GTTI 2013 Session on Telecommunication Networks
This paper is a preliminary version of a paper published in the Second IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services, IoT-SoS 2013

DeTAS: a Decentralized Traffic Aware Scheduling technique enabling IoT-compliant Multi-hop Low-power and Lossy Networks

Nicola Accettura*, Maria Rita Palattella†, Gennaro Boggia*, Luigi Alfredo Grieco* and Mischa Dohler‡
*DEI, Politecnico di Bari (Italy), {n.accettura, a.grieco, g.boggia}@poliba.it
†SnT, University of Luxembourg (Luxembourg), maria-rita.palattella@uni.lu
‡CTTC (Spain), mischa.dohler@cttc.es

***Abstract*—The emerging IEEE802.15.4e standard and IETF RPL routing protocol provide key functionalities useful for a really viable Internet of Things. Indeed, they are core to the organization of multi-hop Low-power and Lossy Networks. Nevertheless, the design of effective scheduling schemes in such systems is one of the major problems, which requires research efforts in order to be solved. Actually, there are no specifications about how schedules should be realized. Trying to fill this gap, the aim of this paper is twofold. First of all, we present a new Decentralized Traffic-Aware Scheduling algorithm, which is able to construct optimum multi-hop schedules in a distributed fashion. Secondly, we prove its effectiveness by means of simulation results, and we compare such descentralized solution with a centralized one. The reported performance results encourage the use of such decentralized scheme, since it allows a very efficient queue management, while minimizing the network duty cycle as well.**

I. Introduction

In Internet of Things (IoT) scenarios, a potentially (very) large number of nodes is able to establish low-power short-range wireless links, thus forming a capillary networking infrastructure that can be connected to the Internet [1]. In order to actualize the IoT vision, IEEE and IETF standardization bodies have recently proposed several interesting protocols that can ease plug and play operations of smart devices in IPv6 networks [2], [3], if properly combined in a communication stack for Low-power and Lossy Networks (LLNs).

From the MAC layer perspective, the IEEE802.15.4e amendment [4] has been released on April 2012 with the aim of redesigning the existing IEEE802.15.4-2011 MAC standard [5] toward a low-power multi-hop MAC, better suitable for the emerging needs of IoT and embedded industrial applications [6], [7], [8]. Among other new features, the Timeslotted Channel Hopping (TSCH) protocol has been proposed to address process automation facilities for oil and gas industry, chemical products, water/waste water treatments, green energy production, climate control [2]. Using TSCH, nodes synchronize on a *slotframe* structure, i.e., a group of slots which repeats over time. Each node follows a schedule which indicates its operative mode (i.e., transmit, receive, or sleep) and, in case it is active, the channel to use and the neighbor to communicate with. The IEEE802.15.4e standard explains how the MAC layer executes a schedule, but it does not specify how such a schedule is built.

At the same time, from a networking point of view, the IETF has recently standardized the Routing Protocol for LLNs (RPL) [9]. It is a gradient-based routing protocol, which can easily organize and manage multi-hop topologies based on short-range low-power links. In the most typical setting, RPL-enabled nodes are connected through multi-hop paths to a small set of LLN sinks, which are usually responsible for data collection and coordination duties. For each of them, a Destination Oriented Directed Acyclic Graph (DODAG) is created by accounting for link costs, node attributes/status information, and an Objective Function, which maps the optimization requirements of the target scenario. In that case, each LLN sink assumes the role of *DODAG root*. A more sophisticated and flexible configuration could contain a single DODAG with a *virtual DODAG root* coordinating several LLN sinks.

The gradient strategy employed by RPL relies on the definition of node's *rank*, which specifies the individual position of a node with respect to its (virtual) DODAG root. As a consequence, the rank should monotonically decrease along the DODAG and towards the destination, in accordance to the gradient-based approach. Specifically, the virtual DODAG root's rank is always 0, while the rank of LLN sinks is 1. In general, for each sensor device, the rank is a real number, whose integer part, r, accounts for the minimum hop-distance from the (virtual) DODAG root. Instead, the decimal part of the rank is used by RPL for selecting the preferred parent among some candidate ones, according to metrics and objective functions. In this paper, we are not concerning the RPL strategy for selecting a preferred parent, hence, for our purposes, the main information is carried into the integer part r of the rank. For this motivation and for the sake of exposition, when referring to rank in what follows, we are considering its integer part r.

Supported by the strong evidence that the IEEE802.15.4e TSCH MAC and the IETF RPL protocols are core to a viable IoT protocol stack [2], we have been facing the problem of building multi-hop schedules for RPL-organized, TSCH-based networks [10]. The first outcome of such investigation has been the Traffic-Aware Scheduling Algorithm (TASA) [11], [12], which builds time/frequency collision-free patterns in a

centralized manner. This strategy allows a master node to collect information about the entire network topology and on the traffic load at each node. Afterwards, such master node computes the schedule for the whole network by exploiting the graph theory's *matching* and *vertex-coloring* techniques. Finally, it provides each node in the network with its time/frequency schedule. Such a centralized approach requires a multi-hop signaling phase, which could impair the power efficiency. Moreover, TASA employs a greedy solution for allocating transmissions for each timeslot. Therefore, TASA does not take into account queue congestion. Finally, such greedy solution finds an optimal time/frequency pattern, whose length is not known a priori: in this sense, supporting topologies with multiple LLN sinks would be not straightforward. Nevertheless, TASA represents a solid ground for comparison in future approaches to packet scheduling in multi-hop TSCH networks.

In this paper, we propose the new Decentralized Traffic Aware Scheduling (DeTAS) algorithm, which, starting from the same requirements of TASA, finds the optimum time/frequency schedule in a decentralized way, while enabling a 1-hop signaling and the queue management of device memory. Remarkably, DeTAS is thought also for multi-sink topologies.

The rest of the paper is organized as follows: we present the DeTAS algorithm in Sec. II and a preliminary performance evaluation with respect to TASA in Sec. III. Finally, in Sec. IV we draw conclusions and future research.

II. Algorithm Description

The suggested DeTAS algorithm is designed for LLN networks organized by RPL in a single DODAG. For this reason, in what follows, we use the RPL terminology [9].

Let $\{n_s\}$, with $s = 1, \ldots, N_S$, be the set of all the N_S sinks, and $\{n_i\}$, with $i = N_S + 1, \ldots, N_S + N$ be the set of source nodes. If $N_S = 1$, the LLN sink n_1 is also a *DODAG root*; otherwise, there is a *virtual DODAG root*, n_0, coordinating all the N_S LLN sinks.

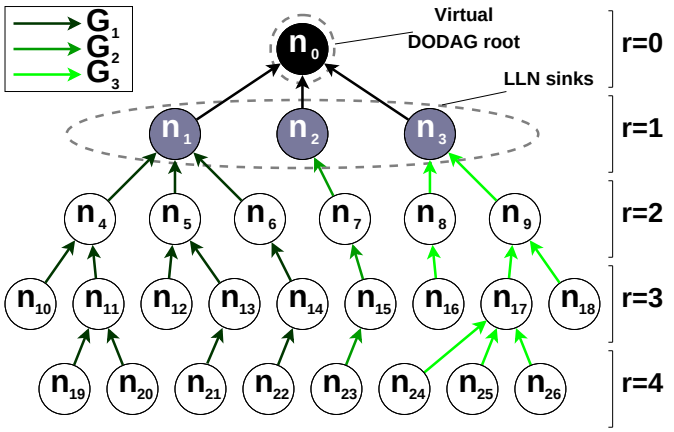

Fig. 1: Example of a LLN with a virtual DODAG root, 3 LLN sinks, and 23 source nodes.

We define *Routing Graph* each destination-oriented tree graph $\mathbf{G_s} = (\mathbf{V_s}, \mathbf{E_s})$, rooted at the LLN sink n_s, with $s = 1, \ldots, N_S$. Note that $\{\mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_S}\}$ is a partition of the set of all nodes (sink and source ones) $\mathbf{V} = \{n_1, \ldots, n_{N_S+N}\}$, and $\mathbf{E_s}$ is the set of directed links between each sensor node n_i and its preferred parent, p_i. Thus, $|\mathbf{E_s}| = |\mathbf{V_s}| - \mathbf{1}$. Furthermore, for each node $n_l \in \mathbf{V}$, with $l = 1, \ldots, N_S + N$, it is possible to identify the set $ch(n_l)$ of its children, and the sub-tree ST_l, composed by n_l itself and all the nodes connected to it through multi-hop paths. It is easy to find that $ST_s \equiv \mathbf{G_s}$, with $s = 1, \ldots, N_S$, i.e., the subtree ST_s rooted at the LLN sink n_s is the *routing graph* $\mathbf{G_s}$ itself. According to the network topology built by RPL, and indicating the rank assigned to a node n_l with r_{n_l}, it results $r_{n_l} = r_{p_l} + 1$.

Figure 1 shows an example network, with the *virtual DODAG root* n_0 coordinating 3 LLN sinks, n_1, n_2 and n_3. In turn, each LLN sink coordinates a *Routing Graph*. In this sense, the 3 *Routing Graphs* $\mathbf{G_1}$, $\mathbf{G_2}$ and $\mathbf{G_3}$ are branches of the same destination-oriented tree graph rooted at n_0.

DeTAS is a traffic-aware algorithm that builds the schedule based on the traffic generated by each source node. We assume that the network supports a multi-point-to-point traffic. In detail, every source node in the set $\{n_i\}$, with $i = N_S + 1, \ldots, N_S + N$, generates a constant[1] integer number of packets, i.e., the *local packet number*, q_i, within a slotframe, destined to one among the LLN sinks in the set $\{n_s\}$, with $s = 1, \ldots, N_S$. For each node $n_l \in \mathbf{V}$, with $l = 1, \ldots, N_S + N$, we define also the *global packet number*, Q_l, as the total amount of packets generated within a slotframe by the nodes belonging to the sub-tree ST_l.

A. A general overview

DeTAS is able to build optimum collision-free schedules for multi-hop IEEE802.15.4e TSCH networks, using a tiny amount of information, locally exchanged among neighbor nodes. At the same time, its decentralized nature allows a distributed computation of such schedule, while keeping very low the amount of signaling messages exchanged among neighbor nodes. Whilst scheduling the traffic, DeTAS manages queue levels, avoiding traffic congestion and, thus, possible packets drops due to overflow of nodes' memory buffer. Finally, DeTAS can exploit the availability of the 16 IEEE802.15.4 frequencies [4] in order to: parallelize several transmissions at the same time, reduce the number of active slots per slotframe (i.e., the network duty cycle), and increase the reliability of wireless links.

In a IEEE802.15.4e TSCH network running DeTAS, all devices are assumed to be synchronized with the same slotframe, having size equal to S time slots. Hence, all nodes follow a common schedule. The latter is built as a *macro*-schedule, given by the combination of a set of *micro*-schedules, one for each *Routing Graph* $\mathbf{G_s}$, with $s = 1, \ldots, N_S$. It has to be

[1] Such an assumption is supported by the fact that traffic is typically different between sensors, but relatively constant over time in emerging heterogeneous embedded applications. Actually, with an efficient signaling, DeTAS could also support variable bit rate traffic flows.

noted that, when $N_S = 1$, the *macro*-schedule includes only the *micro*-schedule related to the *Routing Graph* $\mathbf{G_1}$.

In details, for each *Routing Graph* $\mathbf{G_s}$, with $s = 1, \ldots, N_S$, DeTAS builds a time/frequency *micro*-schedule, using L_s time slots and W_s channel offsets[2]. Such micro-schedule is setup minimizing the number of active slots, L_s, needed for delivering to the LLN sink, n_s, the traffic load generated by the source nodes belonging to $\mathbf{G_s}$, during a single slotframe. More specifically, all transmissions are scheduled in consecutive L_s slots, with $L_s \leq S$, leaving the remaining $S - L_s$ slots empty and, therefore, available for packet transmissions related to other *Routing graphs* or other applications (e.g., other RPL instances). In this manner, the branch of the network represented by the *Routing Graph* $\mathbf{G_s}$, will be able to achieve a duty cycle equal to L_s/S. The length L_s of the *micro*-schedule is computed by the LLN sink n_s using the formulation introduced in Sec.II-C. Instead, the width of the *micro*-schedule, i.e., the number of available channel offsets (and thus the channels), is $W_s = 3$, since frequencies are reused every 3-hops, thus avoiding also collisions due to interference.

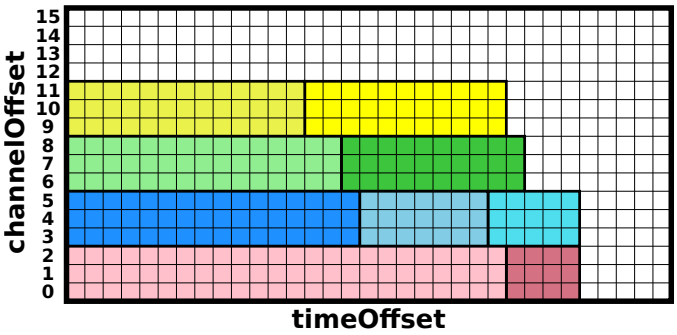

Fig. 2: Displacement of *micro*-schedules within a slotframe structure, using 12 channels.

Afterwards, the *virtual DODAG root*, n_0, collects the length values L_s of each of the N_S *micro*-schedules, as computed from the LLN sinks it coordinates. It arranges such *micro*-schedules into a *macro*-schedule within the network slotframe. Specifically, it makes sure that they do not overlap and they occupy the smallest number of time slots to provide low duty-cycle to the whole *DODAG*. Given that each *micro*-schedule is characterized by a width $W_s = 3$, and assuming that all the 16 IEEE802.15.4 channels are available, it results that maximum 5 *Routing Graphs* can be *micro*-scheduled at the same time within the slotframe, leaving 1 channel offset (and, thus, one frequency) available for broadcast and/or signaling. The *virtual DODAG root* computes the *time offset* and the channel offset shift of each *micro*-schedule, with respect to the start of the slotframe.

More specifically, we assume that the number of channels really used among all the 16 available ones is equal to $3K$, where K is an integer parameter chosen in the interval $[1, 5]$ for *black-listing* purposes (i.e., some channels should not be used if they are more keen to interference and multipath fading [13], [14]). The suggested DeTAS *macro*-scheduling technique collects in a set all the *micro*-schedules' lengths L_s, with $s = 1, ..., N_S$. This set is divided in K subsets, so that the sums computed over such subsets are as close as possible. The problem at hand is known as the *multiprocessor scheduling problem* [15]; we apply the greedy heuristic described in [16] for organizing the *macro*-schedule. Once the partition into K subsets has been done, all the *micro*-schedules belonging to the k-th subset will be activated sequentially, and they will use the values $3(k-1)$, $3(k-1)+1$ and $3(k-1)+2$ as channel offsets. The final *macro*-schedule will have width equal to $3K$ channel offsets and time length equal to the maximum sum among those computed over the K subsets. Fig. 2 shows an example of macro-schedule that use 12 channels (i.e., $K = 4$).

B. Decentralized Micro-scheduling

To setup the *micro*-schedule related to a *Routing Graph* $\mathbf{G_s}$, each source node $n_i \in \mathbf{V_s}$ needs to know the amount of traffic it will receive from its children that should be forwarded to the parent node p_i, toward the LLN sink, n_s. Note that every source node n_i can locally compute its *global packet number*, Q_i, as sum of its *local packet number*, q_i, and the *global packet numbers* of its children, and then forward such information to its parent node, p_i. In particular, the LLN sink, n_s, needs to collect information also about the *local packet number* of its children. Starting from the aforementioned traffic information, exchanged at 1-hop distance, the *micro*-schedule is built in a distributed fashion, with each node, n_i, allocating some slots within the schedule to its children.

The *micro*-schedule is built making sure that no collision occurs in the network. In detail, at any time, the source nodes that can simultaneously transmit belong to the subtrees rooted at only two child nodes of the LLN sink. For instance, considering the LLN sink n_1 in Figure 1 and assuming that n_4 and n_5 are the two selected child nodes at a given time, only the source nodes $\in ST_4$ and $\in ST_5$ can be scheduled for packet transmissions simultaneously. Specifically, a single packet transmission per odd rank is allowed in ST_4 and a single packet transmission per even rank is allowed in ST_5. Furthermore, in order to avoid collision due to interference, the same frequency is reused only after 3-hops. Therefore, each source node n_i will transmit packets to its parent, p_i, using a channel offset equal to $[(r-2) \mod 3]$, and receive packets from its children $\in ch(n_i)$ with a channel offset equal to $[(r-1) \mod 3]$.

To avoid buffer overflow, in DeTAS each node n_i assigns an alternate sequence of "transmit" and "receive" slots (i.e., slots for transmission and reception of packets, respectively) to every child node. In this way, a queue is emptied of 1 packet, as soon as it receives a new one, and viceversa. In detail, $2Q_i$ time slots will be allocated to each node n_i, given that it will receives $Q_i - q_i$ packets from the other nodes $\in ST_i$ and transmits Q_i packets to its parent p_i.

[2]In IEEE802.15.4e, the channel offset is related to the channel hopping procedure and it is translated into a given frequency; the couple (time slot, channel offset) defines the channel used for communications.

Obviously, to allow packets to be correctly sent and received, the transmit slots of the children $\in ch(n_i)$ must be synchronized with the receive slots of the parent, p_i. Thus, it follows that a sub-tree ST_i rooted at a node n_i can be *even*- or *odd*-scheduled, i.e., it can follow an *even* or *odd micro-schedule*.

Definition 1. *A subtree ST_i is* even-scheduled, *if all nodes $\in ST_i$ with* even r *transmit in* even *time slots, indicating with 0 the first slot in the schedule, and those with* odd r *transmit in* odd *time slots.*

Definition 2. *A subtree ST_i is* odd-scheduled, *if all nodes $\in ST_i$ with* even r *transmit in* odd *time slots, indicating with 0 the first slot in the schedule, and those with* odd r *transmit in* even *time slots.*

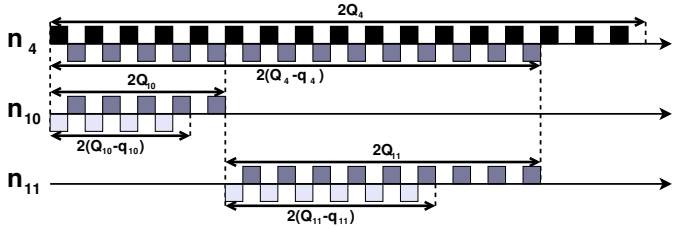

Fig. 3: *Even micro*-schedule for nodes belonging to ST_4 of DODAG in Fig. 1.

Fig. 3 shows an *even micro-schedule* followed by the sub-tree ST_4 of the DODAG in Fig. 1, and, details the time slots reserved to the nodes n_4, n_{10} and n_{11}. We can see that n_4, having *even* r spends the Q_i *even* time slots in transmission, the first $Q_i - q_i$ *odd* time slots in reception, and the last q_i *odd* time slots in idle (having already received all the packets from its children n_{10} and n_{11}).

C. Micro-schedule length L_s

Even though the *micro*-schedule is built with a distributed approach, it is initialized by the LLN sink, n_s, that computes the length L_s of the *micro*-schedule, and selects, among the sub-trees rooted at its children, the ones to be *even*-scheduled, and thus, those to be *odd*-scheduled.

Being DeTAS a traffic-aware scheduling algorithm, the length, L_s, of the *micro*-schedule is a function of the network traffic. In fact, the LLN sink can receive at most one packet per time slot (i.e., $L_s \geq Q_s$). At the same time, the child of the LLN sink, referred hereafter as n_M, having the maximum *global packet number*, i.e., $Q_M = \max_{n_i \in ch(n_s)} Q_i$, will need a schedule long enough for containing Q_M transmit slots and $Q_M - q_M$ receive slots, i.e., $L_s \geq 2Q_M - q_M$. For every randomly scattered physical topology, the proposed DeTAS scheme is able to find the optimum schedule with the minimum length, given by:

$$L_s = \max\ \{2Q_M - q_M,\ Q_s\}\ . \tag{1}$$

To this aim, DeTAS simultaneously guarantees that a single sub-tree rooted at a sink' child is *even*-scheduled, while the sub-tree rooted at another sink' child is *odd*-scheduled, thus, they will not incur in any kind of collision, since the related schedules are perfectly overlapped. Therefore, a LLN sink running DeTAS must divide its children in two lists, i.e., an *even* list and an *odd* one. The sub-trees rooted at the children in the *even* list could be sequentially *even*-scheduled, for a total *even*-schedule with a length equal to L_s^e time slots. At the same time, the sub-trees rooted at the children in the *odd* list could be sequentially *odd*-scheduled, for a total *odd*-schedule with a length equal to L_s^o time slots. Obviously, those lists should be load balanced, since the schedule length L_s will be equal to the maximum between L_s^e and L_s^o. This load balancing problem falls into the class of *multiprocessor scheduling problems* [15]. The greedy heuristic employed in this paper is the same described in [16]: the sink' children are ordered in a descending order, according to their *global packet number*. Then, they are appended subsequently to the list (*even* or *odd*) with the current smallest sum of *global packet numbers*.

DeTAS assumes a different behavior depending on the traffic loads of the children of the LLN sink. Specifically, if $Q_M \geq Q_s/2$, the node n_M will be the only one in the *even*-list. DeTAS will *odd*-schedules subsequently the sub-tree rooted at the nodes in the *odd*-list. At the same time, it will *even*-schedule the sub-tree ST_M for exactly $2(Q_M - \alpha)$ slots, where:

$$\alpha = \min\{2(Q_s - Q_M), q_M\}. \tag{2}$$

It can be shown that, after this phase, the number of packets to be still received by the LLN sink would be equal to α, and that those packets would be located on n_M. Hence, n_M will allocate the following α consecutive time slots for transmitting the pending packets to the LLN sink. It can be verified that eq. (1) is always fulfilled. In such case, the node n_M should be informed about α, in order to compute accordingly the instant when switching from the *even*-scheduling mode to the continuos one.

When $Q_M < Q_s/2$, DeTAS reduces the discrepancy between the sum Q_s^e, computed over the *global packet numbers* related to the *even*-list, and the sum Q_s^o, computed over the *odd*-list, to 1 (if Q_s is odd) or 0 (if Q_s is even). It is worth to note that $Q_s^e + Q_s^o = Q_s$. The LLN sink computes the value β, which levels off the two lists:

$$\beta = \left\lfloor \frac{Q_s^e - Q_s^o}{2} \right\rfloor \tag{3}$$

and it selects the first child in the list with the maximum associated sum of *global packets numbers*, referred hereafter as n_{cut}. If $\beta \geq 0$ ($\beta < 0$)[3], DeTAS will *even*-schedule (*odd*-schedule) firstly the sub-tree ST_{cut} for $2(Q_{cut} - |\beta|)$ time slots and, then, all the sub-trees related to the other nodes appended in the *even*-list (*odd*-list). Simultaneously, it will *odd*-schedule (*even*-schedule) firstly all the sub-trees related to the nodes in the *odd*-list (*even*-list), then the sub-tree ST_{cut}

[3]In the rest of the paragraph, we indicate the alternative case and the related settings in brackets.

for exactly $2|\beta|$ time slots. It can be shown that the *micro*-schedule length is always $L_s = Q_s$, according to eq. (1). In this traffic load conditions, n_{cut} should be informed by the LLN sink about when to *even*- and when to *odd*-schedule ST_{cut}. This information must be accordingly updated and propagated along ST_{cut}.

III. Performance evaluation

Herein, some preliminary simulation results are present in order to show the effectiveness of the proposed scheme in terms of maximum queue occupation in the source nodes. We highlight that the maximum queue occupation has been selected as performance parameter because it indicates how much traffic load can be managed by source nodes, avoiding packet discards due to possible buffer overflows.

Results obtained with DeTAS have been compared with the ones achieved exploiting the TASA algorithm which, as known, has very good performance in IEEE802.15.4e networks [11]. We consider only single sink topologies, i.e., with a single LLN sink acting as *DODAG root*, in order to picture also a fair comparison between DeTAS and TASA.

Results have been obtained by using an ad-hoc simulator written with the Python language. We have simulated a TSCH network with a tree topology, deployed in an area of 200×200 m^2. Each node, located in a random position in the network, has a coverage range $R = 50$ m. Since DeTAS builds the schedule based on the network topology, we analyze the performance in several scenarios, varying the total number of nodes, N in the interval [30, 150]. For each network size, we have also considered 25 different random displacements for the source nodes.

Given that DeTAS allocates time slots and channel offsets based not only on the network topology, but also on the traffic load. We test the algorithm under different traffic conditions: we assume that the average number of packets, $\bar{n}_{pkt}$, generated by each node within a single slotframe can vary between 3 and 5. Specifically, we have imposed that, at the beginning of the slotframe, each node generates a number of packets, randomly selected in the range [1, 5] (if $\bar{n}_{pkt} = 3$) and [1, 9] (if $\bar{n}_{pkt} = 5$). To better characterize the performance comparison, for each average number of packets, we have also considered 25 different random arrangements for the traffic load on each source node. We further assume that the slotframe size S is as large as to contain the whole schedules computed by DeTAS. Finally, we assume for TASA a number of 3 available channels.

Figures 4, 5 and 6, show the maximum queue occupation as function of the rank of the source nodes, in three different network scenario, having respectively 30, 90, and 150 source nodes. These results have been obtained, averaging on 625 simulations runs, derived from the combination of 25 different network-topologies and 25 different traffic load sets.

From the above figures, we can observe how DeTAS queue management provides a bounded quasi-deterministic queue occupation, since the standard deviation is close to 0.

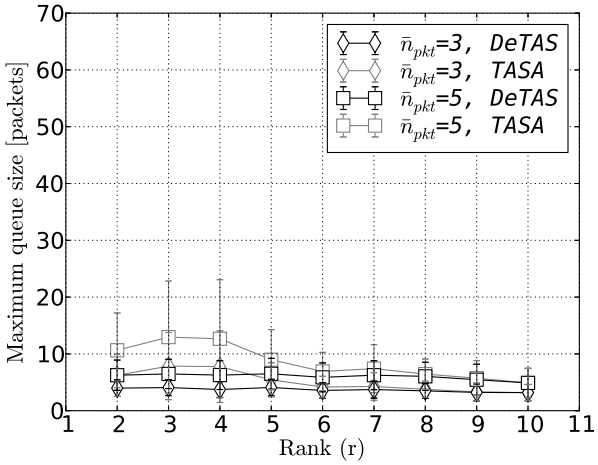

Fig. 4: Maximum queue occupation as function of r, in LLNs with 30 source nodes.

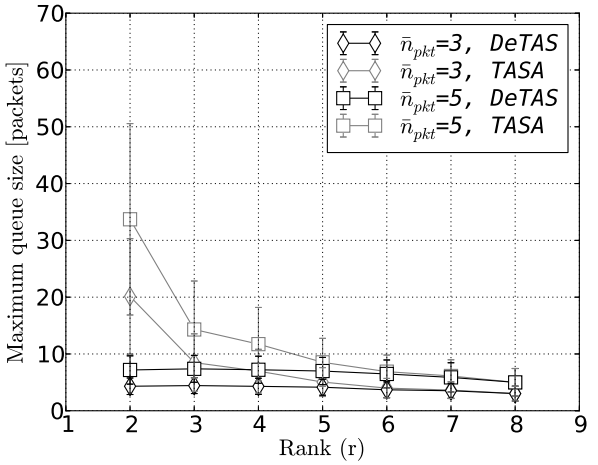

Fig. 5: Maximum queue occupation as function of r, in LLNs with 90 source nodes.

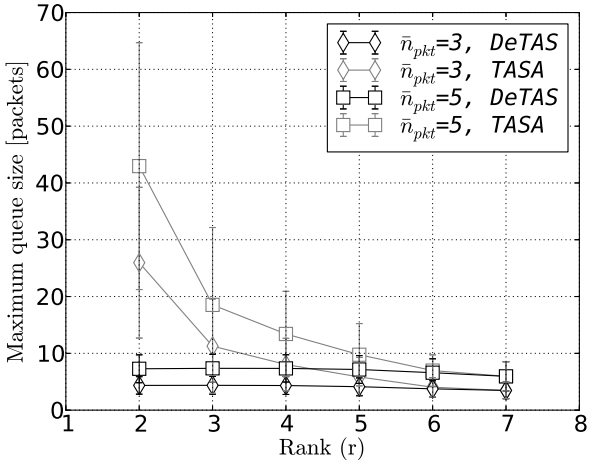

Fig. 6: Maximum queue occupation as function of r, in LLNs with 150 source nodes.

Moreover, DeTAS always maintains the queue occupation close to 2 times the average number of packets generated on each node (i.e., the maximum number of packets that a node can generate within a slotframe). Such a queue occupation is quite constant independently on the node ranks and on the network size. This inference let us guess that DeTAS would totally avoid packet drops due to buffer overflows, allowing the use of smart devices with smaller memories.

DeTAS shows also a great improvement with respect to a centralized solution like TASA. In fact, the TASA maximum buffer occupation increases with a decreasing rank. This means that nodes close to the sink are more keen to buffer overflows. In this sense, the maximum queue occupation on average can be more than 4 times greater than the queue occupation at the beginning of the slotframe.

Focusing on the nodes with smaller ranks, although the behaviors of DeTAS and TASA are almost indistinguishable in small-sized networks (see Figure 4), a thoroughgoing difference is evident for larger networks (Figures 5 and 6), since the DeTAS performance in terms of queue occupation remains constant regardless of the network size. Moreover, varying the average traffic load $\bar{n}_{pkt}$ a very strong difference can be appreciated between DeTAS and TASA: the DeTAS and TASA queue occupation respectively increase proportionally to $\bar{n}_{pkt}$ with a factor 2 and 6. It is easy to find also that the very good queue management of DeTAS translates also in a queueing delay extremely reduced w.r.t. TASA.

IV. Conclusion

In this paper, we have witnessed how a decentralized solution better suits the multi-hop scheduling problem issued in RPL-enabled IEEE802.15.4e TSCH networks. To this aim, a new scheduling scheme, DeTAS, has been introduced and described in details. Enabling a lightweight 1-hop signaling, the presented technique naturally addresses some pendent issues that limit an actual deployment of IoT-compliant multi-hop LLNs. Furthermore, DeTAS exploits very well the time/frequency resources available with TSCH, being already suitable for complex LLN topologies. The reported preliminary simulation results show the effectiveness of the proposed approach in terms of queue management of the nodes' memory buffer. Moreover, they strongly encourages the use of RPL for organizing multi-hop IEEE802.15.4e TSCH networks. In order to better characterize the performance of the DeTAS distribution scheme, future research works will quantify and discuss the signalling overhead. Furthermore, DeTAS will account for the possibility that a node can forward packets towards the LLN sinks using a set of candidate next-hops as relayers, rather than a single preferred parent, thus load-balancing the routing tree and improving the end-to-end reliability through path diversity. Finally, we will deal with experimental tests of the stack employed in this paper, when corroborated by the DeTAS powerful features.

Acknowledgment

This work was supported by the PON projects (ERMES-01-03113, DSS-01-02499, EURO6-01-02238) funded by the Italian MIUR, the Apulia Region Project PS 025, and the FP7 Project OUTSMART-285038, partially funded by the European Community.

References

[1] O. Hersent, D. Boswarthick, and O. Elloumi, *The Internet of Things: Key Applications and Protocols*, 2nd ed. Wiley, 2012.

[2] M. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of (Important) Things," *Communications Surveys & Tutorials, IEEE*, 2012.

[3] J. Hui and P. Thubert, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*, RFC 6282, IETF RFC 6282, September 2011.

[4] *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., 16 April 2012.

[5] IEEE std. 802.15.4, *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, Standard for Information Technology Std., 16 June 2011.

[6] J. P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.

[7] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*, ser. Wiley Series on Communications Networking & Distributed Systems. John Wiley & Sons, 2010.

[8] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of Things: Vision, Applications & Research Challenges," *Ad Hoc Networks*, 2012.

[9] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, RFC 6550, IETF RFC 6550, March 2012.

[10] N. Accettura, M. R. Palattella, M. Dohler, L. A. Grieco, and G. Boggia, "Standardized Power-Efficient & Internet-Enabled Communication Stack for Capillary M2M Networks," in *IEEE Wireless Communications and Networking Conference, WCNC*, April 2012.

[11] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and 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.

[12] ——, "Traffic-Aware Time-Critical Scheduling in Heavily Duty-Cycled IEEE 802.15.4e for an Industrial IoT," in *IEEE Sensors*, October 2012.

[13] B. Kerkez, T. Watteyne, and M. Magliocco, "Feasibility Analysis of Controller Design for Adaptive Channel Hopping," in *ICST International Workshop on Performance Methodologies and Tools for Wireless Sensor Networks, WSNPERF*, October 2009.

[14] T. Watteyne, A. Mehta, and K. S. J. Pister, "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense," in *Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN*, October 2009.

[15] V. Sarkar, *Partitioning and scheduling parallel programs for multiprocessors*. MIT press, 1989.

[16] R. E. Korf, "Multi-way number partitioning," in *Proceedings of the 21st international jont conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., 2009, pp. 538–543.