Transmission Reordering in Self-organizing Network Coordination Framework

Kohji Tomita(✉) and Akiya Kamimura

National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Ibaraki 305-8568, Japan
{k.tomita,kamimura.a}@aist.go.jp

Abstract. By virtue of the rapid progress of IoT technology, communication devices are increasing drastically. Along with the increase, collision of transmission often happens, resulting in restricted throughput. This restriction is mainly caused by a hidden node problem. To resolve that difficulty, a promising methodology is Time Division Multiple Access (TDMA) based on a Pulse Coupled Oscillator (PCO) model. Among them, Self-organizing Network Coordination Framework (SoNCF) presents various benefits. However, in some network topologies, the performance of SoNCF is degraded because the order of random initial phases of nodes is unchanged. As described in this paper, we analyze the effect of transmission ordering on SoNCF using graph theory concepts. We also consider its extension to resolve it through reordering. Its effectiveness was confirmed through simulation.

Keywords: Wireless network · Pulse coupled oscillator
Synchronization · Self-organization

1 Introduction

Internet of things (IoT) devices including sensor nodes are increasing drastically. This trend is expected to continue because of the rapid progress of IoT technology [4]. The current major protocol for wireless communication uses Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA). Using CSMA/CA method, declining throughput caused by collision of transmission is a salient issue when many nodes mutually communicate. This degraded performance is mainly attributable to a phenomenon known as a hidden node problem: a node cannot detect two-hop neighbor's transmission and simultaneous transmission results in collision of packets. The Request To Send/Clear To Send (RTS/CTS) mechanism was proposed to mitigate this difficulty, but its effectiveness has remained limited [5].

Unlike these, a promising methodology to resolve the problem is Time Division Multiple Access (TDMA) based on a Pulse Coupled Oscillator (PCO) model. Along this line, much study has been conducted [1–3,6,7].

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2019
Published by Springer Nature Switzerland AG 2019. All Rights Reserved
P. Cong Vinh and V. Alagar (Eds.): ICCASA 2018/ICTCC 2018, LNICST 266, pp. 14–24, 2019.
https://doi.org/10.1007/978-3-030-06152-4_2

Among them, Self-organizing Network Coordination Framework (SoNCF) has various benefits as described below [6].

- In a stable state after convergence, no collision occurs if the connection network topology is unchanged.
- When the topology of connection network changes dynamically, communication timing is adaptively organized in a distributed manner.
- A sub-optimal division number is computed explicitly and shared among nodes.
- When nodes are sending data repeatedly, congestion can be relaxed by using additional slots.

In spite of the benefits described above, in some network topologies, the SoNCF performance is degraded. In SoNCF, each node transmits its data in its own slot one by one. This ordering relation is characterized by a pushing relation among them in the phase domain. The relative order of transmission is decided by the initial randomness of phase assignment to the nodes. The order is not problematic in many cases, but when many long cycles exist in the connection network, the order can take effect.

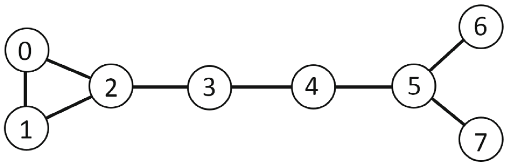
As described in this paper, we analyze the effect of ordering using concepts in graph theory, and consider its extension to resolve such difficulties. We extend the framework so that the order of transmission can be organized to prevent undesirable pushing of nodes in the phase domain.

2 SoNCF

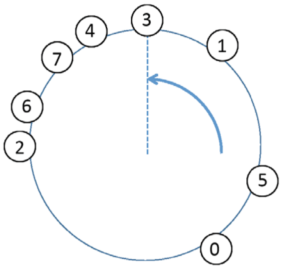
This section presents a review of the framework of SoNCF from [6]. This framework of network coordination where nodes of a wireless network can transmit data without collision is based on TDMA using PCO.

Each node adjusts its transmission timing in consideration of the timing of neighbor nodes. Only nodes within two hops are considered for network coordination. The transmission timing of two-hop neighbors is conveyed via one-hop neighbors.

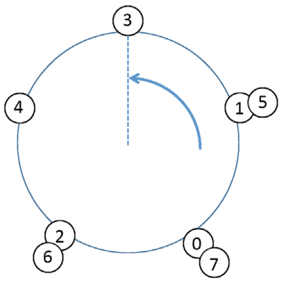
We can illustrate the concepts in SoNCF through the use of an example. Let us consider a graph in Fig. 1(a) composed of eight nodes. Each circle represents a communication node with its ID number. Each edge represents direct communication possibility. In SoNCF, we separate the transmission timing of each node so that no collision occurs and each node transmits one by one in a local perspective. Figure 1(b) presents an initial random arrangement of phases observed from the global perspective. Each node changes its phase counterclockwise in this figure. We assume that each node transmits at the top position denoted by the dotted line, where node 3 is located. The phase difference between nodes is insufficient. Efficient communication is not possible in this state. The nodes calculate the sub-optimal division number as five, where the optimal division number is four. Based on the value, the nodes mutually interact in a pulse coupled manner and adjust their transmission timing adaptively. Finally, their phase is separated into five groups as in Fig. 1(c). Nodes 1 and 5, for instance, transmit simultaneously but do not cause collision because the nodes are sufficiently separated.



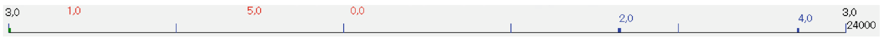
(a) Example network composed of eight nodes.



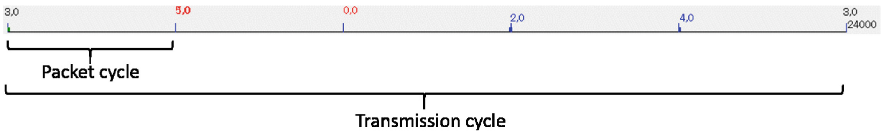
(b) Global initial phases.



(c) Global final phases.



(d) Local view of initial phases from node 3.



(e) Local view of final phases from node 3.

Fig. 1. SoNCF.

We can also observe how the situation is observed from a node. Each node obtains timing information of one-hop neighbors (called real nodes) directly by receiving communication, and of two-hop neighbors (called virtual nodes) through communication via a real node. Figure 1(d) and 1(e), respectively represent arrangements of neighbors' phases from a local perspective of node 3 in the initial and converged states. In Fig. 1(d), the lower blue node IDs (i.e., nodes 2 and 4) and upper red node IDs (i.e., nodes 0, 1, and 5) respectively represent real and virtual nodes seen from node 3. These linear representations correspond to circular representations in Fig. 1(b) and (c). Phases of nodes 6 and 7 are not known to node 3.

In SoNCF, the duration for a node to send a packet is fixed and called a packet cycle. One cycle of transmission by all the nodes, called a transmission cycle, is calculated by multiplying the packet cycle and the division number as presented in Fig. 1(e). In this paper, the packet cycle and transmission cycle are designated respectively as the packet period and transmission period to avoid confusion with cycles of graphs.

For more details related to SoNCF, one can refer to an earlier report [6].

3 Extension by Reordering

3.1 Ordering Problem

When nodes are interacting in the framework of SoNCF, the relative phase order of each node is decided by initial random phase assignment. Then, in the original framework, the relative order of transmission among nodes within two-hops does not change.

If the connection topology is tree-like i.e., if the overall system does not include long cycles, then an undesirable pushing situation in the phase domain disappears by itself. However, if the overall system includes many long cycles, then a cycle of pushing nodes could be generated at the time of execution.

The length of a pushing cycle is at least equal to the division number. Each node can use its packet period if they are equal. However, if a pushing cycle is longer than the division number, then an undesirable situation occurs. Such a pushing state reduces the effective length of transmission slot and prevents enjoyment of the full length of transmission. It also increases the danger of collision.

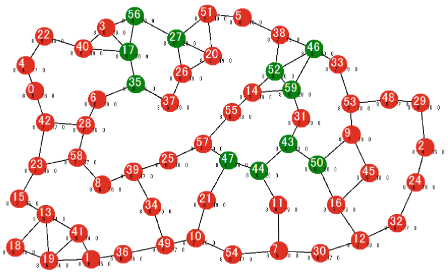
This can be stated more clearly as follows. The nodes involved in the cycle transmit their packets in turn in one transmission period. This transmission means that, for a cycle with length n such that $n > N$, where N is the division number, the overall transmission period is divided into n rather than N, giving a shorter duration of sending than the packet period. We designate such cycles as *long pushing cycles*. Moreover, nodes not included in the cycle might be pushed indirectly by nodes in the long pushing cycles. This situation continues. We designate the nodes pushed indirectly by a long pushing cycle as *accompanying nodes*. Actually, cycles in the squared connection graph are a concern.

We distinguish graphs of two kinds and cycles on them.

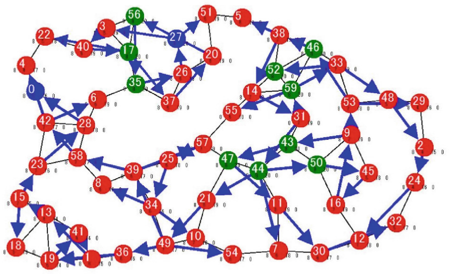
1. For a *connection graph* G where edges are undirected and where they represent direct communication possibility for the current arrangement of nodes, we consider the *squared connection graph* G^2 where edges are between one-hop or two-hop neighbors in the connection graph. Cycles in squared connection graphs are potential pushing cycles at the time of execution. Squared connection graphs and cycles on them do not change unless the network topology is changed.

2. We consider a graph called a *pushing diagram*, where edges are directed and represent the pushing relation in the phase domain between nodes within two-hop neighbors in the connection graph at some moment of execution. Cycles represent the current phase pushing state among nodes, and are a subgraph of the corresponding squared connection graph with a direction in edges. Pushing diagrams are time-dependent.

Example. We visualize the situation using the SoNCF simulator [6]. Figure 2(a) presents an example of a connection graph; Fig. 2(b) presents additional information of the pushing relation by blue arrows. The arrows are between one-hop or two-hop neighbors because, in SoNCF, direct interaction of pushing occurs only neighbors within two hops. In this figure, the width of an arrow represents the strength of pushing. This figure depicts a situation in which many pushing relations exist.



(a) Connection graph.



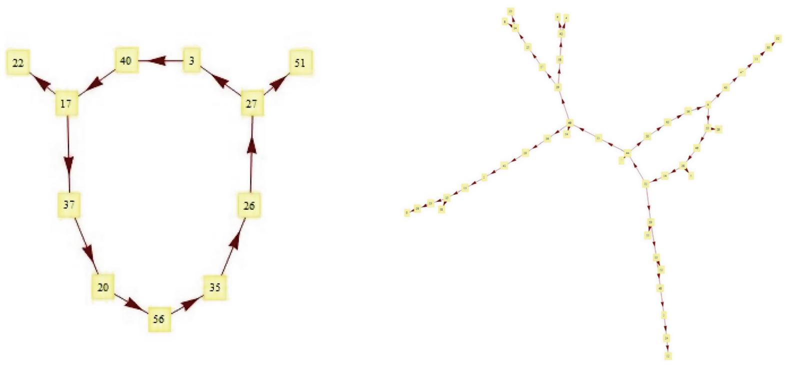
(b) Pushing diagram.

Fig. 2. Connection graph (a) and pushing diagram (b).

Let us consider connected components of the pushing diagram. In the case of a pushing diagram corresponding to Fig. 2(b), where edges are shown by blue arrows, we have two components as in Fig. 3. They are shown based only on a connective relation and are shown irrespective of the location in Fig. 2. Figure 3(a) shows a long pushing cycle as

$$3 \to 40 \to 17 \to 37 \to 20 \to 56 \to 35 \to 26 \to 27 \to 3$$

with length 9. Therefore, the transmission period is divided by 9. Nodes 21 and 52 are accompanying nodes, continuously pushed by the nodes in the cycle. Figure 3(b) includes a long pushing cycle of length 10, with 39 accompanying nodes.



(a) (b)

Fig. 3. Connected components of a pushing diagram.

3.2 Phase Jump

Once such state with a long pushing cycle is generated, the original framework maintains the transmission ordering. This state continues unless the network structure changes. We consider resolving such pushing states.

It would be a solution that nodes in the pushing cycle change their relative transmission order appropriately. However, the nodes do not have information of global pushing relation. Accompanying nodes are outside of the cycle but pushed as well from the nodes in the cycle indirectly. Each node cannot decide whether it is in the long pushing cycle or not. The only available information related to pushing is that it is pushed continuously from a neighbor.

Randomization. The cause by which a long pushing cycle is generated is that a node is in an inappropriate relative phase position of transmission. If several nodes in the cycle change their order to appropriate positions, then the cycle does not exist any more.

In general, there is an appropriate order of transmission that does not cause a long pushing cycle. In other words, the division number is chosen to satisfy this condition.

Some nodes can detect a cycle because nodes in a long pushing cycle (or accompanying nodes) are pushed continuously by a neighbor node if there is such a long pushing cycle in the network. However, it is difficult to generate an appropriate order directly from a purely local perspective. Therefore, we use a stochastic method. A node that might be in the cycle changes its relative order by jumping to a different position. Figure 4 presents that obtained in the case in which there are eight slots and where a node jumps to an empty slot, which is located three slots ahead.

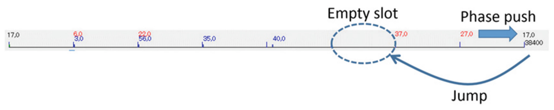


Fig. 4. Jumping.

As presented in a report by [7], a similar method was proposed. In our method, detection of pushing is explicit because the division number is shared among all the nodes. In addition, a target of jumping is chosen from possible candidates corresponding to the slots to which each node should be aligned.

On Collision. Phase jumping possibly causes problems of collisions.

In an earlier report of the literature [6], a method of using empty slots without collision was proposed. In that method, a node is guaranteed to perform extra utilization exclusively among its neighbor nodes. The method is based on assigning privilege to a node by exchanging the amount of remaining data to be sent. Then, an empty slot is searched to be used. The emptiness criterion is that the slot is not used by one-hop or two-hop neighbors.

By introducing additional data sent in the packet, reordering while avoiding collision would be possible. In the case treated in this paper, however, to avoid complication, using such additional information is not preferred. Instead, each node jumps at a low probability to reduce collisions.

Summary. To summarize the contents above, we use the following method for reordering:

1. Set a minimum threshold of pushing value th to consider and a value d of accumulated pushing value to jump.
2. Calculate the sum of the pushing value in consecutive steps pushed at a strength greater than threshold th.
3. The node changes its. transmission timing to an empty slot randomly if the sum exceeds d.

The pushing value is calculated as the overlapping rate of its own slot of packet period and its preceding node's slot. Its range is $[0, 1)$. As described herein, we use the following parameter values: $th = 0.05$ and $d = 2.5$.

We implemented this method in the simulator and conducted simulation experiments.

4 Simulation and Results

4.1 Mesh Network

We first consider a connection network composed of 60 nodes as shown in Fig. 2. We start in a mode without reordering and at the midst of time reordering starts. Figure 5 is a resulting graph showing the change of average pushing values, i.e., the average of the pushing values for all nodes.

Before the reordering starts, the average pushing value is stable at about 0.28 after initial perturbation until time 50. Reordering starts at time 191; phase jumping happens 75 times between time 192 and 205. The average pushing value drops and then remains stable at nearly zero. The occasional positive value after convergence is caused by natural fluctuation because of the phase interaction and is sufficiently small.

The three graphs portrayed in the figure respectively illustrate the pushing state before, during, and after reordering. As the right graph shows, pushing arrows remain, but they are sparse. In appropriate phase ordering, the pushing arrows are weak and disappear soon after appearance, whereas arrows in long pushing cycles are strong, stable, and persistent with nearly constant strength.

This result demonstrates the method's effectiveness.

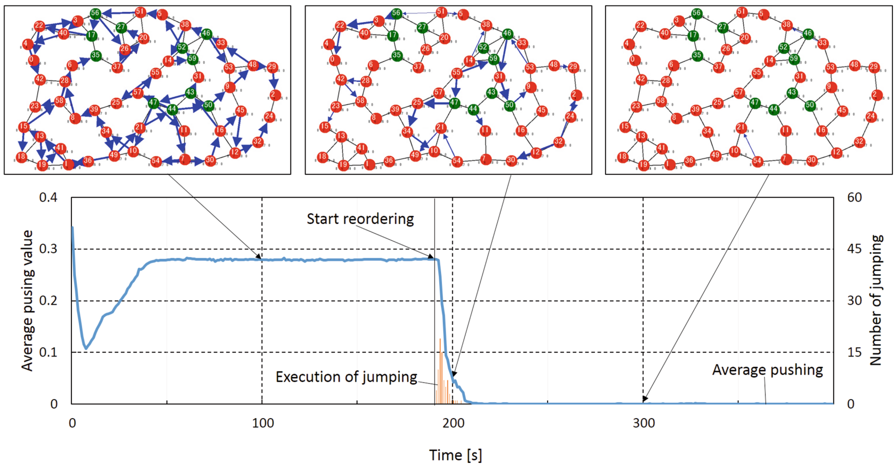


Fig. 5. Change of the average pushing value.

4.2 Regular Network

Next, let us consider connection networks with n^2 nodes arranged in the 2D regular lattice $n \times n$, where $n > 1$. Considering it as an undirected graph, if n is even, there is a Hamiltonian cycle, i.e., a cycle that visits each vertex exactly once. Therefore, one pushing cycle can include all the nodes. It is possible, in principle, that the available transmission period is divided by the number of all nodes n^2 in the worst case.

Hereinafter, we consider the case with $n = 10$ and examine the effect of reordering by comparing cases with and without reordering. Figure 6 presents changes of the average pushing values in all nodes for both cases. This is the average of ten runs of simulation each. As the graph shows, the value without reordering is nearly stable at about 0.16 after convergence. However, the value with reordering converges nearly to zero, demonstrating that the pushing state is resolved.

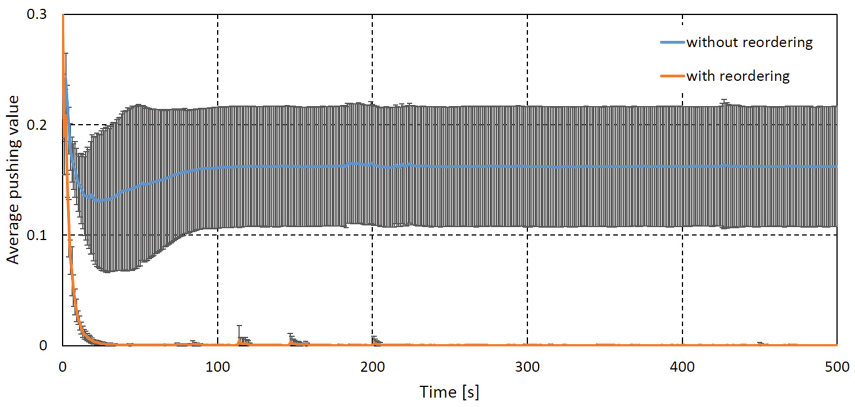


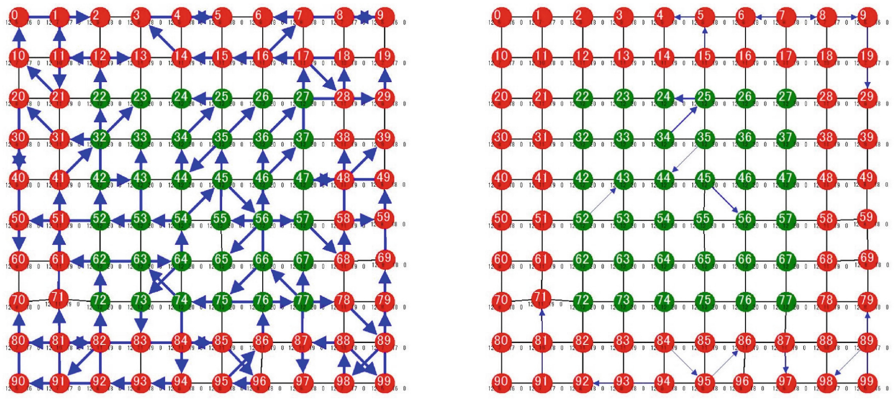
Fig. 6. Comparison of average pushing value with and without reordering.

The improvement can also be confirmed visually from different figures. In typical cases, the pushing arrows are shown in Fig. 7, without (a) and with (b) reordering and in stable states.

4.3 Smaller Division Number

In the simulations described above in Sect. 4.2, we used division number 11, which was decided according to the SoNCF algorithm. To examine extreme cases, we conduct additional simulations for the same 10×10 regular lattice. We fix the division number manually and note the convergence of phases. The optimal division number for this graph is 5.

From the conducted simulation, we confirmed that the phase converged to a stable state for division numbers until 7, started from 11. It was difficult to converge for the number 6, a nearly optimal division number, in the current setting of simulation.



(a) Without reordering (b) With reordering

Fig. 7. Typical stable states with and without reordering.

5 Discussion and Conclusion

We have presented an extension of SoNCF so that the order of transmission is organized in a decentralized manner to an appropriate one. The effectiveness of reordering was confirmed through simulations conducted in mesh and regular lattice networks under various conditions.

The parameter values were chosen empirically. Experimentation showed that these values affect the convergence time but that they have little effect on the convergence. To reduce collision, the stochastic value is useful for *d*. An optimal setting will depend on the calculated division number and the network structure. Further consideration will be necessary.

As stated in the previous section, when the division number is nearly optimal, it is difficult to converge to a stable state. It is difficult to realize in a local manner without using global information.

To use this framework in real applications, further study is required. We assumed for the simulation that the communication condition is sufficiently good and therefore employed a simple model of interference. Actual applications have errors or asymmetry in communication. In addition, energy consumption is expected to be difficult. We have already conducted preliminary experiments using developed communication modules. By coping with the problems described above, we are planning to improve the modules and conduct hardware experiments in future studies.

References

1. Anglea, T., Wang, Y.: Phase desynchronization: a new approach and theory using pulse-based interaction. IEEE Trans. Signal Process. **65**, 1160–1171 (2017). https://doi.org/10.1109/TSP.2016.2633246
2. Degesys, J., Rose, I., Patel, A., Nagpal, R.: DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks. In: International Symposium on Information Processing in Sensor Networks (PSN), pp. 11–20 (2007). https://doi.org/10.1109/IPSN.2007.4379660
3. Gentz, R., Scaglione, A., Ferrari, L., Hong, Y.-W.P.: PulseSS: a pulse-coupled synchronization and scheduling protocol for clustered wireless sensor networks. IEEE Internet Things J. **3**, 1222–1234 (2016). https://doi.org/10.1109/JIOT.2016.2576923
4. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**, 1645–1660 (2013). https://doi.org/10.1016/j.future.2013.01.010
5. Jayasuriya, A., Perreau, S., Dadej, A., Gordon, S.: Hidden vs. exposed terminal problem in ad hoc networks. In: Proceedings Australian Telecommunication Network and Applications Conference, pp. 52–59 (2004)
6. Kamimura, A., Tomita, K.: Self-organizing network coordination framework enabling collision-free and congestionless wireless sensor networks. J. Network Comput. Appl. **93**, 228–244 (2017). https://doi.org/10.1016/j.jnca.2017.06.002
7. Kubo, Y., Sekiyama, K.: Communication timing control with interference detection for wirelesssensor networks. EURASIP J. Wireless Commun. Networking, Article ID: 54968 (2007). https://doi.org/10.1155/2007/54968