Computer Standards & Interfaces 53 (2017) 80–88

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Distributed cell selection for scheduling function in 6TiSCH networks

Thang Phan Duy, Thanh Dinh, Younghan Kim*

School of Electronic Engineering, Soongsil University, Dongjak-gu, Seoul 156-743, Korea

A R T I C L E I N F O

Keywords:
Industrial internet of things
IEEE 802.15.4e
Time slotted channel hopping
6TiSCH
Scheduling function
6top
6P

A B S T R A C T

Recently, the new IETF 6TiSCH Working Group (WG) was created to enable the IPv6 over the deterministic Time Slotted Channel Hopping (TSCH) mode of the 802.15.4e standard for the Industrial Internet of Things (IIoT). Due to the dynamic nature of the target network environment, the 6TiSCH WG is now calling for an efficient distributed scheduling through a Scheduling Function in which the neighbor nodes negotiate with one another to add/delete cells to satisfy the bandwidth requirement. A Scheduling Function consists of the following two main points: when a node should add/delete one or more cells to its neighbor, and the cells that should be selected. Although several Scheduling Function studies have been conducted, the primary focus of the current works is the first point, while the optimal selection method regarding the cells remains an unresolved issue. In this paper, a reliable distributed cell-selection mechanism for the Scheduling Function in the 6TiSCH networks is proposed. The simulation and experiment results show that the proposed mechanism helps in the reductions of the negotiation-error ratio, the number of colliding cells and the signaling overhead compared with the state-of-the-art approaches.

1. Introduction

During recent years, a great number of new applications have been built on the Internet of Things (IoT) where smart and connected entities cooperatively construct a communication network of things [1]. The IoT has been applied in various areas such as healthcare, energy management, and entertainment [2]. Some of the most promising applications related to the industrial sector [3] have fostered the rise of the term "Industrial Internet of Things" (IIoT) [4]. In the IIoT ecosystem, the widespread deployment of wireless sensor networks plays an important role in the provision of sensing information [5] for automation processes at a low operational cost, for pervasive communication, and for energy efficiency [6]. Several standards such as WirelessHART [7] and ISA100.11a [8] were proposed to meet the reliability, energy efficiency, and low end-to-end latency requirements of industrial applications. In 2012, the IEEE released an extended version of the original IEEE 802.15.4-2011 [9], called "IEEE 802.15.4e-amendment" [10]. The IEEE 802.15.4e provides the Time Slotted Channel Hopping (TSCH) mode based on the time synchronization and channel hopping techniques. The purposes are the achievements of a deterministic performance and an ultra-low power consumption. Recently, the TSCH mode was ratified officially as an amendment to the IEEE 802.15.4-2015 [11]. Thanks to the TSCH, the network is more robust against the unreliability that is caused by multipath fading and external interference in wireless environments [12,13]. To address the issues related to the scheduling, network addressing, routing, and interconnection, the IETF *IPv6 over the TSCH mode of IEEE 802.15.4e* (6TiSCH WG) [14,15] was established to enable a further adoption of the IPv6-enabled protocols (e.g., CoAP [16], 6LoWPAN [17], and RPL [18]) with the link-layer TSCH mode.

In the 6TiSCH networks, all of the sensor nodes are synchronized and follow a common schedule. The schedule is modeled by a slotframe structure that is constructed from a repeated sequence of timeslots and a set of available communication channels (i.e., up to 16 channels) [19]. The schedule is also known as a "cell matrix". A cell is indexed by a slotOffset (i.e., indicating a timeslot) and a channelOffset (i.e., indicating a channel). The Scheduling Function allocates each cell to a pair of nodes for a communication at a given time and on a specific channel. The design of the schedule is very important as it affects the performance of the TSCH networks (e.g., traffic, latency, and throughput). Unfortunately, the IEEE802.15.4e TSCH does not specify the way that the schedule is built and maintained [20].

In the literature, a number of scheduling schemes [21–24] were proposed. The schemes proposed in [21] and [22] require the bidirectional communication between the gateway and the sensor nodes for the scheduling, thereby incurring a high signaling overhead. In the cases of [23] and [24], the scheduling algorithms are sophisticated, and their performance is shown in simulations only. Their practicality in the real wireless sensor networks is still questionable. As a consequence, these schemes may not fit into the 6TiSCH networks. Recently,

* Corresponding author.
E-mail addresses: thangecp@gmail.com (T.P. Duy), thanhdcn@dcn.ssu.ac.kr (T. Dinh), younghak@ssu.ac.kr (Y. Kim).

http://dx.doi.org/10.1016/j.csi.2017.03.008
Received 11 June 2016; Received in revised form 19 March 2017; Accepted 19 March 2017
Available online 22 March 2017

the 6TiSCH WG has defined an operation sub-layer, called 6top [25], where the 6top protocol (6P) is implemented to enable a distributed approach for the scheduling [26]. The 6P allows the neighbor nodes to negotiate with one another to add/delete (allocate/deallocate) cells. Therefore, the scheduling operations in the 6TiSCH networks can obtain a flexibility level at a lower cost. In addition, the distributed approach for scheduling is driven by a Scheduling Function that is located in the 6top, which specifies 1) when to add/delete one or more cells to a neighbor and 2) which cells should be selected (i.e., how to select the slotOffsets and channelOffsets for the cells). Two main Scheduling Function schemes were proposed for the 6TiSCH networks, including Scheduling Function Zero (SF0) [27] and PID-based SF [28]. However, both of the Scheduling Function schemes use the same random cell-selection mechanism without considering the slotOffset conflicts and possible interferences. Through the simulations and real experiments presented in the paper, the use of such a random cell-selection mechanism shows that a node suffers from a significantly high chance regarding the selection of the slotOffsets which its neighbor nodes have already used to communicate with the other nodes, especially in the cases of dense networks. Consequently, the cells corresponding with these slotOffsets are not validated in a timely manner for the transmission. Moreover, the allocated cells may be used by more than one pair of nodes in the network, which also potentially leads to a high collision probability. The collision affects the packet transmission reliability [29] and may cause the dropping of packets.

In this paper, we propose a novel distributed cell-selection mechanism in which a slotframe is divided into a number of portions. Each portion is characterized by a density value that is conceptually the ratio between the number of unavailable slotOffsets (i.e., assigned for the communication) in that portion and the length of the portion. The nodes monitor the density of each portion through a *Statistical Monitoring* function that is provided by the Scheduling Function. When a node wants to add a number of cells to its neighbor, the node selects the corresponding slotOffsets in the portion with the lowest density value between the two nodes. In this way, the probability of the negotiation errors is mitigated. In the proposed mechanism, a channelOffset assignment method presented in [30] is also implemented, which enforces a rule that any pair of nodes that is a two-hop rank away is prohibited from transmitting on the same channel. Moreover, the proposed mechanism provides new procedures to ensure that each node effectively selects the channelOffsets for the cells to reduce the collisions.

In summary, the contributions of this paper are as follows:

- A distributed cell-selection mechanism that reduces the negotiation-error ratio and scheduling collisions is first proposed.
- Then, a new 6P protocol that can practically plug the proposed mechanism directly into the Scheduling Function is implemented.
- Through the extensive simulation and testbed results, we show that the proposed mechanism achieves a significant improvement compared with the state-of-the-art schemes in terms of the negotiation-error ratio, number of colliding cells, packet delivery ratio, signaling overhead, and network lifetime.

The remainder of this article is organized as follows: Section 2 provides an overview of the different scheduling approaches in the 6TiSCH networks. In Section 3, the details of the proposal and its integration into the Scheduling Function are presented. Section 4 reveals the performance through a simulation and its implementation. Lastly, in Section 5, the conclusion is drawn.

2. Scheduling in 6TiSCH networks

2.1. Related works

In the TSCH networks, two main scheduling approaches are used to build and maintain schedules, as follows: centralized scheduling and distributed scheduling. The centralized approach is more appropriate for stable networks. In this approach, a central entity collects the information of the whole network such as the bandwidth requirement and the network topology, and it computes a common schedule. The authors of [21] introduced the TASA, a centralized traffic-aware scheduling algorithm. The TASA exploits a matching and coloring technique from graph theory to assign the slotOffsets and the channelOffsets, respectively, to all of the nodes across the tree topology. In another way, the TMCP [22] partitions the network into sub-trees and assigns different channels and timeslots to those nodes belonging to different sub-trees. A limitation of this scheme is that it has not solved the contention problem of the nodes inside the same subtree branch. Farias et al. [31] proposed a queue-based algorithm for the Path Computation Element (PCE) which is a central scheduling entity to increase the reliability in industrial scenarios. Although centralized scheduling is optimal in term of performance, it requires a high signaling overhead due to the bidirectional communication between the network nodes and the central entity.

In the case of highly dynamic networks (e.g., unstable topologies and network traffic rate), distributed scheduling is a more favorable candidate. In this approach, a node exchanges its schedule information with the neighbor nodes and determines the optimal schedule. Soua et al. [23] introduced the Wave algorithm to schedule the nodes in successive waves. Each node with a transmittable packet is assigned a pair of a timeslot and a channel in each wave, and the first wave constitutes the (slot, channel) pattern. Each subsequent wave is an optimized version of the first wave, in which only the slots that contain the transmissions are repeated in the same order as that of the first wave. The authors of [30] used multiprotocol label switching to provide a distributed scheduling for the TSCH networks. The scheme exploits the RSVP-TE over the GMPLS protocol to transport the bandwidth requirement to each node along a track. The allocations of the slotOffset and the channelOffset are recursively conducted between a parent and its child nodes on the track. Accetture et al. [32] proposed the DeTAS, a decentralized version of the TASA, which is more lightweight than the above solutions and is implemented in real testbeds. In the DeTAS, when a node changes its bandwidth, it triggers a new schedule of the whole network by sending a scheduling request containing its bandwidth information to its parent. The scheduling request is propagated until it reaches the sink node. After its receipt of the request, the sink node divides the network into subtrees and computes a micro-schedule of each subtree based on either even-scheduled or odd-scheduled conventions. The slotOffsets are allocated for the nodes in each subtree through broadcast messages that are spread from the sink to the overall network. Moreover, the DeTAS assigns the channelOffsets to each node according to its hop rank in the network. This scheme ensures a perfect interleaving and efficiently avoids the buffer overflow. However, the scheduling is concentrated at the sink, and therefore a high signaling overhead is still a limitation.

2.2. Scheduling in 6TiSCH networks

The 6TiSCH WG has been recently discussing the development of a definition of a *minimal* schedule based on a static scheduling manner, that is pre-configured into nodes or learned by a node when it joins a network [33]. The cells in the schedule are shared in a Slotted ALOHA fashion that is used for both transmissions and the receptions of all kinds of frame. The 6TiSCH, however, only designs the static scheduling for specific situations such as the bootstrap phase or a fallback solution when the network fails.

For others purposes, a Scheduling Function is used to control (de) allocating cells between the neighbor nodes in a distributed manner. The IETF draft [27] presented the SF0 that is the default Scheduling Function of the 6TiSCH. The SF0 uses the *Bandwidth Estimation Algorithm* and *Allocation Policy* [27,34] to decide when a node should add/delete one or more cells to its neighbor. Moreover, the SF0 exploits

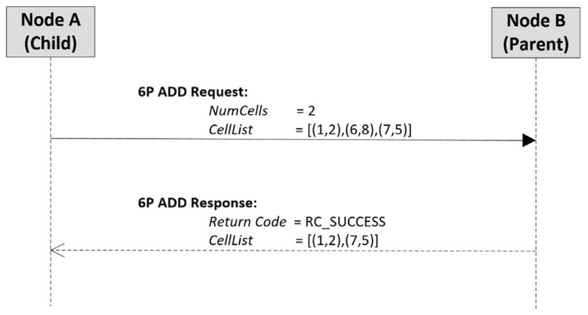

Fig. 1. Two-step 6P transaction.

a threshold-based mechanism to considerably reduce the add/delete operations when the bandwidth increases suddenly. Priteo et al. [28] introduced another Scheduling Function that is based on a PID industrial-control paradigm (PID-based SF). As previously mentioned, both of the Scheduling Function schemes apply the simplest possible policy that randomly selects the cells to add/delete.

Fig. 1 shows the example of the 6P with the two-step 6P transaction that is a complete DELETE or ADD negotiation between two neighbor nodes. Without a loss of the generality, the procedure occurs as follows:

(1) The Scheduling Function running on node A selects three candidate cells randomly. These cells are available at node A, and each cell is a random pair of a slotOffset and a channelOffset.
(2) Node A sends a 6P ADD Request to node B, indicating that it wishes to add two cells (*NumCells*), and it specifies the list of three candidate cells (*CellList*).
(3) The Scheduling Function running on node B checks how many cells in the *CellList* are available at node B. If the number of available cells is not smaller than the *NumCells*, node B will select two of these cells. Node B then sends back a 6P ADD Response to node A, indicating the cells it selected and a Success Response Code ($RC SUCCESS$). Contrarily, an Error Response Code ($RC ERROR$) is sent back.
(4) The result of this 6P transaction is that two cells from node A to node B have been added to the common schedule of both nodes.

In the 6TiSCH, a DELETE negotiation is similar with the ADD negotiation. However, it is always successful because the slotOffsets of all of the candidate cells in the *CellList* of node A are available at node B. With the random selection of the candidate cells, node A suffers from a high probability of the negotiation errors because the number of cells in the *CellList* that have slotOffsets available at node B may be smaller than the *NumCells*. Moreover, this approach also may cause collisions in the network because the allocated cells between node A and node B may also be used by other pairs of nodes. To mitigate the collision effects, the Scheduling Function tries to re-actively relocate the colliding cells with the *6top Housekeeping* function that can monitor the performance and utilization of each cell and possibly relocate the cell within the schedule in the case where the collisions on that cell are detected [28,29]. In this paper, the aim is to proactively avoid the collisions and to use the *6top Housekeeping* as an additional function to mitigate further interferences.

To reduce the end-to-end latency in a multi-hop path, Chang et al. [35] proposed a daisy chain timeslot selection mechanism for the SF0, called the “Low Latency Scheduling Function” (LLSF). Yet, without an additional overhead, the LLSF may only work efficiently in a specific linear-network topology in which each node has no more than one child node. In addition, the collision issue has not yet been solved with the LLSF. With the same goal, the authors of [36] presented a localized scheduling technique. In particular, the technique divides the network into stratums and the slotframe into blocks. The blocks from the contiguous stratums are consecutive, and the size of each block follows a given regulation. Each stratum is constituted of nodes with the same hop rank. All of the nodes in a stratum must randomly select the slotOffets for the allocated cells in an equivalent block. In this way, a packet is delivered before the end of the slotframe. Moreover, the collisions are also reduced because the nodes at each hop use different slotOffsets with other hops without a consideration of how the channelOffsets can be selected optimally. A limitation of the scheme is that the negotiation errors may still happen with the selection of the slotOffsets at each block. Furthermore, the given division method of the technique is not flexible in networks where the nodes are distributed unpredictably or unstable traffic exists.

3. Proposed mechanism

To reduce the negotiation errors as well as the collisions, a simple and novel distributed cell-selection mechanism is proposed for the Scheduling Function. The proposed mechanism consists of the two following main processes: the slotOffset and channelOffset selections.

The proposed cell-selection is designed for the 6TiSCH networks that are formed by the RPL in a single Destination Oriented Directed Acyclic Graph (DODAG). Each DODAG topology is coordinated by the sink node. In addition, the mechanism works based on the following realistic assumptions:

- The network supports a multi-point-to-point traffic.
- In the network, only two kinds of node are presented as follows: the sink node and the source node. A source node only generates packets and forwards packets toward the sink.
- A source node uses only its preferred parent to route its packets; therefore, it has to negotiate a set of cells with its parent.
- Each node has a limited queue length, denoted by L_{queue}, and a slotframe length, denoted by $L_{slotframe}$, with the condition that $L_{queue} < L_{slotframe}$. Each node can generate a number of packets simultaneously, which must not be more than the L_{queue}. Note that due to the constrained resources of the 6TiSCH networks, the L_{queue} must be configured properly. In the paper, we use $L_{queue} = 10$.

3.1. SlotOffset selection

The proposal here is the division of the slotframe into portions, as follows: the number of portions is denoted by N with $N = L_{slotframe}/L_{queue}$; therefore, the portion length is equal to L_{queue}. Each portion is characterized by a density value that is conceptually the ratio between the number of unavailable slotOffsets that have been assigned for the communication in the portion and the length of that portion. A Scheduling Function can monitor the density of each portion through its *Statistical Monitoring* function.

Fig. 2 shows an example that visually illustrates the density of five portions in the slotframe of two neighbor nodes. The black cells indicate the unavailable slotOffsets that are currently used for the communication. The other cells indicate the available slotOffsets. The density of each portion is evidently different when they are compared with one another. As a result, the probability of the selection of an available slotOffset in each portion is also different. Based on this observation, the following slotOffset selection algorithm has been designed.

When a node (i.e., node A) wants to add a number of cells to its neighbor (i.e., node B), node A first sends a density list that contains the density values of its N portions to node B. Based on node A's density list, the Scheduling Function running on node B uses the proposed Best Portion Selection algorithm (BPS), illustrated in Algorithm 1, to determine the portion with the lowest density between the two nodes. The Scheduling Function then selects the slotOffsets for the candidate cells in this portion. The channelOffset selection process is discussed in the next subsection.

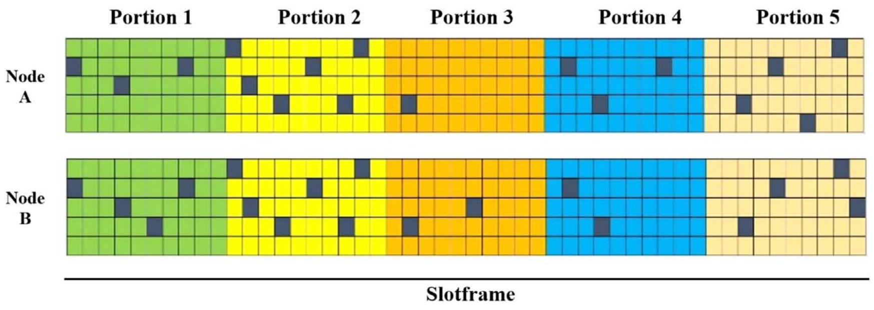

Fig. 2. Density status of the slotframe divided into five portions of two neighbor nodes.

Algorithm 1. Best portion selection algorithm.

```
   Input: nbDensityList[N]
   Output: bestPortionID
   Initialization:
   $id \leftarrow 0,\ aveDensityList[N] \leftarrow 0,\ myDensityList[N] \leftarrow 0$
   Function: SelectBestPortion
1  for id in N do
2      $myDensity[id] \leftarrow statMonitorDensity(id)$
3      $aveDensity[id] \leftarrow (myDensityList[id] + nbDensityList[id])/2$
4  end
5  $bestPortionID \leftarrow MIN(aveDensityList[N])$
6  return bestPortionID
```

.

The BPS algorithm running on node B is described in Algorithm 1. In detail, node B's Scheduling Function first collects the density values of its N portions by using the *statMonitorDensity* function (i.e, *Statistical Monitoring* function). It then calculates the average density value for each portion of B and the corresponding portion of A. The *MIN* function is used to return the portion *ID* of the portion that has the lowest average density between the two nodes. In this way, the algorithm enables the neighbor nodes to select the slotOffsets in the best available portion of the slotframe to reduce the negotiation errors.

3.2. ChannelOffset selection

It is important to note that the pairs of nodes that are more than a two-hop rank away in a cluster-tree may transmit simultaneously due to a low interference probability [30]. Inspired by this observation, in the proposed channelOffset selection, we implement the channelOffset assignment method introduced in [30], that is on the basis of an opportunistic transmission approach [37] and originally is designed by default for the label switching scheduling algorithm, for the Scheduling Function. Following the channelOffset assignment method, a node does not transmit packets on the same channel with its predecessor that is a two-hop rank away. To implement and make the method fit with the operations of the Scheduling Function, new procedures are presented here as follows:

- Each node will only use a constant pair of the selected channelOffsets - *TX* and *RX* for the transmission and reception of the packets, respectively. The channelOffset selection processes are presented below.
- Each node stores its *channelOffset-selection information* that consists of the following three parameters: *TX*, *RX*, and *StatusFlag*. Once the *StatusFlag* of a node is equal to *True*, it indicates that the node has already selected its channelOffsets. Contrarily, the *StatusFlag* is set to *False* when the node does not have its own selected channelOffsets yet.
- At the beginning of the network life, the initial channelOffsets of each node are set to 0, *TX*=*RX*=0. The *StatusFlag* is set to *False* in the case of the source nodes and *True* in the case of the sink. This means that, by default, the sink node has its selected channelOffsets with *TX*=*RX*=0.
- To reduce the impact of the collisions, the following rule is set: a source node will not generate and forward packets if its channelOffsets have not yet been selected.
- If a node receives a request message from its child node, it will send back its *channelOffset-selection information* as an attachment of the response message to the child node. Based on the received information, the child node uses the ChannelOffset Selection algorithm (CS) that is described below to select its channels. This process is repeated until the selected transmission channelOffsets have been applied to all of the nodes in the network.

Algorithm 2. Channel selection algorithm.

```
   Input: nbStatusFlag, nbTX, nbRX
   Initialization: $myTX \leftarrow 0,\ myRX \leftarrow 0,\ myStatusFlag \leftarrow False$
   Function: SelectChannel
1  if $myStatusFlag == False\ and\ nbStatusFlag == True$ then
2      $myTX \leftarrow nbRX$
3      $myAvailableChannels.\ remove(nbTX)$
4      $myRX \leftarrow random(myAvailableChannels)$
5      $myStatusFlag \leftarrow True$
6  end
```

The CS algorithm is described in Algorithm 2. The channelOffset selection process at a child node is enabled when its channelOffsets have not yet been selected and the selected channelOffsets for its parent have been determined. The *RX* of the parent node is assigned as the *TX* of the child node. The child node randomly picks a valid channelOffset, with the exception of the parent node *TX*, for its *RX* and turns its *StatusFlag* to *True*.

Fig. 3 shows an example of the channelOffset selection process in which the packets (1), (3), (5), and (7) are the request messages, and the packets (2), (4), (6), and (8) are the response messages containing the *channelOffset-selection information*. In the proposal, the 6P ADD Request and 6P ADD Response of the 6P transaction are used as the request and response messages, respectively. This adoption will be discussed in the next subsection.

The aim of the proposed channelOffset selection mechanism is the mitigation of the interferences between the nodes that are in the same chain of a network. The nodes belonging to different chains may select the same channelOffsets, thereby potentially causing collisions, as illustrated in Fig. 4 (collision between node n_3 and node n_6). To address this problem, the *6top Housekeeping* function is used.

3.3. Enhanced 6P protocol

An enhanced 6P (E6P) to plug the proposed cell-selection mechanism directly into the Scheduling Function is now proposed. The E6P with the three-step 6P transaction is shown in Fig. 5 and is described as follows:

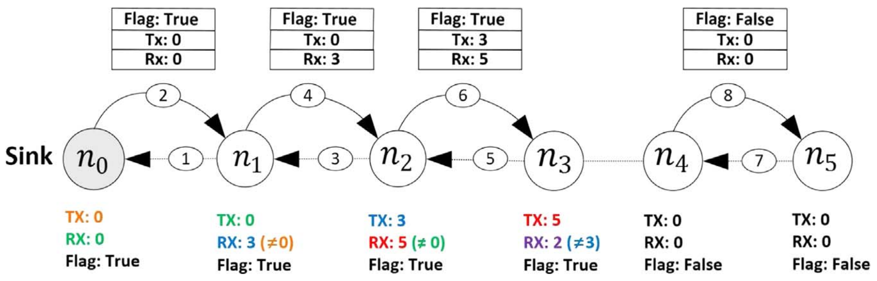

Fig. 3. Proposed channelOffset selection process between neighbor nodes in a same chain of a network.

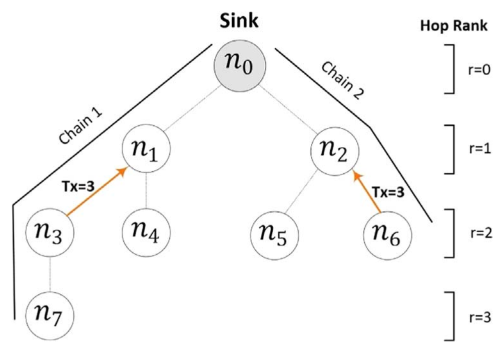

Fig. 4. Collision between two nodes belonging to different chains.

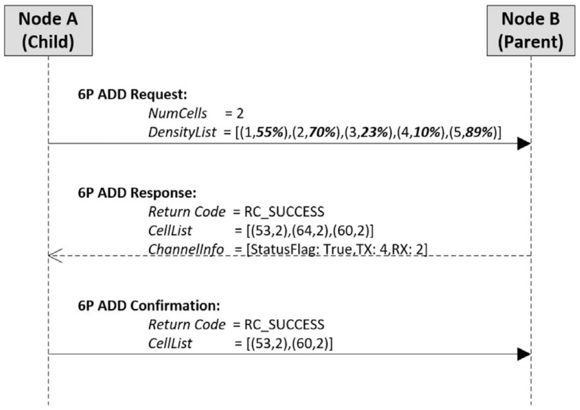
Node A (Child)
Node B (Parent)
6P ADD Request:
NumCells = 2
DensityList = [(1,55%),(2,70%),(3,23%),(4,10%),(5,89%)]
6P ADD Response:
Return Code = RC_SUCCESS
CellList = [(53,2),(64,2),(60,2)]
ChannelInfo = [StatusFlag: True,TX: 4,RX: 2]
6P ADD Confirmation:
Return Code = RC_SUCCESS
CellList = [(53,2),(60,2)]

Fig. 5. Enhanced 6P with three-step 6P transaction.

(1) The Scheduling Function running on node A first sends a 6P ADD Request, indicating the density list (*DensityList*) of node A and the number of requisite cells (*NumCells*) to node B. The *DensityList* contains the *IDs* and density values of node A's portions.
(2) The Scheduling Function running on node B uses the BPS algorithm and node A's *DensityList* to determine a portion with the lowest density between the two nodes. It selects three candidate cells in this portion with the channelOffset that is equal to its *RX*. Then, node B sends back a 6P ADD Response to node A, indicating the candidate cells (*CellList*) and its *channelOffset-selection information* (*ChannelInfo*).
(3) The Scheduling Function running on node A uses the CS algorithm based on node B's *ChannelInfo* to select its *TX* and *RX* and saves them into the buffer. It concurrently chooses two of the available cells in the *CellList* and sends back a 6P ADD Confirmation to node B with the cells it has chosen.
(4) The result of the 6P transaction is that two of the cells from A to B have been added to the schedule of both of the nodes A and B.

Note that the E6P may incur errors when the channelOffsets of node B have not been selected, or the candidate cells of node B cannot sufficiently meet the number of requisite cells from node A for the allocation. As mentioned previously, if a source node has not selected its channelOffsets yet, it cannot generate and forward any packets. As a result, a bandwidth requirement for the Scheduling Function that can enable a 6P transaction to activate the channelOffset selection process at each node is absent. In this work, at the bootstrap phase, it is supposed that the Scheduling Function running on each node forces the addition of at least one cell to its neighbor.

4. Performance evaluation

This section presents the extensive performance evaluation of the proposed mechanism through the simulations and experimental test-beds.

4.1. Simulation results

For the simulations, the 6TiSCH simulator [38] which is an open-source simulator written in Python and developed by members of the 6TiSCH WG, is employed. Currently, the simulator is implemented by default with the IEEE 802.15.4e, RPL, and 6TiSCH-related components such as 6top, 6P, and SF0. The physical model used in the 6TiSCH simulator is the Pister Hack model with the collision considerations [39]. In the simulation implementation,[1] we replace the original 6P with the E6P and apply the proposed cell-selection mechanism to create a new SF0 version called the "enhanced SF0" (ESF0). The performances of the ESF0 with the SF0 and the LLSF, the two state-of-the-art Scheduling Function schemes, are then compared.

The detailed simulation parameters are summarized in Table 1. The simulations are conducted based on an industrial environment scenario with a 1 km×1 km deployment area without any external interference such as an oil-refinery factory [40] or chemical plants [41], where the sensor nodes periodically report sensing data to a coordinator (i.e., the sink). Each node has at least three neighbors with links that comprise a packet delivery ratio (PDR) ≥50%. The minimum acceptable RSSI value that allows for a packet reception is −97 dBm, while the maximum number of MAC retries is set to 5. The battery capacity used by each node is 2200 mAh. The timeslot duration is 10 ms, and a slotframe contains 101 timeslots. The threshold value is set to 0. The application on each node generates a data packet periodically with an interval of 2 s. A node can hold up to 10 packets on its queue. The simulation results are presented based on an average of 100 simulation runs with a 95% confidence interval.

Fig. 6 shows how the network density affects the 6P negotiation-error ratio which is the average ratio between the number of 6P transaction errors and the total 6P transactions during a slotframe cycle. The number of nodes is increased from 40 to 100. The 16 available channels are used for the scheduling. The increasing of the network density causes an increase of the negotiation-error ratio, from 2.5% to 6% in the SF0 case, and from 0% to 1.8% in the ESF0 case. The reason here is the decrease of the number of available slots in the slotframe when each node has a higher number of neighbor nodes. The

[1] https://github.com/duythang/esf0_simulator.

Table 1
Simulation parameters.

Parameter	Value
Area	square 1 km×1 km
Deployment constraint	≥3 neighbors with PDR 50%
Radio sensitivity	−97 dBm
Radio range	100 m
Number of runs per sample	100
Number of cycles per run	200
Max. MAC retries	5
Queue length	10
Timeslot duration	10 ms
Slotframe length	101 timeslots
Battery capacity	2200 mAh
App traffic period	2 s
6top housekeeping period	1 s
OTF threshold	0
OTF housekeeping period	5 s
RPL parents	1
RPL DIO period	1 s

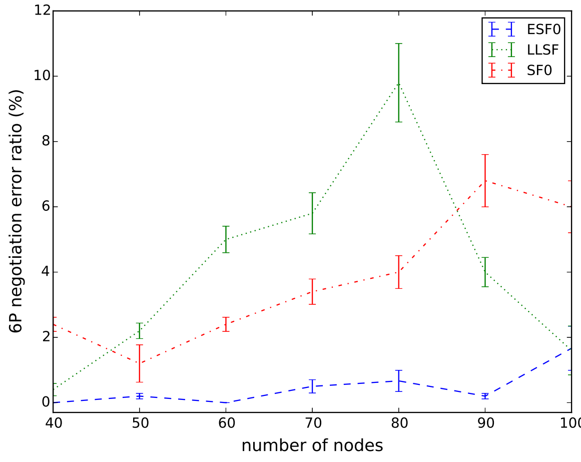

Fig. 6. Average 6P negotiation-error ratio per slotframe cycle versus the network density.

negotiation-error ratio of the LLSF fluctuates between 0.8% and 9.8%. The proposed mechanism, the ESF0, outperforms the LLSF and the SF0 significantly and maintains the negotiation-error ratio below 2% at all of the network densities.

Fig. 7 presents the number of colliding cells under the different network densities. A cell is called a “colliding cell” when the cell is used by more than one pair of nodes, thereby creating the potential to cause collisions in the data transmission. The number of colliding cells is

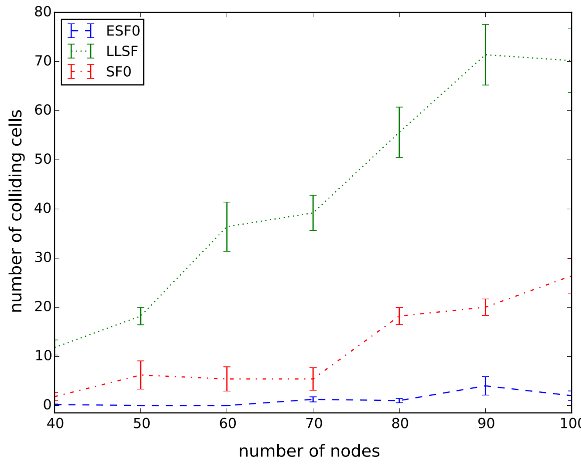

Fig. 7. Average number of colliding cells per slotframe cycle versus the network density.

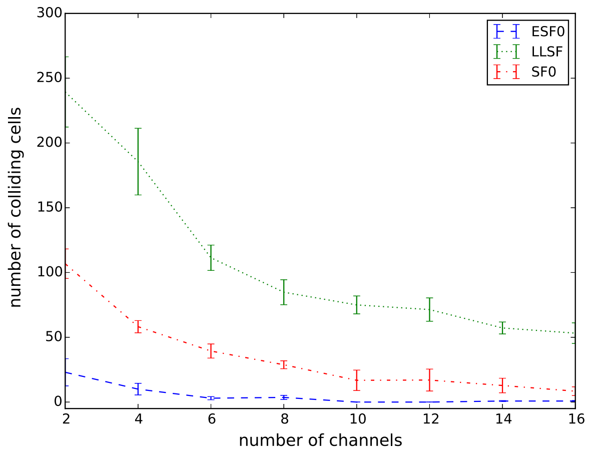

Fig. 8. Average number of colliding cells per slotframe cycle versus the number of available channels.

proportional to the network density. The proposed cell-selection mechanism helps the ESF0 to achieve a lower number of colliding cells compared with the other schemes. In networks with less than 60 nodes, the ESF0 witnesses no colliding cells. For the higher density networks, the recorded colliding cells are few in number (i.e., lower than four). Without a consideration of the mitigation of the collisions, the LLSF suffers from the highest number of colliding cells at all of the network densities compared with the ESF0 and the SF0.

Fig. 8 depicts the way that the number of colliding cells varies regarding the number of available channels for the scheduling in an 80-node network. It is evident that the number of colliding cells is inversely proportional to the number of channels. As with the results of Fig. 7, the ESF0 achieves the best performance.

The impact of the network density and the number of available channels on the end-to-end latency is shown in Figs. 9a and b, respectively. With the random network topologies, the LLSF shows a long end-to-end latency. The ESF0 shows the lowest latency and maintains the latency at less than 150 timeslots in all of the cases. This is because of its significant reduction of the negotiation errors, as well as the collisions, compared with the other schemes.

Fig. 10 shows the number of dropped packets over the network life. In this simulation, the same setting that is used for Fig. 8, with the 16 available channels, is employed. At the beginning of the network life (i.e., the first 50 cycles), the nodes discover one another, start forming a multi-hop structure and add cells to their neighbors. During this phase, a great number of dropped packets are presented. The reason here is the inability of the scheduling operation to meet a sufficient quantity of allocated cells in a timely manner for the packet transmissions due to the negotiation errors and the collision effects. The number of dropped packets decreases to a steady state as the initial phase finishes, and most of the requisite cells are allocated. The figure also shows how the ESF0 prevents the dropped packets more effectively than the SF0 and LLSF.

Fig. 11 shows the average number of signaling packets that are used for the scheduling per slotframe cycle under the various network densities. As expected, the number of signaling packets linearly increases when the number of nodes grows. Although the ESF0 uses three signaling packets for the 6P transaction instead of two, the ESF0 experiences a lesser signaling overhead than the SF0. This is because the nodes running the SF0 suffer from more collisions, which leads to the use of more *6top Housekeeping* signaling packets for the relocation of the colliding cells. This is similar to the case of the LLSF, where the LLSF witnesses a higher signaling overhead compared with the SF0.

Fig. 12 shows a comparison of the energy consumptions of the SF0, LLSF, and ESF0 according to the estimated battery lifetime. In general, the battery lifetime decreases significantly when the number of nodes increases. The greater the number of nodes, the higher the number of

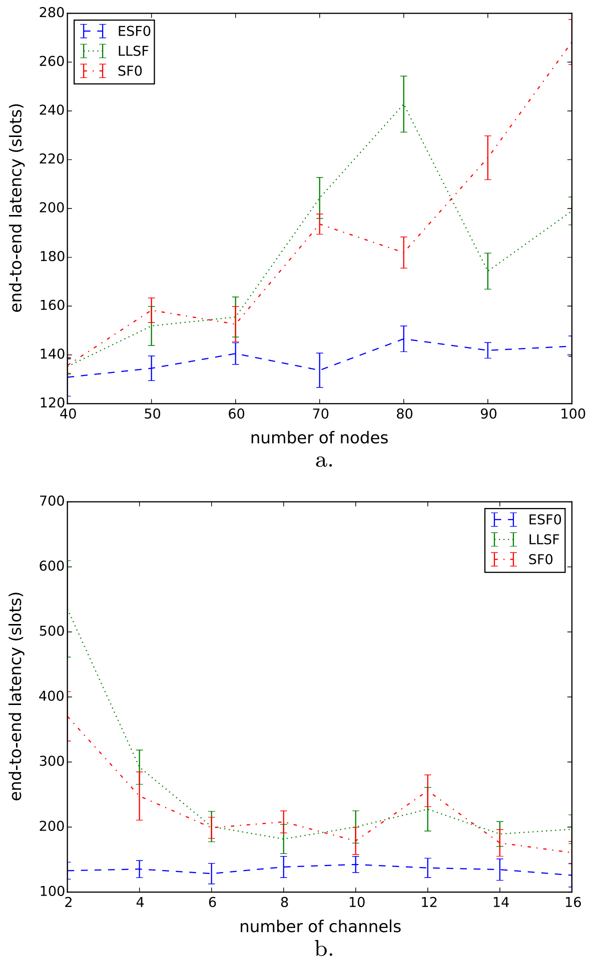

Fig. 9. Average end-to-end latency versus the network density (a) and number of available channels (b).

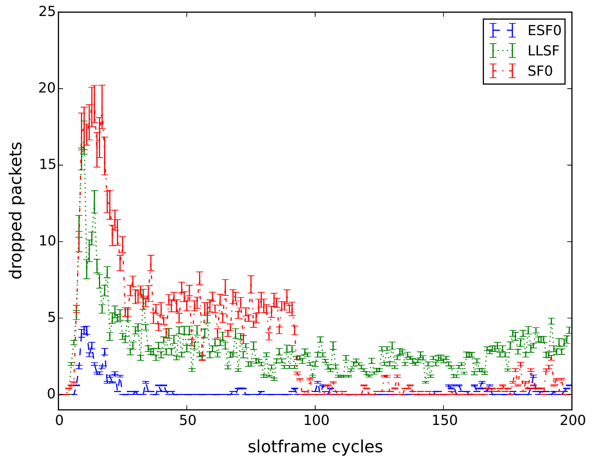

Fig. 10. Number of dropped packets over the network life.

packet transmissions. Consequently, the nodes also consume more energy. The figure shows that the energy consumptions of the nodes running the SF0 and the LLSF are significantly higher than that running the ESF0. The reason here is the high negotiation-error ratio and collisions suffered by the SF0 and the LLSF, which cause a higher number of re-transmissions at the MAC layer compared with the ESF0.

4.2. Implementation and experimental results

Fig. 13 describes the overall software architecture of the OpenWSN

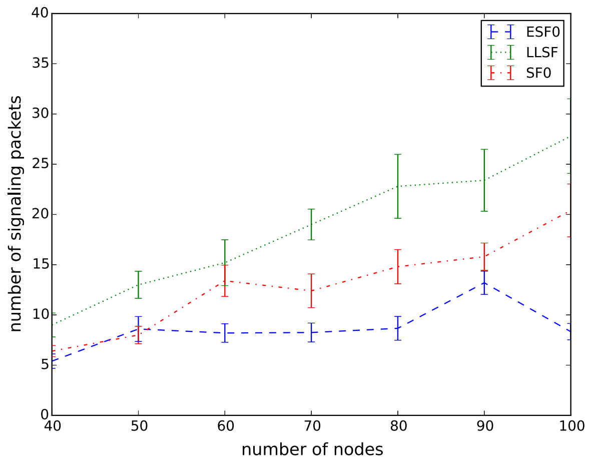

Fig. 11. Average number of signaling packets per slotframe cycle versus the network density.

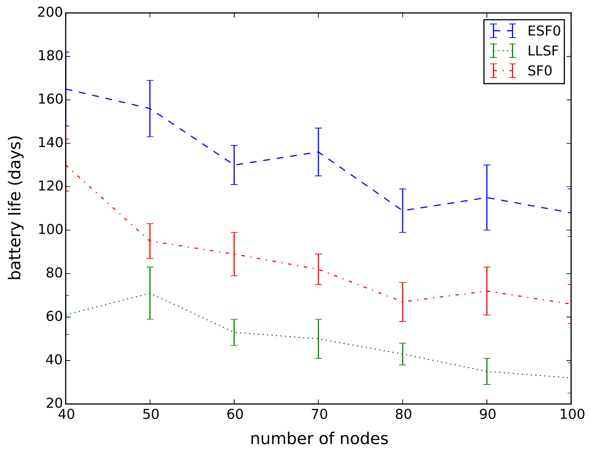

Fig. 12. Average battery life of each network node after 200 slotframe cycles versus the network density.

project [42], which is also used in the present implementation of the ESF0. OpenWSN is the open-source implementation of a protocol stack consisting of IoT standards and is applied on various hardware platforms. The ESF0[2] is implemented based on the SF0 model presented in [27] and [34]. The location of the ESF0 is inside the MAC high layer, and the ESF0 is divided into three logical blocks, as follows: (1) *Bandwidth Estimation Algorithm* uses the statistical information from the queue, the neighbor table, and the schedule to estimate the node's bandwidth requirement. (2) *Allocation Policy* is the set of rules used by the Scheduling Function to decide when a node should add/delete one or more cells to its neighbor to satisfy the bandwidth requirement. (3) *Cell Selection* controls the selection of the cells following the proposed selection mechanism. In the implementation, the 6P is also replaced by the E6P.

The testbeds are deployed on the Tmote-sky nodes and in an indoor environment. Each result that is presented in this section is the average of 30 experiments. The following two main network topologies are considered: star and linear topologies. In all of the scenarios, the timeslot duration is set to 15 ms. In the slotframe of each node, two shared slots with the channelOffset of 0 are pre-configured for both the transmission and the reception of the signaling packets, such as the Enhanced Beacons (EB) advertisement, routing, and 6P transaction. Three serial slots are used for the communications with the serial port, and the queue length is equal to 10. The measured range between the two nodes is 1 m. To accurately test the performance, the sending of the

[2] https://github.com/duythang/esf0_openwsn

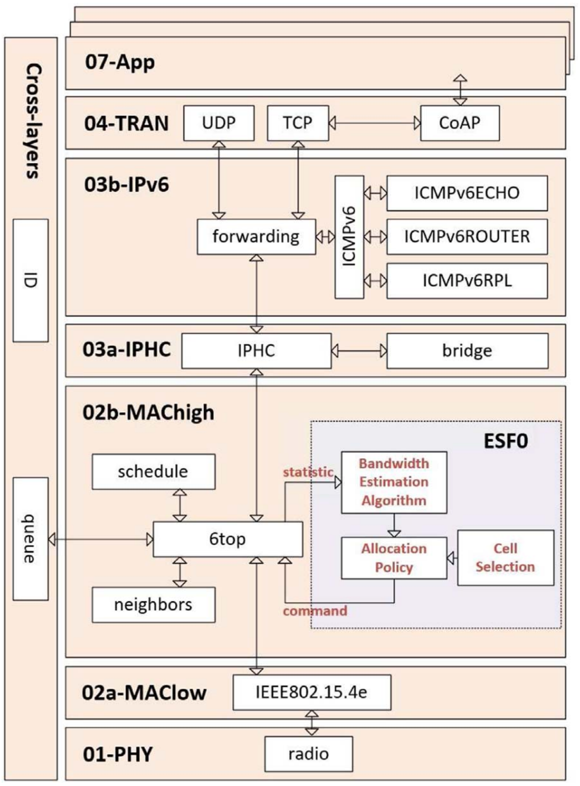

Fig. 13. ESF0 component inside the OpenWSN stack.

data packets that are generated by the source nodes is forced on the allocated cells, and the other packets are only sent through the shared cells.

The first testbed with the star-network topology is illustrated in Fig. 14. In this setting, the aim is the measurement of the negotiation-error ratio of the proposed slotOffset selection mechanism under various network densities. The slotframe is configured with 35 slots. The number of neighbor nodes (i.e, source nodes) is varied from 2 to 6 nodes to investigate the impact of the network density on the performance of the ESF0. Each source node generates a bursty traffic with the recording of five packets every 5 s.

The obtained results are shown in Fig. 15. As expected, in both the SF0 and the ESF0, the error ratio increases as the number of neighbor nodes grows. In the cases with two and three neighbor nodes, the best performance is provided by the ESF0, where the negotiation errors are almost nonexistent.

In the second testbed, a linear-network topology with five hops is deployed, as shown in Fig. 16. This scenario aims to measure the PDR of the proposed mechanism under a high-pressure environment (i.e., high collision probability). By using the linear topology, the absence of the impact of the negotiation errors during the network scheduling is also guaranteed. The collision probability is increased through a

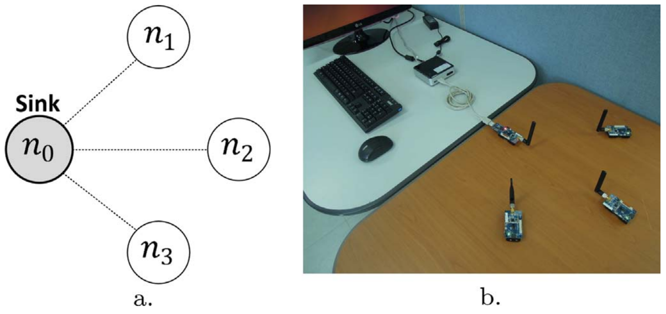

Fig. 14. Star topology (a) and test-bed (b).

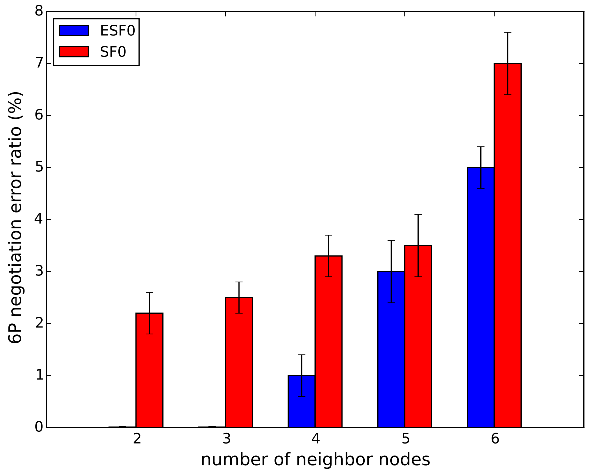

Fig. 15. 6P negotiation-error ratio of the different number of neighbor nodes in the SF0 and ESF0.

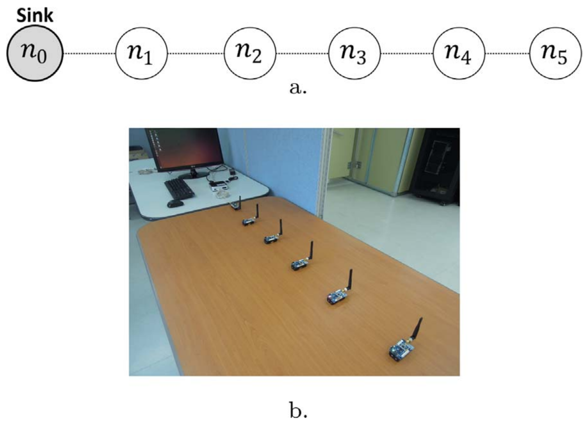

Fig. 16. Linear topology (a) and test-bed (b).

reduction of the slotframe length to 15 slots and the use of only three of the available channels for the scheduling. Each source node runs the *uinject* application [43] to send the dummy data toward the sink with a packet interval of 2 s. The number of retransmissions is set to 1. The end-to-end PDR is estimated at each hop rank.

The average end-to-end PDR values at the hop ranks of the linear-network topology are shown in Fig. 17. In detail, under the collision effects, the SF0 experiences a high number of dropped packets at all of the hop ranks. By implementing the proposed cell-selection mechanism, the ESF0 helps in the significant improvement of the PDR of the

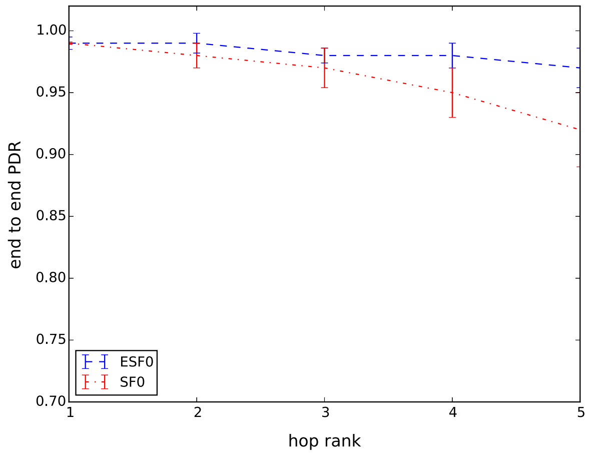

Fig. 17. End-to-end PDR versus the hop rank.

network compared with the SF0. The network running with the ESF0 achieves a high PDR (i.e., above 97%) at every hop.

In fact, the testbeds that use other topologies (i.e., random network deployment) with a higher number of nodes, are also conducted. However, the results show that the performance depends mainly on the number of neighbor nodes of a node in the case of the negotiation-error ratio, and on the hop rank of that node in the case of the PDR. To keep the evaluation tractable, the results from the simple star and linear topologies are, therefore, solely presented to show the performances of the proposed mechanism in relation to the network density (i.e., the number of neighbor nodes) and hop rank parameters, respectively.

The performance trends that are obtained from the experiments show a strong coherence with the simulated results. In a planned future work, the proposed ESF0 implementation will be performed in real industrial fields with a greater number of nodes under different external interference levels. This proposal will also be sent to the 6TiSCH WG for discussion.

5. Conclusions

This article presents an effort for the enhancement of the efficiency of the Scheduling Function in the 6TiSCH through a lightweight distributed cell-selection mechanism. In detail, an efficient slotOffset selection method is introduced with the BPS algorithm to reduce the scheduling negotiation errors. The proposed channelOffset selection ensures that the nodes are not affected by the collisions from the other nodes in the two-hop away domain. The E6P protocol is also designed and implemented to plug the proposed cell-selection mechanism directly into the Scheduling Function. The performance of the proposed mechanism is extensively evaluated using simulations and experiments under various scenarios. The results show that the present study's proposal outperforms the current 6TiSCH Scheduling Function schemes and helps in the reduction of the scheduling errors and collisions, as well as the energy consumption, to improve the overall network life. For a future work, the proposed ESF0 implementation will be tested in real industrial fields with a greater number of nodes under different external interference levels.

Acknowledgements

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-2016-H8501-16-1008) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] L. Atzori, et al., The internet of things: a survey, J. Comput. Netw. 54 (2010) 2787–2805. http://dx.doi.org/10.1016/j.comnet.2010.05.010.
[2] M. Chen, et al., A survey of recent developments in home m2m networks, IEEE Commun. Surv. Tut. 16 (2013) 98–114. http://dx.doi.org/10.1109/SURV.2013.110113.00249.
[3] J. Zhao, et al., Issues on convergecast scheduling in industrial wireless sensor networks, WiCOM, IEEE, 2014, pp. 485–490. http://dx.doi.org/10.1049/ic.2014.0150.
[4] J. Wan, et al., Software-defined industrial internet of things in the context of industry 4.0, IEEE Sens. J. 16 (2016) 7373–7380. http://dx.doi.org/10.1109/JSEN.2016.2565621.
[5] H. Zhang, et al., Time-optimal convergecast with separated packet copying: scheduling policies and performance, IEEE Trans. Veh. Technol. 64 (2014) 793–803. http://dx.doi.org/10.1109/TVT.2014.2322141.
[6] Thang, et al., A rapid joining scheme based on fuzzy logic for highly dynamic ieee 802.15.4e time-slotted channel hopping networks, Int. J. Distrib. Sens. Netw. 12 (2016) 1–10. http://dx.doi.org/10.1177/1550147716659424.
[7] Wirelesshart specification 75: Tdma data-link layer, Hart communication foundation std., rev. 1.1, hcf spec-75, 2008.
[8] Isa-100.11a-2011: Wireless systems for industrial automation - process control and related applications, International society of automation (isa) std., 2011.
[9] Ieee std. 802.15.4, part 15.4: Low-rate wireless personal area networks (lr-wpans), Standard for information technology std., 2011.
[10] 802.15.4e-2012: Ieee standard for local and metropolitan area networks - part 15. 4: Low-rate wireless personal area networks (lr-wpans) amendment 1: Mac sublayer, Ieee std., 2012.
[11] Ieee std. 802.15.4: Ieee standard for low-rate wireless networks, Standard for information technology std. 2015. ⟨http://standards.ieee.org/getieee802/download/802.15.4-2015.pdf⟩.
[12] Thang, K. YoungHan, An efficient joining scheme in ieee 802.15.4e, ICTC, IEEE, 2015, pp. 226–229. ⟨http://dx.doi.org/10.1109/ICTC.2015.7354534⟩.
[13] T. Watteyne, et al., Reliability through frequency diversity: why channel hopping makes sense, PE-WASUN, ACM, 2009, pp. 116–123. ⟨http://dx.doi.org/10.1145/1641876.1641898⟩.
[14] 6tisch working group, ⟨https://datatracker.ietf.org/wg/6tisch/charter/⟩, (Accessed 8 June 2016).
[15] 6tisch homepage, https://bitbucket.org/6tisch/⟩, (Accessed 8 June 2016).
[16] Z. Shelby, et al., The constrained application protocol (coap), RFC 7252 (2014) ⟨http://www.rfc-editor.org/rfc/rfc7252⟩).
[17] Z. Shelby, et al., Neighbor discovery optimization for ipv6 over low-power wireless personal area networks (6lowpans), RFC 6775 (2012) ⟨https://tools.ietf.org/html/rfc6775⟩.
[18] T. Winter, et al., Rpl: Ipv6 routing protocol for low power and lossy networks, RFC 6550 (2012). ⟨https://tools.ietf.org/html/rfc6550⟩.
[19] M. Palattella, et al., Internet of Things: Challenges and Opportunities, Springer, 2014, pp. 111–141 (Ch. 6).
[20] D. Dujovne, et al., 6tisch: deterministic ip-enabled industrial internet (of things), IEEE Commun. Mag. 52 (2014) 36–41. http://dx.doi.org/10.1109/MCOM.2014.6979984.
[21] M. Palattel, et al., Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15.4e networks, PIMRC, IEEE, 2012, pp. 327–332. ⟨http://dx.doi.org/10.1109/PIMRC.2012.6362805⟩.
[22] Y. Wu, et al., Realistic and efficient multi-channel communications in wireless sensor networks, INFOCOM, IEEE, 2008. ⟨http://dx.doi.org/10.1109/INFOCOM.2008.175⟩.
[23] R. Soua, et al., A distributed joint channel and slot assignment for convergecast in wireless sensor networks, NTMS, IEEE, 2014, pp. 1–5. ⟨http://dx.doi.org/10.1109/NTMS.2014.6813995⟩.
[24] R. Soua, et al., Disca: a distributed scheduling for convergecast multichannel wireless sensor networks, IM, IEEE, 2015, pp. 156–164. ⟨http://dx.doi.org/10.1109/INM.2015.7140288⟩.
[25] Q. Wang, X. Vilajosana, 6tisch operation sublayer (6top), Ietf draft, 2015. ⟨https://tools.ietf.org/html/draft-wang-6tisch-6top-sublayer-04⟩.
[26] M. Palattel, et al., Terminology in ipv6 over the tsch mode of ieee 802.15.4e (work in progress), Ietf draft (2016). URL ⟨https://tools.ietf.org/html/draft-ietf-6tisch-terminology-07⟩.
[27] D. Dujovne, et al., 6tisch 6top scheduling function zero (sf0) (work in progress), Ietf draft, 2017. ⟨https://tools.ietf.org/html/draft-dujovne-6tisch-6top-sf0-03⟩.
[28] M. Domingo-Prieto, et al., Distributed pid-based scheduling for 6tisch networks, IEEE Commun. Lett. 20 (2016) 1006–1009. http://dx.doi.org/10.1109/LCOMM.2016.2546880.
[29] K. Muraoka, et al., Simple distributed scheduling with collision detection in tsch networks, IEEE Sens. J. PP (2016) 1. http://dx.doi.org/10.1109/JSEN.2016.2572961.
[30] A. Morell, et al., Label switching over ieee802.15.4e networks, Trans. Emerg. Telecommun. Technol. 24 (2013) 458–475. http://dx.doi.org/10.1002/ett.2650.
[31] A. Farias, D. Dujovne, A queue-based scheduling algorithm for pce-enabled industrial internet of things networks, CASE, IEEE, 2015, pp. 31–36. http://dx.doi.org/10.1109/SASE-CASE.2015.7295844.
[32] N. Accettura, et al., Decentralized traffic aware scheduling in 6tisch networks: design and experimental evaluation, IEEE Internet Things J. 2 (2015) 455–470. http://dx.doi.org/10.1109/JIOT.2015.2476915.
[33] X. Vilajosana, K. Pister, Minimal 6tisch configuration (work in progress), Ietf draft (2016). ⟨https://tools.ietf.org/pdf/draft-ietf-6tisch-minimal-15⟩.
[34] M. Palattella, et al., On-the-fly bandwidth reservation for 6tisch wireless industrial networks, IEEE Sens. J. 16 (2016) 550–560. http://dx.doi.org/10.1109/JSEN.2015.2480886.
[35] T. Chang, et al., Llsf: Low latency scheduling function for 6tisch networks, DCOSS, IEEE, 2016, pp. 1–4. http://dx.doi.org/10.1109/DCOSS.2016.10.
[36] I. Hosni, et al., Localized scheduling for end-to-end delay constrained low power lossy networks with 6tisch, ISCC, IEEE, 2016, pp. 1–6. http://dx.doi.org/10.1109/ISCC.2016.7543789.
[37] A.S. A, Y. Hong, Opportunistic large arrays: cooperative transmissions in wireless multihop ad hoc networks to reach far distances, IEEE Transactions on Signal Processing 16 (2016) 550–560. http://dx.doi.org/10.1109/TSP.2003.814519.
[38] 6tisch simulator, ⟨https://bitbucket.org/6tisch/simulator/⟩, (Accessed 8 June 2016).
[39] L. Hanh-Phuc, et al., Energy-aware routing in wireless sensor networks with adaptive energy-slope control, EE290Q-2, 2009, pp. 1–6.
[40] S. Zats, Wireless sensor networks scaling and deployment in industrial automation, master's thesis, University of California, Berkeley, 2010.
[41] F. Chraim, et al., Wireless gas leak detection and localization, IEEE Trans. Indus. Inf. 12 (2016) 768–779. http://dx.doi.org/10.1109/TII.2015.2397879.
[42] T. Watteyne, et al., Openwsn: a standards-based lowpower wireless development environment, Trans. Emerg. Telecommun. Technol. 23 (2012) 480–493. http://dx.doi.org/10.1002/ett.2558.
[43] Openwsn source code, ⟨https://github.com/openwsn-berkeley/openwsn-fw⟩, (Accessed 30 December 2016).