T^2AS: Topology/Traffic Aware Scheduling to Optimize the End-to-end Delay in IEEE802.154e-TSCH Networks

Erfan Mozaffari Ahrar, Mohammad Nassiri*

Computer Department, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran.
e.mozaffari@alumni.basu.ac.ir, m.nassiri@basu.ac.ir
*Corresponding author
Received: 2020-01-07
Accepted: 2020-09-16

Abstract
The Time Synchronized Channel Hopping (TSCH) mode of IEEE 802.15.4e has been widely used as an access method for the industrial Internet of Things (IoT). It permits to overcome the performance limits of 802.15.4 standard in such networks. It provides bounded latency and increased network capacity while mitigating the effects of interference and multipath fading. In this paper, we tackle two critical concerns of industrial networks, namely end-to-end reliability and delay by proposing two centralized scheduling mechanisms; First, the Height-based Scheduling (HS) that computes the schedule only based on the network topology. Second, T^2AS, which takes into account both traffic demand and network topology to calculate the schedule. The later mechanism uses a composite weighting function that allows scheduling links with more load and longer distance from the root in earlier timeslots. This prioritizes the flows with more traffic to be scheduled earlier. Both algorithms provide subsequential scheduling for multi-hop scenarios. Simulation results, obtained from the OpenWSN emulator, particularly confirm the efficiency of T^2AS in terms of reliability and end-to-end latency. More precisely, it guarantees a reliability of more than 99% for all network sizes. Furthermore, T^2AS provides a noticeable bounded delay by delivering data packets within a single slotframe.

Keywords
Link scheduling, 802.15.4e-TSCH, slotframe, timeslot, cell, OpenWSN.

1. Introduction

Wireless sensor networks (WSNs) play a key role in enabling the Internet of Things. They are increasingly used for a wide variety of applications ranging from environmental monitoring to industrial automation considering specific requirements namely data transmission latency, reliability, and energy efficiency.

A widely-used technology to fulfil these demands is IEEE 802.15.4 standard [1], which specifies physical and MAC functionality for low power, low data rate wireless networks. Although this standard is well adapted to the one-hop communications, it is not suitable for multi-hop scenario mainly because of high energy consumption and increasing interference. According to the standard, all nodes use a single common channel, which increases the unreliability due to interference and fading. Moreover, intermediate nodes suffer from high energy consumption as they stay active more often to relay data packets. The performance of 802.15.4 networks for different applications has already been studied under various conditions in the literature [2, 3].

To better adapt the 802.15.4 standard to multi-channel multi-hop scenario of the industrial IoT, the Internet Engineering Task Force (IETF) proposed IEEE 802.15.4e [4] amendment that specifies several MAC mechanisms to provide reliability, predictable delay, energy conservation, and higher throughput for multi-hop WSNs. It replaces the MAC protocol without changing the underlying physical layer. The TSCH mechanism of IEEE 802.15.4e aims at reducing the impact of the wireless channel unpredictability for low power and lossy networks. Its slotframe structure allows transmitting data packets more reliably with low latency. It also enables saving energy as each node shares a schedule, allowing it to know in advance when to turn its radio on or off.

The standard only provides a framework but it does not mandate any specific scheduling mechanism [5]. In this paper, we propose two centralized mechanisms to compute the common schedule for 802.15.4e-TSCH networks with bounded delay and high reliability.

Our contribution in this paper is threefold:

- We propose Height-based Scheduling, a centralized mechanism that relies on topological characteristics of the network. It allocates the cells required for each node according to its height in the DoDAG (Destination-Oriented Directed Acyclic Graph) topology.
- More importantly, we propose T^2AS, a more efficient mechanism that uses a composite weighting function to help build sub-sequential scheduling for multi-hop scenarios. According to T^2AS, more cells are reserved for a node with a

larger subtree, i.e. subtree of which it is the root. In addition, it tries to schedule links with more load and longer distance from the root in early timeslots. An interesting characteristic of T^2AS is that all data packets could reach the sink within one slotframe.

- We also conduct extensive simulations to evaluate the performance of both mechanisms under quasi-realistic conditions where an instance of the mote firmware is created for each emulated node.

The rest of this paper is organized as follows: Section 2 overviews the IEEE 802.15.4-TSCH standard and OpenWSN protocol stack. Related work is briefly discussed in section 3. Our scheduling mechanisms for TSCH networks are presented in section 4. Section 5 discusses the simulation scenarios and reports the results obtained from the performance evaluation. Finally, the paper is concluded in section 6.

2. Background knowledge

2.1. TSCH mechanism

TSCH proposes a FTDMA-like access method to use diversity in time and frequency to provide reliability to the network. More precisely, time-slotted access permits to achieve bounded latency, multi-channel communications increase network capacity, and finally slow channel hopping mitigates the effects of interference and multipath fading.

In TSCH, time is sliced up into timeslots. Each timeslot is large enough to allow a node to transmit a maximum-length data packet and eventually to receive its corresponding ACK. Absolute Sequence Number (ASN) counts the number of timeslots since the DAGroot has started. Timeslots are grouped into slotframes. A slotframe continuously repeats over time. There are 16 non-overlapping channels available for hopping in TSCH, each with 5MHz of bandwidth. The frequency to use is calculated based on the ASN and the channel offset at the beginning of each timeslot. At the beginning of each slotframe, all nodes are synchronized. The PAN coordinator advertises enhanced beacon (EB) to schedule slotframes. A cell (link) is defined as the pairwise of assignment of a directed communication between devices in a specific timeslot on a given channel offset. Each link can be *shared* or *dedicated*. In a shared link, TSCH uses a CSMA/CA mechanism to avoid collision. However, it uses a contention-free access for a dedicated link [4]. Fig. 1 illustrates a sample schedule for a slotframe with three timeslots and four non-overlapping channels in an 8-node network topology. As shown in Fig. 2, within a time slot, a node transmits a data packet and receives its corresponding acknowledgement. If the acknowledgement packet is not received in the same timeslot, then it will be retransmitted in another timeslot according to the TSCH schedule.

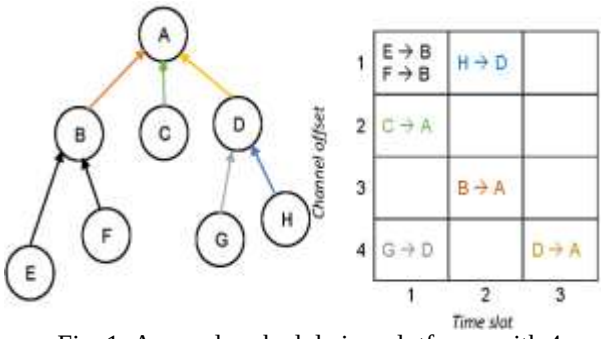


Fig. 1. A sample schedule in a slotframe with 4 channels and 3 timeslots

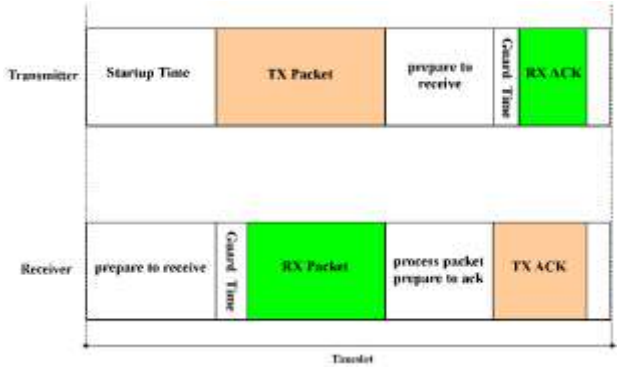


Fig. 2. Structure of a timeslot in TSCH

2.2. OpenWSN Protocol Stack

OpenWSN [6] and D-MSR [7] are two examples of protocol stacks that implement the 802.15.4e standard. D-MSR is implemented in the NS-2 environment. However, OpenWSN provides a full protocol stack for low-power Internet-connected wireless mesh networks. It implements, on top of IEEE802.15.4e, IoT standards such as 6LoWPAN, RPL, and CoAP enabling an OpenWSN network to connect to the IPv6 Internet. The resulting hardware-independent protocol stack (cf. Fig. 3) is becoming a foundation for the IoT industry.

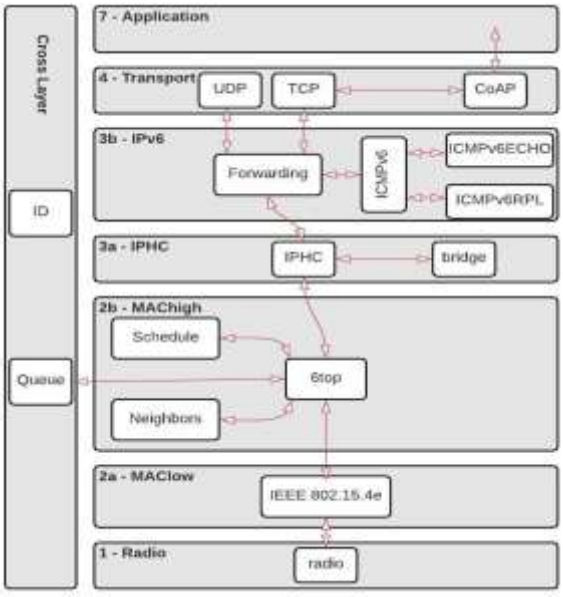


Fig. 3. OpenWSN protocol stack

Applications for specific purposes are located at the User Applications layer and operate on top of several

transport layer protocols. Constrained Application Protocol (CoAP) [8] at the transport layer enables RESTful interaction with individual motes without the overhead of TCP and HTTP. Unlike HTTP, CoAP is specifically designed for LLN networks. It consists of a 4-byte header on top of UDP. A CoAP-enabled mote can act as both a web client and a web server.

IPv6 layer has three main tasks to accomplish: packet forwarding, routing using RPL protocol [9] and managing control packets through ICMPv6. RPL is able to quickly setup network routes, to distribute routing information among nodes, and to adapt the topology in an efficient way. The nodes are connected via a multi-hop network to a root device, which is responsible for data collection and coordination tasks.

At the IPHC sub-layer, IPv6 over Low power WPAN (6LoWPAN) is used for compressing the IPv6 header to minimize packet size. The maximum frame size in the 802.15.4 standard is 127 bytes while the IPV6 header size is 40 bytes. Therefore, without header compression, the 40-byte header of IPv6 packets would occupy almost a third of each frame.

MAC layer is divided into two sublayers: The MAChigh sub-layer is used to specify the slotframe structure and to accomplish link scheduling. A slotframe consists of a number of timeslots and repeats over time. The other sublayer is mainly responsible for the channel hopping mechanism. Physical (Radio) layer is specified for supported platforms in OpenWSN. This layer differs across a variety of platforms. Finally, the cross-layer part is used to interconnect different layers of the protocol stack.

3. Related Work

The reliable data transfer requires precise scheduling to keep the number of transmissions as low as possible and to guarantee a certain level of reliability. Since a TSCH network relies on a strict organization of the transmissions, several scheduling algorithms have been proposed in the literature.

The centralized approaches rely on a Path Computation Element, which needs often to know the traffic generated by each node, and the topology used to route the packets. TASA [5] represents a pioneering piece of work, defining the different constraints to schedule accurately the transmissions. It tries to multiplex the transmissions to create a compact schedule. AMUS [10] aims to provide low delay guarantees for time-sensitive applications, by scheduling sequentially the transmissions along the route. Besides, redundant cells are assigned to vulnerable links with a heavy load to deal with retransmissions. MODESA [11] deals with a sink equipped with several radio interfaces and tries to multiplex the transmissions through the different interfaces of the sink to maximize the throughput. SchedEx [12] fortifies the schedule, by allocating additional slots to lower-bound the end-to-end reliability. The number of extra timeslots is calculated in function of the packet reception rate of every link. Huynh et al. [13] proposed a centralized scheduling method that concentrates on reliability and energy efficiency by deciding on the best next hop to forward the packet. LOST adopts localized scheduling: it only needs single hop information to determine the schedule with the constraint of prohibiting bad radio channels usage [14]. [15] proposes a centralized and compact scheduling algorithm tailored to a multipath routing mechanism, which enables multiple transmitters to use a shared cell for delivering data packets to a common receiver. The flows are prioritized regarding their distance to the sink. The authors in [16] aim to provide an acceptable trade-off between end-to-end delay and network lifetime by over-provisioning extra cells for retransmission packets. OST[17], schedules the slotframe on-demand in two steps: first, it periodically meets the bandwidth requirements of the current traffic load. Second, it observes burst traffic and allocates additional slots accordingly to route the traffic.

Distributed approaches were also proposed, where each node decides reactively how many cells to reserve. DiSCA *et al*. [18] is a distributed algorithm that tries to minimize the number of timeslots required to deliver all the packets to the sink. The algorithm proceeds iteratively (in rounds), allocating at the round i the i^{th} transmission of each node. Domingo-Prieto *et al.* [19] considers the TSCH scheduling as a closed-loop control problem and proposed a PID-based distributed scheduling for 6TiSCH networks. At each slotframe, the PID controller determines the number of cells that should be added or removed per neighbour, according to the state of the queues and the previous usage of slotframes cells. Sabzevari *et al.* investigate funnelling effect [20] in WSNs with converge-cast traffic. They allocate dedicated cells to the links closer to the sink as they relay much more traffic compared to the leaves. Each node estimates the number of cells according to the information it obtains from its neighbours. One of the default scheduling functions designated for 6TiSCH is SFx [21] that is relatively inspired by OTF [22]. SFx estimates the number of appropriate cells between neighbour nodes, according to the currently scheduled cells and cell requirements of the neighbour nodes. It tracks the number of incoming and outgoing packets and uses these statistics for new cell reservations.

In addition to data packets scheduling, enhanced beacons scheduling is also of importance for 802.15.4e networks. In [23], Zou *et al.* investigated the effects of a number of co-located PAN coordinators on the performance of TSCH networks. They proposed an algorithm to schedule enhanced beacons (EB) transmissions where neighbouring PAN coordinators exchange their schedules to detect conflicts and reallocate the timeslots if necessary. De Guglielmo *et al.* [24] formulate EB scheduling in TSCH networks as an optimization problem with the goal to minimize the average joining time of a node. In [25], the authors divide the nodes into different groups using an optimization method. By a Mixed-Integer Linear Programming (MILP) model, one of the sets is selected intermittently to cover the whole network while the other sets are in energy saving mode. This procedure is repeated over time with different sets.

Our main work is a new centralized scheduling mechanism, which relies on both traffic demands and network topology and aims to provide more reliability and low delay for 802.15.4e-TSCH networks.

4. Topology/Traffic Aware Scheduling

In this section, we propose two link scheduling mechanisms for TSCH networks with converge-cast traffic: The first one, HS, is a topology-based scheduling that computes the scheduling matrix based on the DoDAG structure. A DoDAG is a directed acyclic graph rooted at a single destination, which is the sink in our case.

However, the second method we call T^2AS takes into account both the traffic demand and the network topology. The number of cells allocated to a specific transmission link (a pair of nodes) depends on the location of its nodes (for the two algorithms) in the DoDAG structure as well as the amount of traffic to be carried over that link (for T^2AS).

A summary of notations we use here is shown in Table I.

Table I. A summary of notations

Notion	Meaning
G(V, L)	DoDAG graph with node set *V* and link set *L*
CS(L)	conflict set of link set *L*
$n_c(u \rightarrow v)$	number of cells reserved for transmission of a packet from *u* to *v*
height(u)	height of node *u* in the corresponding tree topology
$w_t(v)$	weighting function for node *v* at time *t*
ST(v)	subtree rooted at node *v*
$load_t(u)$	number of data packets available in node *u* at time *t*
distance(u)	hop-count from node *u* to reach the sink
$\bar{E}_{slotframe}$	average energy consumed by all nodes in a slotframe
$N_{tx}(N_{rx})$	number of TXs(RXs) during one slotframe
$C_{tx}(C_{rx})$	TX(RX) current consumption
V	low power voltage
L	packet length
R	bit rate

Let *G(V, L)* be a DoDAG with node set *V* and link set *L*. Now, assume a set of links $L' \subseteq L$. Let S' represents the set of nodes that correspond to the endpoints of L'. We define the *conflict set* of L' in Eq. (1) as the set of links in $L \backslash L'$ for which one endpoint is inS'. Mathematically, this would be denoted as:

$$\boldsymbol{CS}(\boldsymbol{L'}) = \{(\boldsymbol{a}, \boldsymbol{b}) | (\boldsymbol{a} \in \boldsymbol{S} \oplus \boldsymbol{b} \in \boldsymbol{S})\} \quad (1)$$

Where $\oplus$ is the `XOR` operation. Two links are said to be in conflict if their corresponding nodes cannot transmit/receive at the same time over the same channel. For both mechanisms, we assume that each node has only one radio interface. We also assume using dedicated slots for transmission.

4.1. Height-based Scheduling (HS)

In our first centralized mechanism, the number of cells reserved for each link $u \rightarrow v$ depends on the location of its nodes in the network topology and does not depend on the traffic volume. More precisely, the number of cells allocated to $u \rightarrow v$ is computed as:

$$n_c(u \rightarrow v) = \begin{cases} height(u) + 1, & \text{if } u \text{ is an inner node} \\ 1, & \text{if } u \text{ is a leaf node} \end{cases} \quad (2)$$

where $height(u)$ represents the height of u in the corresponding tree topology and $n_c(u \rightarrow v)$ refers to the number of cells reserved for the node *u(v)* as transmitter(receiver). In a tree, the height of a node is the number of edges on the path from the node to the deepest leaf. The height of a leaf node is 0.

Algorithm **1** illustrates the principal steps of this mechanism. For HS algorithm, we define *links* as a vector of the following `struct` for each link in the network:

`<from, to, height, src_type, nCells>`.

The attribute `height` represents the height of the node `from`. `src_type` determines if `from` is a leaf or an inner node. `nCells` denotes the number of cells to be scheduled for the current link and is computed according to Eq.(2).

At the beginning, the *links* vector is initialized by using the DoDAG structure of the network (line 12). It is then sorted in descending order by `distance` (line 14). According to the sorting procedure, an outer link is placed before an inner one when both are at the same `distance` from the sink. *csLinks* maintains the links to be scheduled in the current timeslot. A new cell is scheduled for the current link *links[i]* if:

1. its corresponding `nCells` is greater than zero (lines 16-17) and
2. it is not in conflict with any other link already scheduled in the current timeslot.

Then, a maximum number of non-conflicting links are scheduled within the same timeslot but on different channels (lines 19-24). The next cells for the current link *i* are scheduled in the next subsequent timeslots (cf. **while** loop). This process is repeated using the outer **for** loop until all links are scheduled.

The HS mechanism permits sub-sequential multi-hop scheduling. For instance, in a multi-hop sequence like $A \rightarrow B \rightarrow C$, the link $A \rightarrow B$ is scheduled in an earlier timeslot compared to $B \rightarrow C$. Consequently, a multi-hop sequence is scheduled in one slotframe, which is resulted in a lower end-to-end delay. However, HS does not consider the traffic demand. Therefore, an increase in the traffic may result in a noticeable degradation in delay and packet loss ratio.

The time complexity of the HS algorithm is derived as follows: The complexity of the outer `for` loop (lines 15-29) is *O(n)*, where *n* denotes the number of nodes since it schedules all the links. The `while` loop runs *C* times where *C* is the average number of cells allocated for each link (usually a small value). The complexity of the inner `for` loop is of *O(n)*. Thus, the overall time complexity is $O(Cn^2)$.

Algorithm 1. HS Scheduling

```
1:  procedure SCHEDULELINK(I, ts, ch)          // Schedule links
2:      (a, b) ← (links[i].from, links[i].to)
3:      schedule((a,b)) in slotframeMatrix(ts, ch)
4:      links[i].nCells ← links[i].nCells - 1;
5:      csLinks ← csLinks ∪ {(a, b)}
6:  end procedure
7:  Input: G(V, E)                             // DoDAG topology
8:  Output: slotframeMatrix                    // scheduling matrix
9:  Global Variables: links, csLinks
10: Initialization:
11:   ts ← 1; ch ← 1        // Initializing timeslot and channel offset
12:   set links attributes using DoDAG topology.
13:   slotframeMatrix ← Ø; csLinks ← Ø
14: sort(links)                        // sort links in descending order
15: for i = 1 to #links do
16:   while links[i].nCells > 0 do
17:       SCHEDULELINK(i, ts, ch)
18:       ch ← ch + 1
19:       for j = 1 to #links; j ≠ i do
20:           if links[j].nCells > 0 && (links[j].from, links[j].to) ∉
    CS(csLinks) then
21:               SCHEDULELINK(i, ts, ch)
22:               ch ← ch + 1
23:           end if
24:       end for
25:       csLinks ← Ø
26:       ch ← 1
27:       ts ← SFLength % (ts + 1)
28:   end while
29: end for
30: return slotFrameMatrix
```

4.2. Topology and Traffic Aware Scheduling(T²AS)

Our second scheduling mechanism for TSCH networks uses both topology and traffic demand to propose a more flexible schedule matrix. By taking into account both traffic and topology as shown in Eq.(3), T²AS provides high reliability as well as bounded latency. We define a weighting function $w: V \rightarrow N^0$, where V denotes the set of nodes in the network and N^0 is the set of non-negative integers. The weight value for a given node *v* may vary over subsequent timeslots. It is defined as follows:

$$w_t(v) = \sum_{u \in ST(v)} load_t(u) * distance(u) \qquad (3)$$

Where *ST(v)* is the subtree rooted at node *v*, $load_t(u)$ is the number of data packets available in *u* at time *t* targeted to the root, and *distance(u)* is the number of hops from *u* to reach the root. We need to schedule links with more load and longer distance to the root in early timeslots. In other words, we prioritize links with greater weights according to Eq.(3). By applying this weighting function, we consider leaf nodes not to be scheduled at the end of the slotframe. This function provides a balance in scheduling for congested nodes and for nodes with a longer distance from the sink as well.

Algorithm 2 illustrates the principal steps of our proposed mechanism. Here, each entry of vector *links* comprises the following attributes:

`<from, to, distance(dist), load, weight(wg)>`

The `load` and `weight` of a link point to the load and the weight of its `from` node. In the pseudo-code, load(*a*) and weight(*a*) are used to represent *links[i].load* and *links[i].weight* respectively, where *i* denotes the entry that contains link *(a,b)*.

The main procedure of our mechanism is as follows: Starting from the first timeslot (*ts = 1*), the objective is to schedule the maximum possible number of links in each timeslot while scheduling nodes with higher weight in the earlier timeslots. As in HS, *csLinks* maintains the links to be scheduled in the current timeslot. At the beginning of each timeslot, the *csLinks* initialized to empty. As shown in line 13, all links in *links* are verified in decreasing order of *weight*. The `nextMaxWeight()` function returns the link with the next maximum *weight*. Then, the candidate link $a \rightarrow b$ is scheduled in the current timeslot if two following conditions hold:

1. *load(a) > 0* ;
2. $a \rightarrow b$ is not in conflict with any other links already scheduled in the current timeslot

After verifying all links, the vector *links* is updated (lines 17-24). For each link in *csLinks*, the load and weight parameters of the corresponding link in *links* must be updated. For each scheduled link *(a,b)*, *load(a)* and *load(b)* are decremented and incremented by 1 respectively (lines 27-30). Based on the updated load values, weights are recomputed using the procedure *updateWeights* according to Eq.(3) (lines 1-5). The same procedure is repeated for the next timeslot. We assume that T_{max} is the minimum value to schedule vector *links* in an entire slotframe.

Fig. 4 illustrates a sample run of our mechanism for a 4-node tree topology. The traffic pattern is converge-cast and it is assumed that every node has one data packet to send during each slotframe. The evolution of *load* and *weight* parameters for the *links* vector are depicted in Fig. 4 and the scheduled link is also highlighted for each timeslot. For this scenario, the minimum possible value for T_{max} is 3. Two cells are assigned to $c \rightarrow a$, one for transmitting its own packet and the other one for forwarding the packet sent by *d*.

The time complexity of this algorithm is $O(Sn^2)$ where *S* is the length of the slotframe.

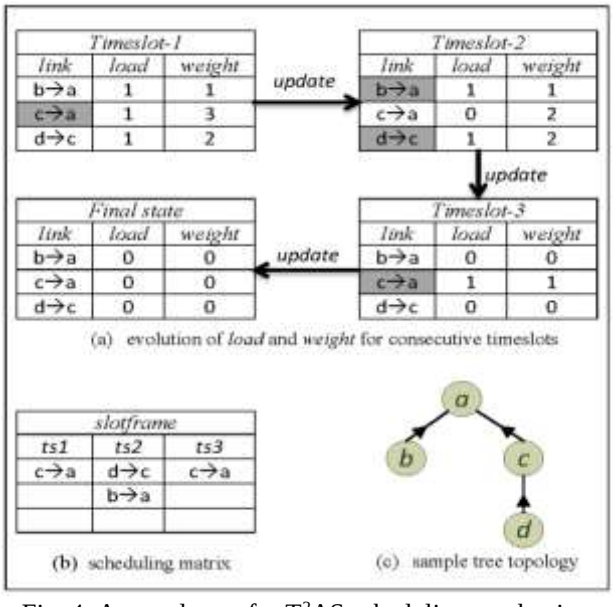


Fig. 4. A sample run for T²AS scheduling mechanism

Algorithm 2. T²AS Scheduling

```
1:  procedure UPDATEWEIGHTS()
2:      for a ∈ Vnodes do
3:          weight(a) ← ∑_{n∈ST(a)} load(n) * dist(n)
4:      end for
5:  end procedure
6:  Input: load              // traffic vector during a slotframe period
7:  nodes                    // vector of nodes
8:  links                    // vector of edges in the DoDAG topology.
9:  Output: slotframeMatrix                    // scheduling matrix
10: Initialization:
11: ts ← 1, ch ← 1                    // timeslot and channel offset
12: slotframeMatrix ← Ø
13: sort(links)                       // sort links in descending order
14: for ts = 1 to TS_max do
15:    UPDATEWEIGHTS()
16:    csLinks ← Ø
17:    while !allLinksVerified do
18:        (a, b) ← nextMaxWeight()
19:        if load(a) > 0 && ((a, b) ∉ CS(csLinks)) then
20:            schedule (a,b) in slotframeMatrix(ts, ch)
21:            csLink ← clsLink ∪ {(a, b)}
22:            ch ← ch + 1
23:        end if
24:    end while
25:
26:    /* Update load and weight for this link. */
27:    for (a, b) ∈ CS(csLinks) do
28:        Load(a) ← load(a) - 1
29:        Load(b) ← load(b) + 1
30:    end for
31:    ch ← 1
32: end for
33: return slotFrameMatrix
```

5. Implementation and Performance Evaluation

Our proposed mechanisms were implemented in the OpenWSN simulator. The experimental results obtained from this platform have been already confirmed to be close to the reality according to [26].

In order to get more accurate results, the OpenWSN simulator creates an instance of the mote firmware for each emulated node. This means every detail of a mote is simulated, which makes it running slowly. Therefore, a large number of motes would run with a very low speed and cause a long time to synchronize. Consequently, we decided to run the experiments for various network sizes ranging from 4 to 16 nodes with a random tree topology. For each network size, we also generated five different random topologies to have representative results. For example, Fig. 5 represents a sample random topology with 16 nodes, which has been used in our simulation. We assume no node failure in the network, thus we can totally focus on latency and reliability. We, first provide a microscopic evaluation of our two proposed mechanisms by reporting performance results in detail and per-flow for both. Then, we compare their performance with those of Funnelling-aware TSCH (FA-TSCH), a prior work already implemented in OpenWSN by our research group [20].

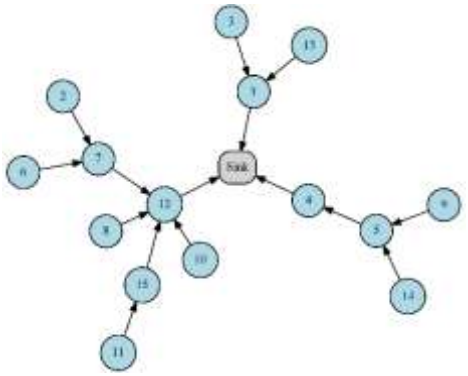


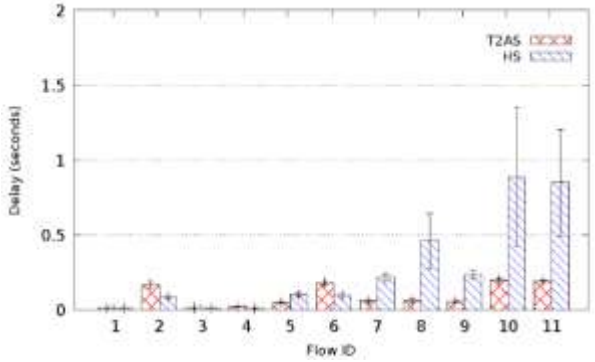
Fig. 5. 16-node topology

The size of the slotframe is set at 23 timeslots to adapt it to the size of the network. The Chipcon CC2420 radio transceiver is used. Each point represents the average of 10 simulation runs and the results are displayed with a *95%* confidence intervals. Moreover, each node generates one data packet at the beginning of each slotframe. To keep the nodes fully synchronized and to avoid collisions in control messages, three shared cells are specified at the beginning of each slotframe as the minimal schedule. As a central entity, DAGroot is fully aware of the topology, the routing tree and the traffic demands, thus it builds the schedule and disseminates it through the minimal schedule. Simulation parameters are summarized in Table II.

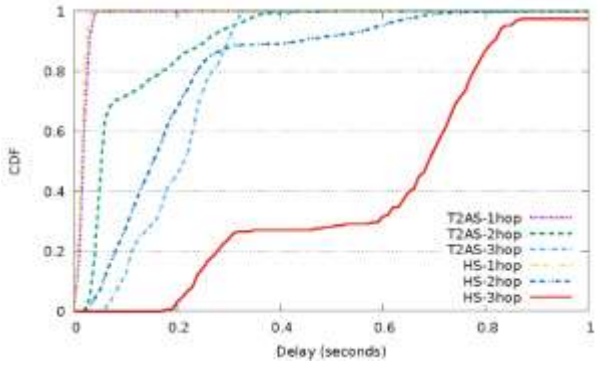
Table II. Simulation Parameters

Parameter	Value
Physical	802.15.4 2400MHz OQPSK
Radio	CC2420 current consumption [C_{tx} = 17.4 *mA* and C_{rx} = 18.8 *mA*] supply voltage [1.8 *volts*]
Access mode	802.15.4e-TSCH
Timeslot duration	15 ms
Slotframe size	23 slots
# of channels	16
# of shared cells	3
Routing protocol	RPL
CBR	1 pkt / slotframe
Packet size	100 bytes
Network configuration	random topology
Network size	4 to 16 sensor nodes

The first simulation scenario uses a 12-node random topology (Fig. 6). There are 11 active converge-cast flows destined to the sink. Among these flows, two are of 3-hop, six are of 2-hop, and three are of one-hop length. Fig. 6 (a) shows the average end-to-end delay for each traffic flow in this random topology. As shown in the figure, T^2AS delivers generated data packets within one slotframe even for longer flows. However, the longer flows incur longer delays under HS and it sometimes takes more than three slotframes for their packets to reach the sink. The cumulative density function (CDF) of end-to-end delay for flows with different length (cf. Fig. 6 (b)) again confirms the good performance of T^2AS compared to HS.



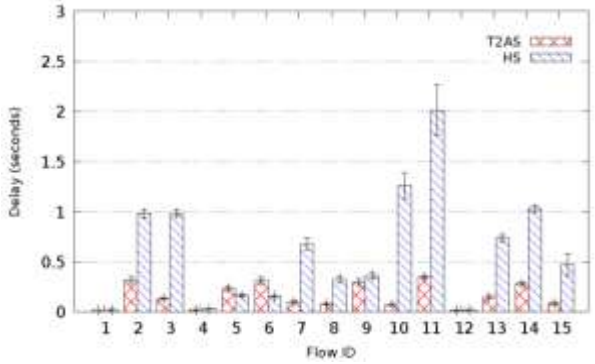
(a)



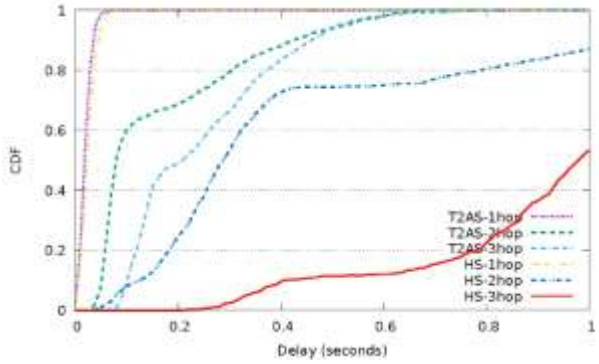
(b)

Fig. 6. Simulation results for a 12-node random topology. (a) end-to-end delay, (b) CDF of e2e delay.

The results for the same scenario are also reported for a 16-node random network in Fig. 7. Here, five flows are of 3-hop, seven are of 2-hop, and three are of 1-hop length. As shown in Fig. 7 (a), e.g. for flow 11, T^{2}AS outperforms HS by a factor of almost 6 in terms of delay. Moreover, HS results in high jitter specifically for longer flows. Fig. 7 (b) shows the CDF of end-to-end delay for flows with different length in the 16-node network. We can observe from the figure that, while under T^{2}AS, it takes at most one slotframe for more than *85%* of packets belonging to 3-hop flows to reach the sink, only *10%* for the same flows under HS algorithm are delivered to the sink within one slotframe. The main reason is that HS does not reserve as many cells as needed based on the traffic generated by the network.



(a)



(b)

Fig. 7. Simulation results for a 16-node random topology. (a) end-to-end delay, (b) CDF of e2e delay.

The instantaneous queue size of the relaying node 12(Fig. 5), with a subtree of size seven, is reported for both algorithms in Fig. 8. At time 50 seconds, all nodes in the subtree are already joined to the network via node 12. The maximum queue size was set to 10 for all simulation scenarios. After a while, the queue size under HS blows up because enough cells have not been reserved for node 12 to forward the data packets.

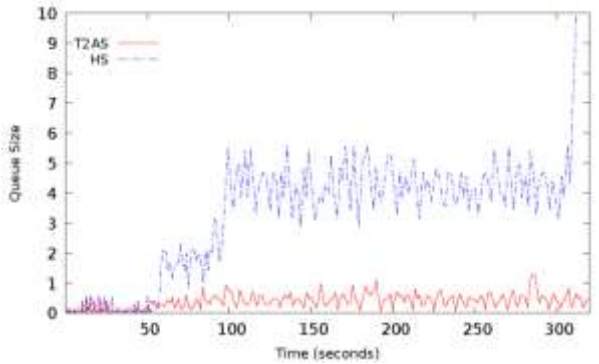
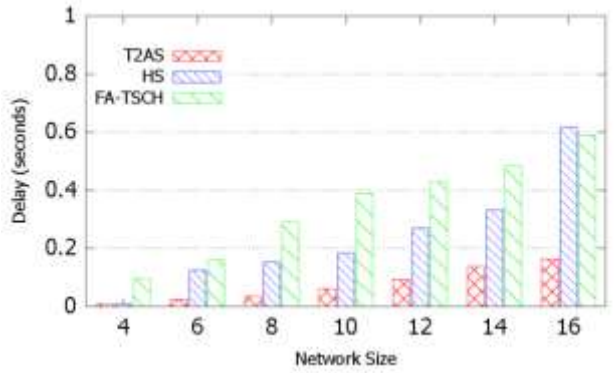
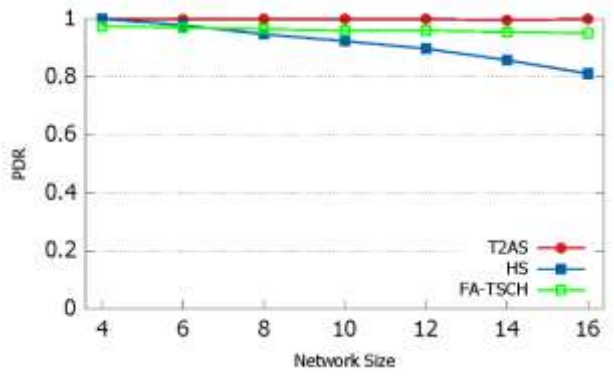


Fig. 8. Instantaneous queue size for an internal node with a 7-node subtree (a 16-node random topology)

To compare the scalability of our mechanisms, we run simulations for various network sizes. The results shown in Fig. 9 confirm the acceptable scalability of T^{2}AS over FA-TSCH and HS in terms of both packet delivery ratio (PDR) and end-to-end delay. For 16-node random networks in Fig. 9 (a), T^{2}AS outperforms HS by a factor of 3.5 and FA-TSCH by a factor of 3 in terms of average end-to-end delay. As the number of nodes increases, FA-TSCH performs better with regard to packet delivery and suffers less from the long delay compared to HS. The difference in latency between the T^{2}AS and FA-TSCH is due to the behaviour of the FA-TSCH algorithm; Unlike T^{2}AS, the links of a given flow are not necessarily scheduled one after another when using FA-TSCH. This likely causes a bigger delay. T^{2}AS mechanism achieves very good reliability when the traffic pattern is known a priori. Moreover, as shown in Fig. 9 (b), FA-TSCH offers higher reliability compared to HS and its PDR degrades more smoothly by increasing the network size.



(a)



(b)

Fig. 9. Simulation results for network with varying size. (a) Average e2e delay w.r.t. network size, (b) Average packet delivery ratio w.r.t. network size.

Finally, the amount of energy consumed by each node is particularly important in wireless sensor networks. We define $\bar{E}_{slotframe}$ in Eq.(4) as the average amount of energy consumed by all sensor nodes in the network during a single slotframe.

$$\bar{E}_{slotframe} = N_{tx}\left(V * C_{tx} * \frac{L}{R}\right) + N_{rx}(V * C_{rx} * \frac{L}{R}) \qquad (4)$$

Where V is the low power voltage, $C_{tx}(C_{rx})$ represents TX(RX) current consumption, $N_{tx}(N_{rx})$ is the number of TXs(RXs) during one slotframe, L is the packet length, and R denotes the channel bit rate. Since the sink node is assumed to have an unlimited resource of energy, its energy consumption is not considered when computing $\bar{E}_{slotframe}$. As shown in Fig. 10, for a heavy traffic volume, T^{2}AS consumes more energy compared to HS since it permits using more cells to forward traffic during the same period. Note that, according to Fig. 9(a), Fig. 9(b) and Fig. 10, T^{2}AS outperforms HS in terms of PDR and delay at the cost of a slight increase in energy consumption. However, because T^{2}AS adapts its scheduling matrix to the size of the traffic demand, it consumes less energy compared to HS when the incoming traffic is light.

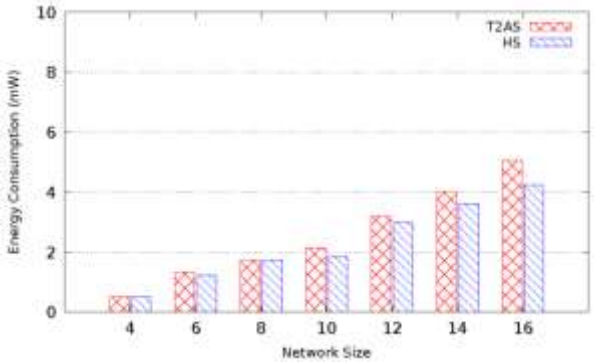


Fig. 10. Average energy consumption during one slotframe for network with varying size

6. Conclusion

In this paper, we proposed two centralized scheduling algorithms for multi-hop IEEE 802.15.4e-TSCH networks. The T^{2}AS algorithm that considers both topology and traffic, schedules the cells in such a way that each data packet could be delivered to the sink within the same slotframe it was generated. On the other hand, HS schedules the cells only based on their distance to the sink. Both algorithms guarantee multi-hop sub-sequential scheduling which results in lower end-to-end delay. The simulation results, obtained from OpenWSN confirm the efficiency of the T^{2}AS mechanism specifically in terms of network delay and PDR which are critical for time-sensitive automation applications, while HS occupies fewer cells and consumes much less energy.

As a future work, we plan to investigate the efficacy of our scheduling mechanisms when deployed along with a node-disjoint multipath routing. It is also possible to study the effect of reusing shared cells to piggyback unacknowledged data during the next slotframe.

7. References

[1] IEEE, "IEEE standard for local and metropolitan area networks–part 15.4:Low-rate wireless personal area networks (lr-wpans)", *IEEE Std802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, Sept 2011, pp. 1–314.

[2] F.C. Jiang, H.W. Wu, C. T. Yang, "Traffic load analysis and its application to enhancing longevity on ieee802.15. 4/zigbee sensor network", *The Journal of Supercomputing*, vol. 62, no. 2, pp. 895–915, 2012.

[3] M. Nassiri, M. Boujari, S. V. Azhari. "Energy-aware and load-balanced parent selection in rplrouting for wireless sensor networks", *International Journal of Wireless and Mobile Computing*, vol. 9, no. 3, pp. 231–239, 2015.

[4] IEEE, "IEEE Std 802.15.4e, Part. 15.4: Low-Rate Wireless Personal AreaNetworks (LR-WPANs) Amendament 1: MAC sublayer", *IEEE Computer Society*, 2012.

[5] M. R. Palattella, N. Accettura, M. Dohler, L.A. Grieco, G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15. 4e networks", In 23rd IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), September 2012, Sydney, Australia, pp. 327–332.

[6] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, K. Pister, "Openwsn: a standards-based low-power wireless development en-

vironment". *Transactions on Emerging Telecommunications Tech-nologies*, vol. 23, no. 5, pp. 480–493, 2012.
[7] P. Zand, A. Dilo, P. Havinga, "D-MSR: A distributednetwork management scheme for real-time monitoring and pro-cess control applications in wireless industrial automation", *Sensors*, vol. 13, no. 7, pp. 8239–8284, 2013.
[8] Z. Shelby, K. Hartke, C. Bormann, "The constrained application protocol (CoAP)", RFC 7252, June 2014.
[9] T. Winter, "Rpl: Ipv6 routing protocol for low-power and lossynetworks", RFC 6550, March 2012.
[10] Y. Jin, P. Kulkarni, J. Wilcox, M. Sooriya-bandara, "A centralized scheduling algorithm for ieee 802.15.4e tsch based industrial low power wireless networks". In IEEE Wireless Communications and Networking Conference (WCNC), April 2016, Doha, Qatar, pp. 1–6.
[11] R. Soua, P. Minet, E. Livolant, "MODESA: Anoptimized multichannel slot assignment for raw data convergecastin wireless sensor networks", In Proceedings of IEEE 31st International Performance Computing and Communications Conference(IPCCC), December 2012, Austin, TX, USA, pp. 91–100.
[12] F. Dobslaw, T. Zhang, M. Gidlund, "End-to-EndReliability-aware Scheduling for Wireless Sensor Networks", *IEEE Transactions on Industrial Informatics*, vol. 12, no. 2, pp. 758–767, 2016.
[13] T. Huynh, F. Theoleyre, W.J. Hwang, "On the interest of opportunistic anycast scheduling for wireless lowpower lossy networks", *Computer Communications*, vol. 104, pp. 55–66, 2017.
[14] D. Zorbas, V. Kotsiou, F. Théoleyre, G. ZPapadopoulos, C. Douligeris, "Lost: Localized blacklist-ing aware scheduling algorithm for ieee 802.15. 4-tsch networks", *In Wireless Days (WD)*, April 2018, Dubai, United Arab Emirates, pp. 110–115.
[15] E. M. Ahrar, M. Nassiri, F. Theoleyre, "Multipath aware scheduling for high reliability and fault tolerance in low power industrial networks", *Journal of Network and Computer Applications*, vol. 142, pp. 25–36, 2019.
[16] K. Brun-Laguna, P. Minet, Y. Tanaka, "Optimized scheduling for time-critical industrial IoT", In IEEE Global Communications Conference, December 2019, Hawaii, USA, pp. 1-6.
[17] S. Jeong, H.S. Kim, J. Paek, S.W. Bahk, "OST: On-demand tsch scheduling with traffic-awareness", In IEEE Conference on Computer Communications (INFOCOM), July 2020, Virtual Conference.
[18] R. Soua, P. Minet, E. Livolant, "DiSCA: A distributed scheduling for convergecast in multichannel wireless sensor networks", In Proceedings of International Symposium on Integrated Network Management, May 2015, Ottawa, ON, Canada, pp. 156–164.
[19] M. Domingo-Prieto, T. Chang, X. Vilajosana, T. Watteyne, "Distributed pid-based scheduling for 6tischnetworks". *IEEE Communications Letters*, vol. 20, no. 5, pp. 1006–1009, 2016.
[20] M. Sabzevari, M. Nassiri, "A distributed mechanism for cell scheduling to reduce funneling effect in 802.15.4e-based wireless networks", *Tabriz Journal of Electrical Engineering*, vol. 46, no. 3, pp. 221–232, 2016 (in persian).
[21] D. Dujovne, L. Grieco, M. Palattella, N. Accettura, "6tisch ex-perimental scheduling function (sfx)", *Draft, IETF*, March 2018.
[22] M. R. Palattella, T. Watteyne, Q. Wang, K. Mu-raoka, N. Accettura, D. Dujovne, L. A. Grieco, T. Engel, "On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks", *IEEE Sensors Journal*, vol. 16, pp. 550–560, 2016.
[23] M. Zou, J.L. Lu, F. Yang, M. Malaspina, F. Theoleyre, M.Y. Wu, "Distributed scheduling of enhanced beacons for ieee802.15.4-tsch body area networks". In International Conference on Ad-Hoc Networks and Wireless (ADHOC NOW), July 2016, Lile, France, pp. 3–16.
[24] D. De Guglielmo, S. Brienza, G. Anas-Tasi. "A model-based beacon scheduling algorithm for ieee 802.15.4e tsch networks", In 17th IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2016, Coimbra, Portuga, pp. 1–9.
[25] H. Bakhshi, S. H. Keshmirifar, "Lifetime improvement and cover-age maximization of cluster-based wireless sensor network usingmulti hop routing", *Tabriz Journal of Electrical Engineering*, vol. 47, no. 4, pp. 1637–1647, 2018(in persian).
[26] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne,T. Chang, K. Pister, "A realistic energy consumption model for tsch networks", *IEEE Sensors Journal*, vol. 14, no. 2, pp. 482–489, 2014.