help of a central controller. From such a feature of a bio-inspired algorithm, the necessity for the application of bio-inspired algorithms to multi-hop networks in which autonomous and distributed operations are required can be justified. The firefly algorithm, which was inspired by the flashing behavior of fireflies, is the foremost example of a bio-inspired algorithm [3]. The firefly algorithm has been applied to wireless networks with the purpose of synchronizing the time of performing periodic tasks or the time information of nodes in wireless networks. There has also been research regarding de-synchronization (DESYNC), which is the logical opposite of the synchronization shown in the firefly model [4][5][6][7].

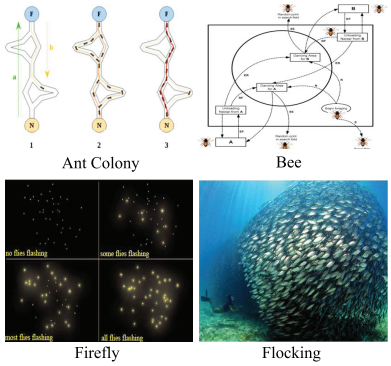

Figure 1: Example of bio-inspired algorithms

DESYNC is a simple algorithm for achieving desynchronization among the nodes in networks. Rather than synchronizing entities so that they perform periodic tasks at the same time, DESYNC entities perform their tasks as far away as possible from all of the other nodes. Suppose there are n nodes in a fully-connected network. When node i reaches the end of its cycle, it fires to indicate the termination of its cycle to the other nodes. In the DESYNC algorithm, node i records the times of the two firing events: the event that

*Corresponding author

.

BICT 2015, December 03-05, New York City, United States

DOI 10.4108/eai.3-12-2015.2262570

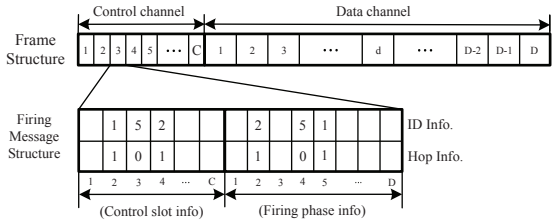

Figure 2: Proposed a frame and firing message structure

precedes its own firing ($\phi_{i+1}(t)$), and the one that occurs just afterwards ($\phi_{i-1}(t)$). Node i calculates the midpoint of its reference phase, and once it is calculated node i jumps toward it as follows:

$$\phi_i' = (1-\alpha)\phi_i(t) + \alpha\phi_{mid}(t),$$
$$where\ \phi_{mid}(t) = \frac{1}{2}[\phi_{i+1}(t) + \phi_{i-1}(t)] \quad (1)$$

Most DESYNC algorithms have been studied in single hop networks [4][5], so each node is able to receive control messages for resource allocation from all nodes. In addition, a specific scheme has not been discussed to obtain the resource allocation information of two-hop neighbor nodes in order to solve the hidden node problem of DESYNC algorithms in multi-hop networks [6][7]. In addition, collisions of control messages for real world scenarios have not been considered. In this paper, we propose a TDMA fair resource allocation scheme that operates in a distributed manner in multi-hop networks, called the Multi-Hop DESYNC (MH-DESYNC) algorithm, based on DESYNC algorithms. The MH-DESYNC algorithm defines a practical frame structure and a firing message structure in order to solve the hidden-node problem in real multi-hop networks.

2. PROPOSED ALGORITHM

2.1 Frame structure and firinmessage

We propose a new frame structure, as shown in Fig. 2. Each frame consists of a control channel and a data channel. The numbers of control time slots and data time slots are C and D, respectively. Each node allocates a control time slot and uses the same control time slot in every following frame. The node transmits its own firing information and the neighbor node's firing information to the neighbor nodes through the allocated control time slot. In this way, the neighboring nodes are able to completely update their two-hop neighbors' firing phase information. Through this information, each node updates its firing phase and allocates the data time slots.

Each node allocates a control time slot on the control channel when a node enters the existing network. Then, they transmit firing messages through the allocated control time slot. The firing message structure is partitioned into a control slot information area and a firing phase information area, as shown in Fig. 2. The control slot information is set to C and the firing phase information area is set to D. The control slot information includes the node ID and the *hop* information. Each node records the control slot information in the allocated control time slot. For example, if node 5 allocates the third control time slot, it records its own ID information (5) and the *hop* information (0) in the third control slot information area. Also, each node records the information of its neighboring nodes to the control time slots allocated to those neighbors. For example, if nodes 1 and 2, which are node 5's neighboring nodes, are allocated to the second and fourth control time slots, respectively, node 5 records its neighbor node ID information (1,2) and *hop* information (1) in the second and fourth control slots information area, respectively; this is shown in Fig. 2.

The firing phase information includes the node ID and *hop* information, similar to the control slot information area. Each node records the firing phase information of oneself and its neighboring nodes in the allocated firing phase time slots, similar to control time slot. Each node transmits a firing message that consists of the allocation information regarding the control time slot and firing phase for itself and its one-hop neighbors so that each node can know the firing message information of its two-hop neighbors. Therefore, MH-DYSYNC can solve the hidden node problem in multi-hop networks.

2.2 Operating procedure

In the MH-DESYNC algorithm, the operating procedure for logical firing is as follows: 1) allocate a control time slot, 2) allocate and update the firing phase, and 3) allocate the data time slots. This section explains each of these three procedures.

2.2.1 Control time slot allocation

Each node listens for firing messages during one frame, whenever a new node enters the existing network. Then node can know within two-hop neighbor's information about control time-slots. Each node chooses with equal probability a control time-slot which is not occupied within its two-hop neighborhood. Each node transmits a firing message through a control time-slot that they have selected. If a node allocates a control time-slot, it uses the same control time-slot in every next frame. However, if a conflict occurs in the control time-slot is, it chooses a control time-slot again in next frame.

This is an example that nodes may not be able to allocate a control time slot due to collisions, when the frame structure is $C = 4$, $D = 8$, as shown in Fig. 3. Given the arrangement of nodes shown in Fig. 3(a), suppose that node $n2$ allocates the fourth control time-slot in the frame and then nodes $n1$ and $n3$ concurrently enter the network. If nodes $n1$ and $n3$ choose the third and second control time slots, they are successful because the control time slots do not conflict as shown in Fig. 3(b). However, if nodes $n1$ and $n3$ choose the same control time slot, they are not able to allocate a control time slot due to the collision, as shown in Fig. 3(c).

Control time slot collision detection.

Each node detects the control time slot collision through the firing messages received from its neighboring nodes. If a node successfully allocates a control time slot, the neighboring nodes record the control slot information (ID and *hop* information) in the firing message of the control slot information area, as shown in Fig. 3(b). However, if there is a control time slot conflict, the neighboring nodes do not record the control slot information in the firing message of

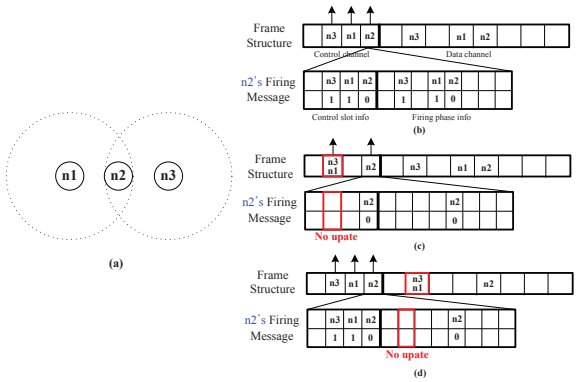

Figure 3: Example of MH-DESYNC collisions

the control slot information area, as shown in Fig. 3(c). In this way, the node detects the control time slot collision.

2.2.2 *Firing phase allocation and update*

After each node allocates the control time slots, it can recognize the allocated firing phase time slot in its two-hop neighborhood through the firing phase information area. Each node chooses with equal probability one of the total data time slots, and the phase of the selected data time slot determines the virtual firing phase time slot.

If two or more nodes choose the same firing phase time slot, the firing message collisions occurs in terms of the logical mean. This is an example that a node may or may not be able to allocate a firing phase time slot due to collisions, as shown in Fig. 3(d). Given the arrangement of nodes shown in Fig. 3(a), suppose that node n2 allocates the fourth control time slot in the frame, and then nodes $n1$ and $n3$ concurrently enter the network. If nodes $n1$ and $n3$ allocate control time slots without a collision and choose the fourth and second firing phase time slots, respectively, they are both able to allocate a firing phase time slot because they do not conflict as shown in Fig. 3(b). However, if nodes $n1$ and $n3$ choose the same firing phase time slot, they do not allocate a firing phase time slot due to the collisions, as shown in Fig. 3(d).

Firing phase time slot update.

The firing phase information of the node is transmitted in the next frame. We define the firing phase of node i in the n-th frame as $\phi_i(n)$, and $N_2(i)$ is the set of neighbors within two hops of node i. Then, we can define the next firing phase of node i as $n(i)$, and the previous firing phase of node i as $p(i)$. The equations are shown below

$$n(i) = arg_{j \in N_2(i)} min\{(\phi_j(n) - \phi_i(n)) \ mod \ D\} \quad (2)$$

$$p(i) = arg_{j \in N_2(i)} max\{(\phi_j(n) - \phi_i(n)) \ mod \ D\} \quad (3)$$

Node i updates the firing phase time slot in the $(n+1)$-th frame as shown in equation 4.

$$\phi_i(n+1) = ceil\left(\frac{\phi_{n(i)}(n) + \phi_{p(i)}(n)}{2}\right) \quad (4)$$

In other words, node i updates the firing phase time slot at the midpoint between the next firing phase $\phi_{n(i)}(n)$ and the previous firing phase $\phi_{p(i)}(n)$.

Firing phase time slot collision detection.

Each node can detect a collision of the firing phase time slot through the received firing message from its own neighboring nodes. If a node successfully allocates a firing phase time slot, the neighboring nodes record the firing phase information in the firing message of the firing phase information area, as shown in Fig. 3(b). However, if there is a firing phase time-slot collision, the neighboring nodes do not record the firing phase information to the firing message of the firing phase information area, as shown in Fig. 3(d). In this way, a node detects the collision of the firing phase time slot.

2.2.3 *Data time slot allocation*

Each node independently determines the amount of data time slots in each frame through this procedure. Each node calculates the midpoint between the previous firing phase $\phi_{p(i)}(n)$ and its own firing phase time slot $\phi_i(d)$. This is calculated as shown below

$$\phi_{i_l}(n) = ceil\left(\frac{\phi_{p(i)}(n) + \phi_i(n)}{2}\right) \quad (5)$$

Each node also calculates the midpoint between its own firing phase time slot $\phi_i(d)$ and the next firing phase $\phi_{n(i)}(n)$. This is calculated as shown below

$$\phi_{i_r}(n) = ceil\left(\frac{\phi_i(n) + \phi_{n(i)}(n)}{2}\right) \quad (6)$$

Node i allocates the data time slots between $\phi_{i_l}(d)$ and $\phi_{i_r}(d)$ in the n-th frame.

3. SIMULATION

In this Section, we perform a simulation to verify that the MH-DESYNC algorithm works well in multi-hop networks. Every node in the cycle graph has a degree of 2. Our simulations used 6 nodes in a ring topology. The number of control time slots is set to 5, and the number of data slots is set to 20. Suppose all of the nodes enter the network at the same time. Then we analyze the simulation results regarding the change in the firing phases.

In multi-hop networks, the simulation results show that there are four desynchronized configurations for the cycle graph, as shown in Figs. 4, 5, 6, and 7.

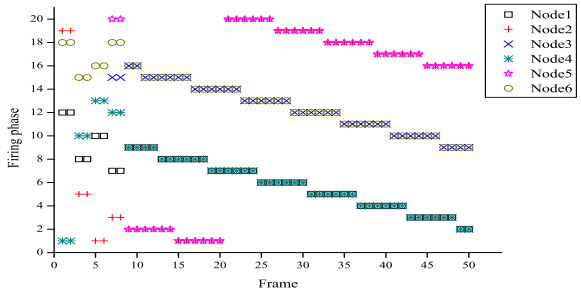

Figure 4: Simulation result 1

In the first configuration shown in Fig. 4, the firing phase time slot converges to three phase clusters of two nodes each. This configuration represents the best case in terms of the data time slot reuse factor. This case is a result of having a three phase cluster with two nodes because of consecutively

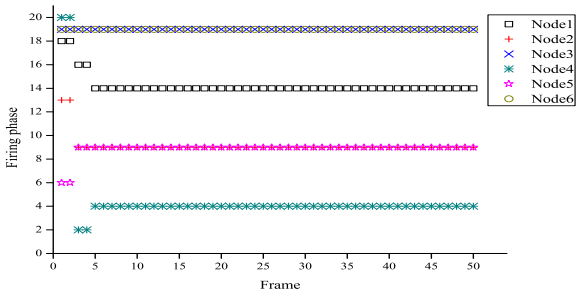

Figure 5: Simulation result 2

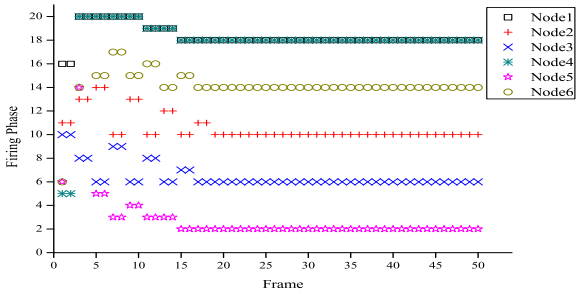

Figure 6: Simulation result 3

allocated initial firing phase time slots, as shown in Fig. 4. Then, each firing phase cluster of the two nodes ultimately updates the same firing phase. The second configuration is shown in Fig. 5. Here the firing phase time slot converges to four phase clusters, each of which has either two nodes or a single node. Therefore, each node has allocated five data time-slots in every frame. The third configuration is shown in Fig. 6. Here the firing phase time slot converges to five phase clusters of either two nodes or a single node. Therefore, each node has allocated four data time slots in every frame. The fourth configuration is shown in Fig. 7. Here the firing phase time slot has converged to six phase single nodes. This configuration represents the worst case scenario in terms of the data time slot reuse factor.

Through this simulation, we have confirmed that the proposed MH-DESYNC works well in a multi-hop network. We also observe that there are different desynchronized configurations from different initial firing phase time slots in multi-hop networks. In the cycle graph, when six nodes enter the network at the same time, all nodes require at least 3 frames, with an average of 6.57 frames, to allocate the control time

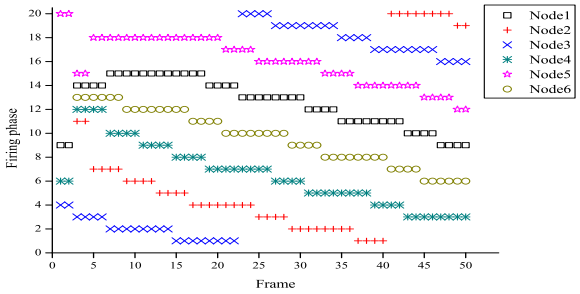

Figure 7: Simulation result 4

slots without a collision. Similarly, all nodes require at least three frames and an average of 7.29 frames to allocate the firing phase time slots without a collision.

4. CONCLUSION

In this paper, we proposed a MH-DESYNC algorithm that is a bio-inspired TDMA resource allocation scheme operating in a distributed manner in multi-hop networks. In MH-DESYNC, we proposed a frame structure that includes a control channel and data channel, and an operating procedure to solve the hidden node problem. The proposed MH-DESYNC does not transmit to a physical firing signal through actual data time slots, unlike the previous DESYNC algorithm; instead, we defined a control channel and a practical firing message. Our simulation shows that MH-DSYNC can resolve the hidden node problem effectively and can find different results according to the initial firing phase.

5. ACKNOWLEDGMENTS

This research was supported by Agency for Defense Development (ADD-IBR-245).

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1007) supervised by the IITP (Institute for Information & communications Technology Promotion)

6. REFERENCES

[1] Dorigo, M., Birattari, M., Stutzle, T., Ant colony optimization, *IEEE computational Intelligence Magazine*, vol. 1, no. 4, pp. 28-39, Nov, 2006.

[2] Olfati-Saber, R., Flocking for multi-agent dynamic systems: algorithms and theory, *IEEE Transactions on Automatic Control*, vol.51, no.3, pp. 401- 420, March 2006.

[3] G. Werner-Allen, G. Tewari, A. Patel, R.Nagpal, and M. Welsh, Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects, *international conference on Embedded networked sensor systems*, pp142-153, Nov, 2005.

[4] Julius Degesys, Ian Rose, Ankit Patel, and Radhika Nagpal, *Self-organizing Desynchronization and TDMA on Wireless Sensor Networks,* BIOWIRE 2007, pp192-203, April, 2007.

[5] Ankit Patel, Julius Degesys, Radhika Nagpal, *Desynchronization: The Theory of Self-Organizing Algorithms for Round-Robin Scheduling,* SASO '07, pp.87-96, July, 2007.

[6] Julius Degesys and Radhika Nagpal, *Towards Desynchronization of Multi-hop Topologies, SASO 08*, pp.129-138, Oct, 2008

[7] Arik Motskin, Tim Roughgarden, Primoz Skraba and Leonidas Guibas, Lightweight Coloring and Desynchronization for Networks, *INFOCOM 2009, IEEE*, pp.2383-2391, April, 2009