Sensors and Materials, Vol. 31, No. 5 (2019) 1419–1427
MYU Tokyo

S & M 1868

Slotframe Partitioning-based Cell Scheduling for IEEE 802.15.4 Time Slotted Channel Hopping

Jung-Hyok Kwon,[1] Eui-Jik Kim,[2*] and Dongwan Kim[3**]

[1]Smart Computing Laboratory, Hallym University,
1 Hallymdaehak-gil, Chuncheon, Gangwon-do 24252, South Korea
[2]School of Software, Hallym University,
1 Hallymdaehak-gil, Chuncheon, Gangwon-do 24252, South Korea
[3]Department of Electronic Engineering, Dong-A University,
37 Nakdong-Daero 550beon-gil, Saha-gu, Busan 49315, South Korea

(Received April 30, 2018; accepted March 13, 2019)

Keywords: cell scheduling, IEEE 802.15.4 time slotted channel hopping (TSCH), industrial Internet of Things (IIoT), slotframe partitioning, 6top protocol

This paper presents a slotframe partitioning-based cell scheduling (SPCS) for IEEE 802.15.4 time slotted channel hopping (TSCH). SPCS divides a slotframe into multiple partitions of different lengths and sequentially allocates cells depending on the depth of sensor nodes. The operation of SPCS consists of slotframe partitioning and cell allocation. In the former, the root determines the number and length of partitions, taking into account the depth of sensor nodes and the number of slotOffsets required for each partition. In the latter, a pair of neighboring sensor nodes adds the cells included in a specific partition of the slotframe using a 6P transaction. SPCS minimizes end-to-end delay by sequentially allocating cells according to the depth of sensor nodes. To evaluate the performance of SPCS, an experimental simulation is conducted. The results show that SPCS has an end-to-end delay lower than that of the existing 6P.

1. Introduction

Industrial Internet of Things (IIoT) has recently received considerable attention in various industries, since it has a great advantage in various industrial applications such as real-time monitoring, process automation, and predictive maintenance.(1–3) In IIoT, a number of sensor nodes are generally employed, and such nodes send their packets to the intended receiver via wireless links. This environment can cause long delays and frequent connection failures, which can result in severe system damage including machine malfunction and process failures. Therefore, IIoT essentially requires communication technology that guarantees deterministic delay and high reliability.

To meet the stringent requirements of IIoT, the IEEE 802.15.4 standard was amended in 2015, and time slotted channel hopping (TSCH) was added to its medium access control (MAC) layer.(4)

*Corresponding author: e-mail: ejkim32@hallym.ac.kr
**Corresponding author: e-mail: dongwankim@dau.ac.kr
https://doi.org/10.18494/SAM.2019.2262

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

TSCH uses a slotframe structure consisting of slotOffsets and channelOffsets to support contention free-based time slotted access with multichannel and channel hopping capabilities. Contention free-based time slotted access provides a predictable delay by eliminating collisions between sensor nodes. The multichannel capability increases the network capacity by allowing multiple sensor nodes to send packets simultaneously via nonoverlapping channels. Finally, channel hopping improves the network reliability by reducing the effects of interference between the sensor nodes. However, the IEEE 802.15.4 standard does not specify a scheduling method for allocating and deallocating resources in TSCH.(5)

The Internet engineering task force (IETF) 6TiSCH working group (WG) proposed the 6TiSCH operation sublayer (6top) protocol, abbreviated 6P, for resource allocation and deallocation in TSCH.(6) 6P employs the routing protocol for low-power and lossy networks (RPL) as the default routing protocol.(7) RPL forms a destination-oriented directed acyclic graph (DODAG), in which a number of sensor nodes are connected to a single root in a multihop manner. For resource allocation and deallocation, 6P defines a 6P transaction that adds/deletes cells within the slotframe through a negotiation between sensor nodes in the vicinity. Each cell is represented by a pair of slotOffset and channelOffset in the slotframe. 6P transactions can either be two-step and or three-step 6P transactions.(8) The former is used when the sender selects the cells to be added, while the latter is used when the receiver is responsible for selecting the cells. In both cases, the selected cells are maintained in a list called the CellList. 6P enables sensor nodes to allocate and deallocate resources by adding and deleting cells. However, 6P may suffer from long end-to-end delays since it randomly selects cells without considering network environmental factors such as the routing path and interference.

Many studies have been conducted to address this problem. Duy *et al.* proposed a CellList generation method that considers the partial cell allocation density within the slotframe.(9) For this, one slotframe is divided into portions of equal length, and the cells in the portion with the most available cells are selected for transmission. However, since this method considers only the density of the portion, it may suffer from a long end-to-end delay when packets are delivered to the root via multihop links. Moreover, there is a limitation that the sender cannot select the cell to be allocated since the method only uses three-step 6P transactions. Hosni and Théoleyre proposed a cell scheduling method that divides a slotframe into blocks of different lengths and adds the cells considering the depth of the sensor node (i.e., hop distance from the sensor node to the root).(10) However, this method determines the length of each block considering only the specific route in the topology, thereby leading to transmission failure and long end-to-end delays due to interference from sensor nodes in different routes.

In this paper, we propose a slotframe partitioning-based cell scheduling (SPCS) for IEEE 802.15.4 TSCH that sequentially allocates cells depending on the depth of sensor nodes to minimize end-to-end delay. The operation of SPCS consists of slotframe partitioning and cell allocation. In slotframe partitioning, the root determines the number and length of partitions in order to divide a slotframe into multiple partitions. To this end, the root counts the depth of the leaf node included in the route with the maximum number of hops and calculates the number of slotOffsets required for each partition. During cell allocation, a pair of neighboring sensor nodes adds the cells included in the specific partition of the slotframe using a 6P transaction.

An experimental simulation was conducted to demonstrate the effectiveness of SPCS. The results showed that SPCS achieves better performance with respect to the end-to-end delay compared with the existing 6P.

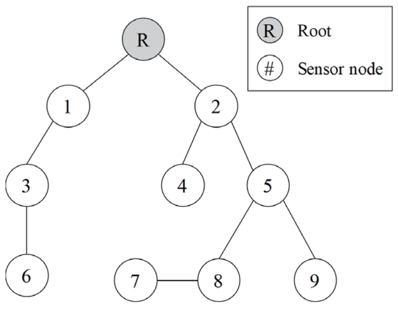
The remainder of this paper is organized as follows. In Sect. 2, the operation of SPCS is described in detail. In Sect. 3, the simulation setup and results are presented. Finally, Sect. 4 concludes this paper.

2. Operation of SPCS

In SPCS, the root maintains the network information table (NIT), which includes the routing table for all sensor nodes and the list of interfering nodes (i.e., one-hop neighbors) for each sensor node. Figure 1 shows an example of network topology and NIT. When the network topology is constructed as shown in Fig. 1(a), the root maintains the NIT depicted in Fig. 1(b). In our work, we used RPL to construct the network topology (i.e., DODAG). Thus, the root maintains the routing table for all sensor nodes by receiving the destination advertisement object (DAO) packets. In RPL, each node has a list of interfering nodes by receiving the DODAG information object (DIO) packets from one-hop neighbors. To make the root maintain the list of interfering nodes for each sensor node, we assume that each sensor node sends DAO packets with the list of interfering nodes. We further assume that the interfering range for each node is the same and that the slotframe structure is defined and notified by the root.

The operation of SPCS consists of slotframe partitioning and cell allocation. Slotframe partitioning is the process of dividing the slotframe into multiple partitions of different lengths, and cell allocation is the process of adding cells to the slotframe using information for the partitions (i.e., index and length of partitions). Slotframe partitioning and cell allocation are performed by the root and sensor node, respectively.

For slotframe partitioning, the root selects the routes with leaf nodes (i.e., sensor nodes without any child) from all routes in the network topology and assigns the index to a selected route according to the depth of the leaf node. The larger the depth of the leaf node is, the smaller the index of the route becomes. If the multiple routes have the leaf node of the same depth, the index of the route depends on the ID of the leaf node. In this case, the smaller the



(a)

Sensor node	Next hop	Interfering node
1	-	R, 3
2	-	R, 4, 5
3	1	1, 6
4	2	2, 5, 7, 8
5	2	2, 4, 8, 9
6	3	3, 7
7	8	4, 6, 8
8	5	4, 5, 7, 9
9	5	5, 8

(b)

Fig. 1. (a) Network topology and (b) NIT.

ID of the leaf node is, the smaller the index of the route becomes. Figure 2 shows the selected routes and the index of each selected route when the network topology is equal to the condition shown in Fig. 1(a). In the figure, route 0 has the leaf node with the largest depth.

The root decides the number of partitions by determining the depth of the leaf node within route 0. In Fig. 2, the number of partitions is four. Each partition has its own index. A partition with a small index is located in the front portion of the slotframe. The root determines the length of each partition (i.e., the number of slotOffsets included in each partition). For this, the root determines the weight of each partition, which is the minimum number of slotOffsets required for each partition. Each sensor node can have the packets to be sent to its parent. Thus, to obtain the minimum number of slotOffsets required for each partition, the number of slotOffsets for the traffic flow of a pair of nodes (i.e., child → parent) is set to one. For each route, the traffic flow is assigned to different partitions depending on the depth of the child (i.e., sender). In the event that the traffic flow is generated by the leaf node, it is assigned to partition 0. When the depth of the child is reduced by one, the traffic flow is assigned to the partition whose index is incremented by one. Figure 3 shows the traffic flows assigned in each partition when the selected routes are the same as in Fig. 2.

Afterward, to determine the weight of each partition, the root counts the number of slotOffsets required for each partition, considering the numbers of traffic flows, non-overlapping channels, interfering nodes, and redundant nodes. The initial weight of each partition is set to zero; it is incremented by one when the number of slotOffsets is incremented by one. If a sensor node has a child, it should forward the packets received from the child. In other words, the sensor nodes close to the root have more packets to send. Therefore, if the traffic flow is assigned in partition *i*, the sender requires *i* + 1 different slotOffsets to forward the packets received from its descendants. The slotframe has multiple channelOffsets to utilize non-overlapping channels. In this regard, the sensor node and its interfering node can send packets at the same time using different channelOffsets. Moreover, sensor nodes that do not interfere with each other can send packets using the same slotOffset and channelOffset. When one sensor node is used for different traffic flows (i.e., redundant node) in a partition, it sends and receives the packets using different slotOffsets. Figure 4 shows the weight of each partition when the number of non-overlapping channels is set to 4. The traffic flows assigned to each

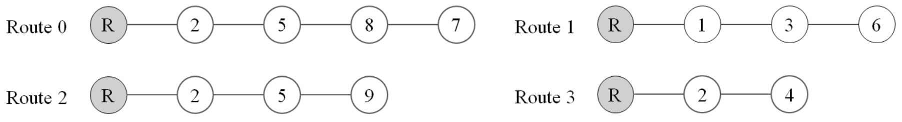


Fig. 2. Selected routes.

Partition 0	Partition 1	Partition 2	Partition 3
7 → 8 6 → 3 9 → 5 4 → 2	8 → 5 3 → 1 5 → 2 2 → R	5 → 2 1 → R 2 → R	2 → R

Fig. 3. Traffic flows assigned in each partition.

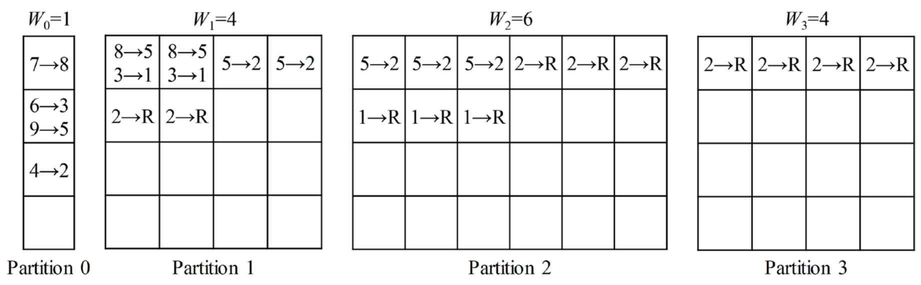


Fig. 4. Weight of each partition.

partition are shown in Fig. 3; the interfering nodes for each sensor node are shown in Fig. 1(b). In the figure, W_i is the weight of partition i.

Upon determining the weight of each partition, the root determines the length of each partition using the number of slotOffsets, the number of partitions, and the weight of each partition. More precisely, the length of each partition is calculated as

$$L_i = \begin{cases} \left\lceil kW_i \Big/ \sum_{i=0}^{n-1} W_i \right\rceil, \ i \in [0, n-2] \\ k - \sum_{j=0}^{n-1} L_j, \ i = n-1, \end{cases} \tag{1}$$

where L_i is the length of partition i, k is the number of slotOffsets, and n is the number of partitions. If the weight of each partition is the same as in Fig. 4 and k is equal to 100, the length of each partition becomes 7, 27, 40, and 26. Finally, the root sends the number of partitions and the length of each partition to sensor nodes through an enhanced beacon (EB). A sensor node selects one of the partitions by comparing its own depth with the partition index upon receiving the EB.

For cell allocation, each sensor node uses 6P transactions with information for each partition. SPCS can use both two-step and three-step 6P transactions for cell allocation. For two-step 6P transactions, if the sender needs to add cells to the slotframe, SPCS randomly selects the specific number of available cells included in the selected partition to generate a CellList. To identify the available cells, each sensor node maintains the scheduling information, which includes the list of the cells already added by neighbors and itself. In the scheduling information, the cell is represented by (slotOffset, channelOffset). The sender sends the NumCells (i.e., the number of cells to be added) and CellList to the receiver. The receiver randomly selects the requested number of cells listed and sends it to the sender. When the cells are successfully added to the slotframe, both nodes update their scheduling information. For three-step 6P transactions, the sender sends only NumCells to the receiver, and the receiver generates the CellList using the information for the partition and sends it to the sender. Finally, upon receiving the CellList, the sender selects the cells and sends the selected cells to the receiver. Figure 5 shows an example of cell allocation with a two-step 6P transaction between nodes A and B.

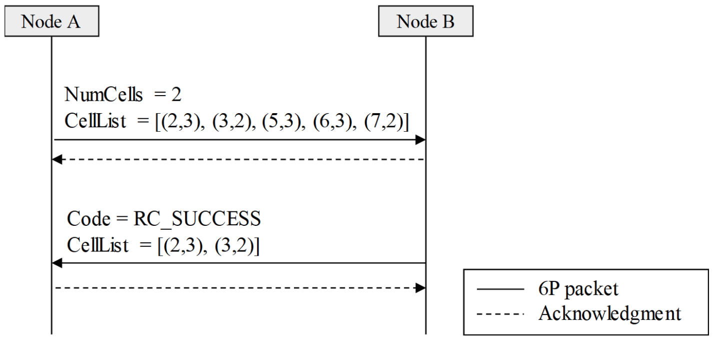

Node A
Node B
NumCells = 2
CellList = [(2,3), (3,2), (5,3), (6,3), (7,2)]
Code = RC_SUCCESS
CellList = [(2,3), (3,2)]
6P packet
Acknowledgment

Fig. 5. Cell allocation with two-step 6P transaction.

3. Simulation Setup and Results

To verify the effectiveness of SPCS, we conducted an experimental simulation under various scenarios and compared the end-to-end delay for SPCS with that of the existing 6P. In the simulation, the number of sensor nodes was varied from 5 to 50, and such nodes were randomly deployed in a 100 × 100 m^2 area. Each sensor node had at least one neighbor and the maximum transmission range for each sensor device was set to 20 m. The number of slotOffsets in a slotframe was set to 100 and 200 in each scenario, and the number of channelOffsets was set to 12. The duration of the time slot was set to 15 ms, so the durations of one slotframe in each scenario were 1.5 and 3 s. The packet size was set to 128 bytes and each sensor node sent 20 packets per minute. The total simulation time was 300 s and the simulations were repeated 1000 times. Table 1 shows the simulation parameters in detail.

Figure 6 shows the end-to-end delay for varying number of sensor nodes when the number of slotOffsets is set to 100. Overall, SPCS outperforms the existing 6P since it adds cells to the slotframe depending on the depth of the sender. As the number of sensor nodes increases, the number of routes and the depth of the leaf nodes tend to increase. Therefore, the end-to-end delay increases when more sensor nodes are deployed. In particular, SPCS achieves better performance than the existing 6P as the number of sensor nodes increases. This is because when the number of sensor nodes increases, the existing 6P increases the number of sensor nodes sending packets via the next slotframe. Unlike the existing 6P, SPCS guarantees that the packets are delivered to the root before the end of the slotframe. In the simulation, SPCS had an average end-to-end delay 10.2% shorter than that of the existing 6P.

Figure 7 shows the end-to-end delay when the number of slotOffsets increases to 200. In the figure, SPCS achieves a shorter end-to-end delay than the existing 6P regardless of the number of slotOffsets. This is because SPCS allows the sensor nodes to add cells considering their depth. The increased number of slotOffsets leads to longer end-to-end delays since the sensor nodes have to wait for a long time to access the channel. The difference in end-to-end delay between SPCS and the existing 6P increases as the number of slotOffsets increases. This is because the sensor node using the existing 6P waits for a longer period when it sends the packet through the next slotframe. On average, SPCS has an end-to-end delay 13.6% shorter than that of the existing 6P.

Table 1
Simulation parameters.

Parameter	Value	Parameter	Value
Physical layer	IEEE 802.15.4	Number of slotOffsets	100, 200
MAC layer	IEEE 802.15.4 TSCH	Number of channelOffsets	12
Routing protocol	RPL	Duration of time slot	15 ms
Simulation area	100 × 100 m^2	Packet size	128 bytes
Number of sensor nodes	5–50	Traffic rate	5 packets/min
Maximum transmission range	20 m	Total simulation time	300 s

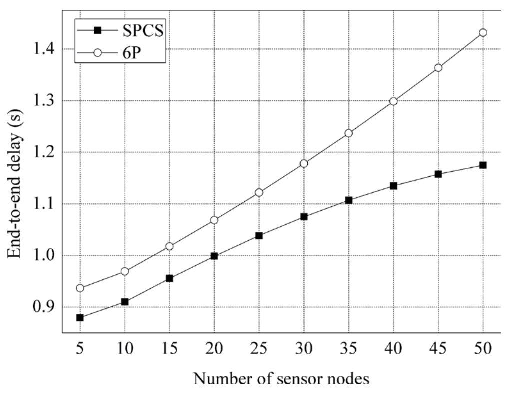


Fig. 6. End-to-end delay for varying number of sensor nodes (100 slotOffsets).

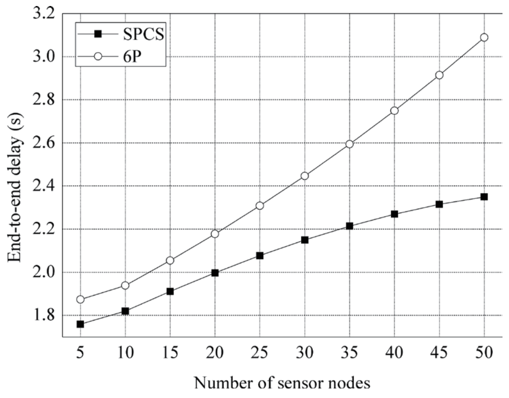


Fig. 7. End-to-end delay for varying number of sensor nodes (200 slotOffsets).

4. Conclusions

In this paper, we propose SPCS, which divides a slotframe into multiple partitions of different lengths and sequentially allocates cells depending on the depth of sensor nodes. The operation of SPCS consists of slotframe partitioning and cell allocation. The former is performed to determine the number and length of partitions, and the latter is performed to add the cells included in a specific partition of the slotframe. An experimental simulation was conducted to verify the effectiveness of SPCS, and the end-to-end delay of SPCS was compared with that of the existing 6P. The results showed that SPCS outperforms the existing 6P. Specifically, when the number of slotOffsets is set to 100 and 200, the end-to-end delays of SPCS are respectively 10.2 and 13.6% shorter than that of the existing 6P.

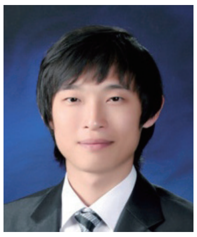
Acknowledgments

This work was supported by Korea Evaluation Institute of Industrial Technology (KEIT) grant funded by the Korea government (MOTIE) (20001056, Development of Low-Power Massive MISO SWIPT for IoT Wearable).

References

1 H.-H. Lee, J.-H. Kwon, and E.-J. Kim: J. Supercomput. **74** (2018) 4385. https://doi.org/10.1007/s11227-016-1915-4
2 R. T. Hermeto, A. Gallais, and F. Theoleyre: Comput. Commun. **114** (2017) 84. https://doi.org/10.1016/j.comcom.2017.10.004
3 J.-H. Kwon, H.-H. Lee, Y. Lim, and E.-J. Kim: Appl. Sci.-Basel **6** (2016) 427:1. https://doi.org/10.3390/app6120427
4 IEEE Standard for Low-Rate Wireless Networks, IEEE Standard 802.15.4-2015, Apr. 2016.
5 M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel: IEEE Sens. J. **16** (2016) 550. https://doi.org/10.1109/JSEN.2015.2480886
6 6TiSCH Operation Sublayer Protocol (6top), IETF Standard draft-ietf-6tisch-6top-protocol-12, Jun. 2018.
7 RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, IETF Standard RFC 6550, Mar. 2012.
8 A. Aijaz and U. Raza: IEEE Sens. J. **17** (2017) 6825. https://doi.org/10.1109/JSEN.2017.2746183
9 T. P. Duy, T. Dinh, and Y. Kim: Comp. Stand. Inter. **53** (2017) 80. https://doi.org/10.1016/j.csi.2017.03.008
10 I. Hosni and F. Théoleyre: Comput. Commun. **110** (2017) 103. https://doi.org/10.1016/j.comcom.2017.05.014

About the Authors



Jung-Hyok Kwon received his B.S. degree in Electronic Engineering from Soongsil University, Seoul, Korea in 2010, his M.S. degree in Electronics and Computer Engineering from Korea University, Seoul, Korea in 2012, and his Ph.D. degree in Convergence Software from Hallym University, Chuncheon, Korea in 2019. From April 2013 to June 2015, he was a research engineer at software R&D Lab., LIG Nex1 Co., Ltd. Since March 2019, he has been a research professor at the Smart Computing Laboratory, Hallym University. He was selected as a recipient of the Global Ph.D. Fellowship Program sponsored by the National Research Foundation of Korea in 2016. His research interest includes design and performance analysis of medium access control protocols for next-generation communication systems, vehicular communications (V2X), machine-to-machine communications, machine learning, and Internet of Things. (jhkwon@hallym.ac.kr)

Eui-Jik Kim received his B.S., M.S., and Ph.D. degrees in Electronics and Computer Engineering from Korea University, Seoul, Korea in 2004, 2006, and 2013, respectively. From September 2005 to December 2005, he was with the Intel Korea R&D Center at Intel Corporation. From February 2006 to July 2009, he was with the DMC R&D Center at Samsung Electronics Co., Ltd., Suwon, Korea. From August 2009 to August 2013, he was with the Advanced Institute of Technology at KT Corporation, Seoul, Korea. Since September 2013, he has been a professor at the School of Software, Hallym University. His current research interests include wireless/mobile networks with an emphasis on QoS guarantee and adaptation, wireless sensor networks, wireless LAN/PAN/BAN, vehicular communications (V2X), Internet of Things, machine-to-machine communications, and ICT convergence services. (ejkim32@hallym.ac.kr)

Dongwan Kim received his B.S. degree in Electrical Engineering from Korea University, Seoul, South Korea, in 2003, his M.S. degree in Information and Communication Engineering from POSTECH, Pohang, South Korea, in 2006, and his Ph.D. degree in Electrical and Computer Engineering from Korea University in 2015. He is currently an assistant professor at the Department of Electronics Engineering, Dong-A University, Busan, South Korea. From 2006 to 2016, he was a senior engineer at Samsung Electronics Company Ltd., Suwon, South Korea. His current research interests include the hardware efficient design of communication systems and low- power MAC protocol in mobile communication systems. (dongwankim@dau.ac.kr)