A Performance Analysis of Orchestra Scheduling for TSCH networks

Article · August 2017

DOI: 10.1002/itl2.4

4 authors:

Sana Rekik
Ecole Nationale d'Ingénieurs de Sfax
8 PUBLICATIONS **40** CITATIONS
SEE PROFILE

Nouha Baccour
Ecole Nationale d'Ingénieurs de Sfax
29 PUBLICATIONS **896** CITATIONS
SEE PROFILE

Mohamed Jmaiel
Digital Research Center of Sfax
316 PUBLICATIONS **1,381** CITATIONS
SEE PROFILE

Khalil Drira
Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)
184 PUBLICATIONS **956** CITATIONS
SEE PROFILE

A Performance Analysis of Orchestra Scheduling for TSCH networks

Sana Rekik, Nouha Baccour, Mohamed Jmaiel
ReDCAD Laboratory
University of Sfax, Tunisia
{sana.rekik, nouha.baccour, mohamed.jmaiel}@redcad.org

Khalil Drira
LAAS-CNRS
Univ. de Toulouse, France
khalil.drira@laas.fr

Abstract—**The IEEE 802.15.4 TSCH (Time Slotted Channel Hopping) represents the latest generation of low-power and highly reliable MAC protocols. It orchestrates the medium access according to a time-frequency communication schedule. However, TSCH specification does not provide any practical solution for the establishment of the schedule. Orchestra is a recent scheduling solution for TSCH that brings significant advantages such as, the use of simple scheduling rules, the low signaling overhead, and the high delivery ratio it is able to fulfill. In this paper, we investigate the use of TSCH protocol with Orchestra scheduling approach for the Internet of Things (IoT) applications. Performance analysis results demonstrate that despite its unique features, Orchestra may incur high end-to-end communication delay, especially when the traffic is not uniformly distributed in the network. This limitation makes Orchestra not sufficiently convenient for several delay-sensitive IoT applications, such as smart grid applications.**

Keywords— Wireless Sensor Network, TSCH protocol, Orchestra scheduling, performance evaluation.

I. Introduction

Currently, Wireless Sensor Networks (WSNs) are recognized as a promising communication technology for enabling the IoT, where sensor devices - the ”things” - gather data readings from the physical field and deliver data to the Internet. However, the deployment of WSNs in IoT applications brings new challenges as these applications have stringent requirements, mainly on latency, reliability and lifetime.

MAC protocols play a crucial role to meet the requirements of IoT applications. The TSCH protocol, part of the IEEE 802.15.4 standard [1], represents the latest generation of low-power and highly reliable MAC protocols. It has been specifically proposed to meet the requirements of industrial applications. Further, the IEEE 802.15.4 TSCH along with the IETF 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) [2] and RPL (IPv6 Routing Protocol for Low Power and Lossy Networks) [3] protocols are considered as key building blocks of the emerging 6tisch architecture [4], under standardization within the IETF 6tisch Working Group.

With TSCH, channel hopping is added on top of time slotted MAC in order to counteract frequency selective fading and improve the reliability of radio links. TSCH orchestrates the medium access according to a communication schedule that indicates to each node what to do in each slot and frequency channel: transmit, receive, or sleep. The IEEE 802.15.4 standard only specifies how the MAC layer executes the schedule. However, it does not define how to establish it. Therefore, several scheduling algorithms have been proposed for TSCH networks. They can be broadly classified as either centralized [5]–[7] or distributed [8]–[10] algorithms. Both centralized and distributed scheduling techniques involve additional communication overhead: each node is supposed to communicate with its neighbors or with a central entity, to exchange network and traffic information in addition to scheduling information. Orchestra [11] is a recent scheduling technique that is neither centralized nor distributed. Rather, it is referred to autonomous scheduling technique as each node autonomously builds its own schedule without any negotiation with its neighbors. The schedule is computed based on available routing information and is automatically updated whenever the routing topology evolves. Orchestra presents several strengths making it a promising scheduling solution for TSCH networks. It is able to build schedules, using simple scheduling rules, without triggering a signaling overhead.

In this paper, we analyze the performance of TSCH MAC, using Orchestra for the establishment of the schedule, in order to assess its adequacy for WSNs in IoT applications. Simulation results show that Orchestra-based TSCH protocol provides high reliability but may experience high end-to-end delay, particularly when the traffic is not uniformly distributed in the network. Indeed, the amount of data traffic to deliver to the root depends on the sensor node location in the routing tree-like topology. Generally, the more the node is close to the root the more data it has to forward. However, Orchestra allocates the same number of time slots to all nodes, which leads to additional communication delays due to packets enqueuing in the buffers of congested nodes. This limitation makes Orchestra not sufficiently convenient especially for several IoT applications where low latency is a key requirement.

The rest of this paper is organized as follows. In the next section, a discussion of recent works related to TSCH scheduling algorithms is given. We provide an overview of Orchestra scheduling approach in Section III. In Section IV, we analyze the performance of Orchestra-based TSCH protocol. Finally, we conclude in Section V.

II. Related work

Several scheduling algorithms have been proposed for TSCH networks. They can be broadly classified as either centralized [5]–[7] or distributed [8]–[10] algorithms.

Generally, centralized algorithms assume that a central entity has a full knowledge of the routing topology, the physical connectivity, and the traffic load of each node, which allows building then disseminating the schedule to all the nodes in the network. Unfortunately, centralized scheduling algorithms could not be suitable for TSCH networks deployed in the IoT: first, they can present scalability issues, which is one of the requirements of WSN-based IoT applications. Second, they result in considerable signaling overhead as any change in the network state should be communicated to the central entity to re-compute and redistribute new schedules.

Distributed scheduling algorithms have been proposed to resolve these problems. DeTAS (Decentralized Traffic Aware Scheduling) [8] is a distributed algorithm that targets RPL networks with several sink nodes. The network is modeled as multiple routing graphs, each rooted at a different sink. For each graph, DeTAS builds a micro-schedule accommodating the transmissions of the nodes belonging to it. The micro-schedules are combined into a global macro-schedule.

Wave [9] is another example of distributed scheduling algorithms for TSCH, were the schedule is build by computing a series of waves. The sink node sends a message to its children to trigger the computation of the first wave. Each node having received the message selects a cell (a time slot and a channel) in the wave to transmit its first packet. Then, it notifies this assignment to its conflicting nodes. Once computed, the first wave constitutes the (time slot, channel) pattern: each successive wave is an optimized copy of the first wave, where the cells that are no more needed (i.e., that does not contain transmissions) are removed.

In [10], the authors proposed a distributed TSCH schedule, based on the known PID (Proportional Integral and Derivative) control algorithm. The proposed schedule adjusts the number of allocated slots to the traffic demand and reacts to sudden traffic variations (such as bursty traffic) by adding/removing cells in the schedule using 6top protocol [12].

The OTF (On-the-Fly) bandwidth reservation [13] is another distributed scheduler that aims to dynamically adjust the node's schedule: based on the number of packets being sent to a given neighbor and the number of cells already reserved to that neighbor, OTF estimates the number of cells to be added/removed (to that neighbor). Then, it asks the 6top protocol to allocate/deallocate these cells, which triggers a negotiation process between neighbor nodes.

Generally, in distributed scheduling solutions, each node computes its local schedule based on information exchanged with its neighbors, such as topology and traffic information, in addition to scheduling information. However, distributed scheduling algorithms reduce but do not eliminate the signaling overhead compared to centralized approaches.

Orchestra [11] is a recent scheduling technique that is neither centralized nor distributed (refer to Section III). Rather, it is referred to autonomous scheduling technique as each node autonomously builds its own schedule without any negotiation with its neighbors. The schedule is computed based on available routing information, and is automatically updated whenever the routing topology evolves. Orchestra presents several strengths making it a promising scheduling solution for TSCH networks. It is able to build schedules, using simple scheduling rules, without triggering a signaling overhead. Further, it was proven that Orchestra can achieve high delivery ratio [11]. These features motivate us to investigate the performance of TSCH MAC protocol with Orchestra scheduling approach for WSNs in IoT applications.

III. Overview of Orchestra

Orchestra is a scheduling approach for TSCH and RPL networks. The RPL routing protocol [3] organizes the network in form of one or multiple Destination-Oriented Directed Acyclic Graphs (DODAGs), each one rooted towards a sink node denoted as DODAG root. The 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. Although RPL can manage several kinds of traffic flows (to and from the DODAG root or between any pair of nodes in the network), we have focused on the dominant multipoint-to-point traffic, i.e., flowing from the nodes in the network towards the DODAG root, which is more related to monitoring applications in industrial environments.

The Orchestra schedule is computed in an autonomous fashion, where each node in the network locally and autonomously maintains its own schedule based on information from the routing layer. Specifically, each node's schedule consists of different slotframes having different lengths (periods). Each slotframe is assigned to a particular traffic plane: TSCH MAC, RPL routing, and application. The MAC slotframe schedules the transmissions of TSCH beacons that ensure TSCH association and parent-child synchronization. The routing slotframe schedules the transmissions of broadcast RPL signaling, while the application slotframe is used for unicast data traffic transmissions from any node to its RPL parent. The slotframe lengths are mutually prime (ensuring the slotframes cycle independently). Each slotframe has a unique identifier (denoted handle), such as the smaller the handle, the higher the priority of the slotframe. If slots from different slotframes overlap, the slot of the highest priority slotframe takes precedence. Figure 1 illustrates an example of Orchestra schedule containing three slotframes – MAC, routing and application slotframes– of lengths 5, 3, and 2 slots respectively. In this example, priority increases from top to bottom.

Each slotframe is composed of slots that can be dedicated (i.e., contention-free) or shared (i.e., contention-based with CSMA back-off). The slot coordinates (time and channel offset) can be either fixed or variable —a function of the receiver/sender identifier. Generally, Orchestra identifies the following four types of slots:

- Common Shared (CS) Slots: installed at fixed time and channel offset and used for RPL beacons. At every node, a CS slot results in a single shared slot used by all the nodes in the network for both transmission and reception.

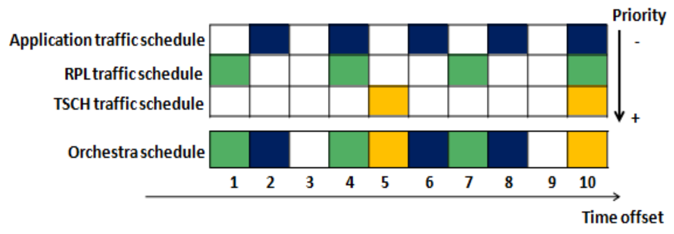

Fig. 1. Orchestra slotframes illustration (the slotframes lengths are 2, 3 and 5 corresponding to the application, routing and MAC slotframes respectively).

- Receiver-based Shared (RBS) Slots: installed at coordinates that are derived from the identifier of the receiver node. Typically, the RBS are used for child-to-parent communication (i.e., application slotframe). At each node, a receiver-based shared slot consists in one reception (Rx) slot, which offset is computed as the node (receiver) identifier modulo the slotframe length, and one transmission (Tx) slot per neighbor, which offset is computed as the neighbor identifier modulo the slotframe length.
- Sender-based Shared (SBS) Slots: installed at coordinates derived from the identifier of sender node. At each node, a SBS slot consists in one Rx slot per neighbor, which offset is computed based on the neighbor identifier, and one Tx slot computed based on the node (sender) identifier.
- Sender-based Dedicated (SBD) Slots: slightly different from the SBS, in the sense that they use dedicated slots (instead of shared), and thus they result in contention-free transmissions.

In this paper, we use an Orchestra setup consisting of three slotframes, described as follows

- MAC slotframe: has the highest priority. It is made of one SBD slot, so that TSCH beacons transmissions are contention-free. It has 397 slots long, as proposed in [11].
- Routing slotframe: has a lower priority than the MAC slotframe. The RPL broadcast messages transmissions take place in one CS slot. The default slotframe length proposed in [11] is 31 slots.
- Application slotframe: has the lowest priority. It consists of a RBS used for unicast data transmissions. We fix the application slotframe length to 11 slots as it was demonstrated in [11] that the contention rates of RBS increases at longer slotframes.

IV. Performance analysis

In this section, we analyze the performance of TSCH based on Orchestra for schedule establishment, using COOJA [14] simulator, available as part of Contiki Operating System. The main performance metrics we focus on are the reliability and the end-to-end delay.

We have considered an RPL organized and TSCH-based network. In our TSCH implementation, we used 15 ms as time slot duration and 8 retries as maximum number of packet retransmissions. The nodes are deployed in a uniform grid topology. We varied the number of nodes from 49 (i.e., 7x7 grid) to 100 (i.e., 10x10 grid). The root node is located at coordinates (0,0) and the grid unit is equal to 30 meters. All the nodes in COOJA were configured as Tmote Sky motes whose transmission power is set to 31 dBm. The data rate is equal to 2 packets per minute. To enable the establishment of the topology, the nodes begin the transmission of data packets after a delay of 2 minutes. Each simulation run lasts 1 hour to ensure convergence to a steady state. Simulation results are provided with a 90% of confidence interval.

To evaluate the sensitivity of Orchestra to traffic demand, we investigate the performance of TSCH with Orchestra, under different traffic loads. Two different scenarios are considered:

- In the first scenario, only leaf nodes generate data packets and the rest of the nodes just forward these packets to the root node.
- In the second scenario, all the nodes are data sources except the root node. This scenario leads to congestion at nodes close to the root node, the so-called funneling effect. Hence, this scenario allows analyzing the protocol performance when the traffic is not uniformly distributed in the network.

Figure 2 and Figure 3 show the average delivery ratio and the average end-to-end delay of Orchestra in both scenarios, respectively. It can be observed that Orchestra provides high reliability (Figure 2) but it can incur high packet delays as shown in Figure 3. From this figure, it is clear that scenario 2 leads to a much higher communication delay compared to scenario 1. Further, in scenario 2 the delay increases exponentially with the increase of number of nodes, in contrast to scenario 1 where the delay is almost constant. These results are justified as follow: in Orchestra schedule, each node reserves one Rx slot and one Tx slot per slotframe, independently from the amount of data traffic it has to deliver to the root node. However, this traffic load varies according to the node location in the routing tree-like topology. For instance, nodes close to the root have higher traffic load than leaf nodes. Hence, in scenario 2, where all nodes are data sources, there was an accumulation of packets in the queues of nodes close to the root due to their high load. Consequently, each of these nodes required several slotframes (one slotframe provides only one Tx slot) to empty its queue. These phenomena lead to unwanted delays at these particular nodes, which negatively affects the overall communication delay.

To confirm the above observations, the distribution of traffic loads in the network has been analyzed. To reflect the traffic load at a particular node, we use the maximum number of packets in its queue (the peak transmission queue). Figure 4 illustrates the cumulative distribution function (CDF) of traffic loads in the networks for both scenarios. Note that we only show the CDF curve for 81 nodes (Figure 4). The remaining CDF curves corresponding to 49, 64, and 100 network sizes are omitted due to lack of space. From Figure 4, it can be observed that in scenario 1, the queues are almost empty: the reserved slots are sufficient to send packets generated by leaf nodes without waiting in queues. However, in scenario 2, the

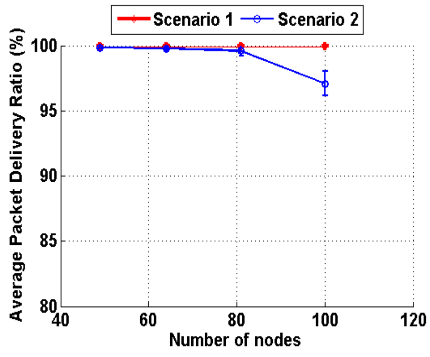

Fig. 2. End-to-end reliability of Orchestra-based TSCH, as a function of the number of nodes.

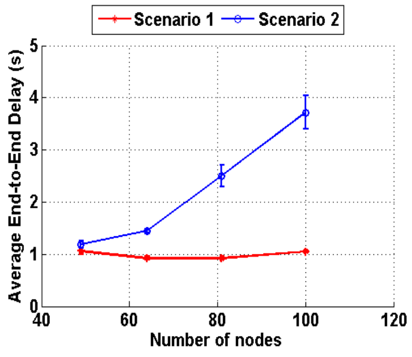

Fig. 3. End-to-end delay of Orchestra-based TSCH, as a function of the number of nodes.

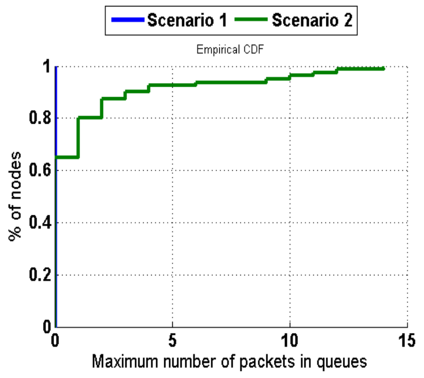

Fig. 4. Empirical CDFs of peak transmission queue — The number of nodes is fixed to 81.

number of packets in queues is between 0 and 14. Further, almost 5% of nodes has more than 10 packets in their queues. This congestion at 5% of nodes — those close to the root, affects the end-to-end communication delay in the network. For instance, given a slotframe having 11 slots long, where the duration of a single slot is 15 ms, a node having 10 packets in its queue requires at least (without re-transmissions) 10 slotframes to empty its queue. Thus, the minimum delay to deliver the packets in the queue is 1,65 s. This delay is definitely unacceptable for several critical IoT applications. For instance, in smart grid, the substation automation monitoring application, sensed data must be received within a delay of 0,2 s in order to make the necessary protection actions [15].

In summary, analysis results show that despite its outstanding features, Orchestra has the limitation of computing the TSCH schedule at each node, independently from its traffic load/demand, which can drastically affect the communication delay. Therefore, a new protocol that takes advantage of Orchestra strengths yet improves the communication delay needs to be designed in order to meet the requirements of delay-sensitive IoT applications.

V. Conclusion

In this paper, we have conducted a performance analysis of Orchestra-based TSCH networks. Analysis shows that the network may experience high end-to-end communication delay mainly because the schedule is not adaptive to traffic demand and some nodes may suffer from high number of queued packets. Based on these results, we have concluded that a new protocol needs to be designed to take advantage of Orchestra strengths and lift its restraints. The new protocol should be aware of the traffic demand of each node by adapting the number of communication slots to the node's traffic demand.

References

[1] WG802.15, "IEEE 802.15.4-2015 - IEEE standard for low-rate wireless networks," 2016.

[2] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-power Wireless Personal Area Networks (6LoWPANs)," 2012.

[3] T. Winter, "RPL: IPv6 routing protocol for low-power and lossy networks," 2012.

[4] P. Thubert, "An architecture for IPv6 over the TSCH mode of IEEE 802.15.4, draft-ietf-6tisch-architecture-11," January 27, 2017.

[5] 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 2012, Sydney, Australia, September 9-12*, 2012, pp. 327–332.

[6] R. Soua, P. Minet, and E. Livolant, "MODESA: an optimized multichannel slot assignment for raw data convergecast in wireless sensor networks," in *31st IEEE International Performance Computing and Communications Conference, IPCCC 2012, Austin, TX, USA, December 1-3*, 2012, pp. 91–100.

[7] R. Soua, E. Livolant, and P. Minet, "MUSIKA: A multichannel multisink data gathering algorithm in wireless sensor networks," in *9th International Wireless Communications and Mobile Computing Conference, IWCMC 2013, Sardinia, Italy, July 1-5*, 2013, pp. 1370–1375.

[8] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized traffic aware scheduling in 6tisch networks: Design and experimental evaluation," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 455–470, 2015.

[9] R. Soua, P. Minet, and E. Livolant, "Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15.4e TSCH networks," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 4, pp. 557–575, 2016.

[10] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne, "Distributed PID-based scheduling for 6tisch networks," *IEEE Communications Letters*, vol. 20, no. 5, pp. 1006–1009, 2016.

[11] S. Duquennoy, B. A. Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *13th ACM Conference on Embedded Networked Sensor Systems, SenSys 2015, Seoul, South Korea, November 1-4*, 2015, pp. 337–350.

[12] Q. Wang and X. Vilajosana, "6tisch operation sublayer (6top) interface. draft," *IETF, July*, 2015.

[13] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-fly bandwidth reservation for 6tisch wireless industrial networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, 2016.

[14] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *LCN 2006, The 31st Annual IEEE Conference on Local Computer Networks, Tampa, Florida, USA, 14-16 November 2006*, 2006, pp. 641–648.

[15] V. C. Gungor, D. Sahin, T. Koçak, S. Ergüt, C. Buccella, C. Cecati, and G. P. Hancke, "A survey on smart grid potential applications and communication requirements," *IEEE Trans. Industrial Informatics*, vol. 9, no. 1, pp. 28–42, 2013.