Forte: Hybrid Traffic-Aware Scheduling for Mobile TSCH Nodes

Iliar Rabet*‡, Hossein Fotouhi*, Mário Alves †, Maryam Vahabi*, and Mats Björkman*
* *Mälardalen University*, Västerås, Sweden † *Politécnico do Porto (ISEP)*, Portugal ‡ *Afry*, Västerås, Sweden
Email: * {iliar.rabet, hossein.fotouhi, maryam.vahabi, mats.bjorkman}@mdu.se, † mjf@isep.ipp.pt

2024 IEEE 49th Conference on Local Computer Networks (LCN) | 979-8-3503-8800-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/LCN60385.2024.10639734

***Abstract*—Applications of the Internet of Things (IoT), particularly within Industrial IoT, impose stricter reliability and efficiency requirements on low-power wireless technologies. This has driven the creation of new medium access protocols, such as *Time Slotted Channel Hopping* (TSCH). Recently, autonomous schedulers, which manage wireless links without node negotiation, are gaining popularity due to their lightweight and reliable operation. However, challenges arise with node mobility and dynamic traffic, as current schedulers use a static allocation method. To overcome this gap, we propose *Forte*, a hybrid scheduler that combines autonomous scheduling for basic connectivity with a centralized on-demand scheduler that allocates extra timeslots and frequency channels so that nodes adapt to the dynamic requirements. The centralized module formulates a Lyapunov optimization to guarantee queue stability while minimizing negotiation overhead and nodes' duty-cycles. *Forte* outperforms the state-of-the-art by reducing packet end-to-end delay and increasing packet delivery ratio, all with minimal duty-cycle increase.**

***Index Terms*—6loWPAN, RPL, Time Slotted Channel Hopping (SCH), Mobility, Internet of Things (IoT), IEEE 802.15.4, Orchestra, Contiki, COOJA**

I. Introduction

Some *Internet of Things* (IoT) applications such as in the context of Industrial IoT may impose more stringent reliability, timing and scalability requirements while communicating over *Low-power Lossy Networks (LLN)s*. In this context, IEEE 802.15.4-2015 standardized the *Time-Slotted Channel Hopping* (TSCH) protocol, which enables nodes to avoid interference and collisions by leveraging time synchronization and channel hopping, based on an agreed-upon schedule between participating nodes [1]. This schedule significantly impacts the behavior of the network and the energy consumption of the nodes.

The TSCH standard does impose any specific scheduling mechanism, leaving this open to optimize based on the requirements. Therefore, nodes can negotiate in a distributed or centralized manner. Another interesting class of schedulers is the one of *autonomous schedulers*, such the one we consider as our benchmark - *Orchestra* [2]. In *Orchestra*, nodes use locally available information to agree on the schedule, thus avoiding negotiations with their neighbors. This was a relevant improvement against previous approaches, since autonomous scheduling reduces overhead of extra packets used for network management.

The work in this paper has been supported by THCS European project via RENEW project, and the Swedish Governmental Agency for Innovation Systems (VINNOVA) through GREENER and PROVIDENT projects, and by the Excellence in Production Research (XPRES) Framework. The computations was enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

One major problem with autonomous schedulers is the lack of flexibility due to their static allocation of resources. IoT networks should be able to adapt to the changes in the environment that can be caused by mobility or evolving traffic demand. Mobility-aware TSCH schedulers have been studied in the literature mostly under centralized schemes. These schedulers allow good flexibility that comes at the price of extra negotiation between the nodes and controller [3].

We believe that a hybrid scheduling (centralized + autonomous) scheme can be the best approach to support traffic-aware scheduling. In such a setting, the autonomous scheduler provides basic connectivity while significantly reducing the overhead of bringing up the network. On the other hand, a centralized scheduler can be used to ensure traffic awareness in the sense that nodes monitor their queue backlogs and request a centralized controller for more resources once their queues seem to be congested. The fact that our proposed hybrid scheduler builds on an autonomous scheduler introduces a new constraint for the scheduling problem that renders the existing works [4] on link scheduling irrelevant.

In this paper, we present the *Forte*[1] framework. *Forte* is a hybrid TSCH scheduler that aims at bringing the best of autonomous and centralized scheduling together. *Forte* extends the state-of-the-art Orchestra scheduler by integrating a centralized scheduler.

One major challenge in the design of *Forte* is optimizing the trade-off between queue stability and the cost of negotiations between nodes and the central controller. The cost of negotiation depends on the nodes' link quality and their hop distance to the controller. *Forte* resorts to Lyapunov optimization as it offers a mathematical framework that can be used to formulate queue stability in parallel with other costs.

In summary, the work reflected in this paper embeds the following contributions:

- We propose *Forte*, a hybrid TSCH scheduler that takes advantage of the lightweight operation of autonomous schedulers and the flexibility of centralized schedulers

[1]The term *Forte* in orchestral music refers to a piece of a symphony that is intended to be played louder.

- We model the problem as a Lyapunov optimization problem to achieve a trade-off between the cost of negotiation and queue stability
- We implement *Forte* using Contiki-NG and make it available as open-source[2].
- We evaluate the performance of *Forte* against the selected benchmark scheduler - *Orchestra* - using our implementation using the COOJA simulator, for several representative scenarios/setting, showing its advantages under mobility and dynamic traffic arrival.

The rest of the paper is organized as follows: Section 2 reviews some background regarding the 6TiSCH architecture and the underlying protocols. Section 3 addresses the previous works in the literature including TSCH schedulers that support mobile nodes. In Section 4, we provide the details of the proposed *Forte* framework, which is then evaluated in Section 5. Finally, Section 6 concludes the paper.

II. Background

IEEE 802.15.4e [1] introduced three modes of MAC operation: (i) *Time-Slotted Channel Hopping* (TSCH), (ii) *Low Latency Deterministic Network* (LLDN), and (iii) *Deterministic Synchronous Multichannel Extension* (DSME). TSCH utilizes time synchronization and frequency hopping to provide a deterministic MAC layer in which nodes communicate on a pre-scheduled timeslot and channel.

A. 6TiSCH protocol stack

Additionally, in 2021, the *Internet Engineering Task Force* (IETF) finalized the standardization of the 6TiSCH framework [5], a full IoT protocol stack that integrates IPv6 over *Low-Power Wireless Personal Area Networks* (6LoWPAN) and the *Routing Protocol for Low-Power and Lossy Networks* (RPL) on top of *Time-Slotted Channel Hopping* (TSCH). This framework has played a significant role in fostering a more deterministic behavior in Industrial IoT networks.

RPL is the *de facto* routing protocol for low-power networks and provides scalability to support hundreds of nodes with constrained devices. Nodes build a distributed data structure by tracking only their parents, forming a Destination Oriented Directed Acyclic Graph (DODAG).

RPL operates in two modes:

- *Storing mode*: requires every node to keep downward routes (children) in a table, resulting in a larger memory footprint.
- *Non-storing mode*: downward routing information is maintained by the root node only; downward packets use source-routing.

The root node initiates the routing protocol by broadcasting a DODAG Information Object (DIO), which is subsequently relayed to the farther nodes by receiving nodes. Each RPL node chooses its parent based on the parent's rank (a numeric representation of node's position concerning the root node) as defined by the DIO and the node's link quality, assessed using the *Expected Transmission Count* (ETX) metric.

[2]https://github.com/iliar-rabet/TSCH-mobility

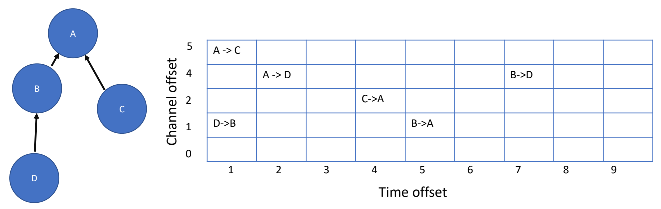


Fig. 1. An example network is illustrated running based on the RPL DODAG and corresponding TSCH schedule. The schedule shows that different pairs of nodes can utilize the same time offset on different channels for communication (visible on time offset 1). The guaranteed schedule avoids collisions thus providing a deterministic and flexible medium access.

B. TSCH schedule

The slotframe, a two-dimensional table based on which TSCH transmissions are scheduled, serves as a blueprint for wireless communication. Each pair of nodes is assigned a cell in the slotframe (with a time offset and channel offset) for transmitting the data/control packets. Figure 1 demonstrates an example network and its schedule. Each node can be programmed to switch its radio into transmission mode, receiver mode, or idle/sleep. Cells can also be scheduled for unicast or broadcast packets.

For a successful transmission, the receiver must be listening on the same channel at the right time. Acknowledgements are also transmitted during the same timeslot, while packet retransmissions have to wait for the next allocated cell.

Joining a TSCH network relies on receiving *Enhanced Beacons* (EB) for synchronisation and obtaining the necessary information, such as the *Frequency Hopping Sequence* (FHS), *Absolute Slot Number* (ASN), and slotframe length. ASN is a global counter that determines the current timeslot. The coordinator and the recently joined nodes advertise the network periodically by broadcasting EBs.

The nodes repeat the schedule over time and the channel hopping mechanism translates the channel on the schedule to the physical channel using Equation 1.

$$frequency = FHS[(channel + ASN) \mod |FHS|] \quad (1)$$

Existing TSCH schedulers are generally classified as (i) centralized, (ii) distributed and (iii) autonomous. Centralized and distributed schedulers, use a protocol called (6TiSCH) Operation Sublayer (6top) Protocol (6P) for negotiating over TSCH. Autonomous schedulers avoid these negotiations and decide on the cells based on the information obtained solely from the EB and RPL packets.

C. TSCH autonomous schedulers - the Orchestra benchmark

Recently, autonomous schedulers such as Orchestra [2] gained a lot of attention from the academic community for their low overhead and reliable operation. The idea behind autonomous schedulers is that no negotiation is performed, neither in a distributed nor in a centralized manner. Nodes use the locally existing information and a hash function to decide on the schedule and minimize collisions. Orchestra achieves

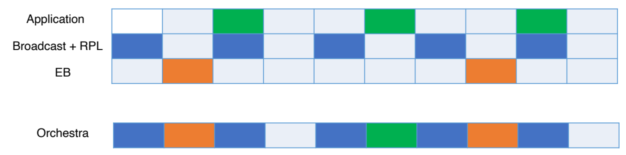


Fig. 2. An Example of Orchestra scheduling 3 repeating slotframes, with EB having the highest priority followed by broadcast (RPL) and unicast (application) cells.

this by utilizing the routing information for scheduling timeslots. Different variants of Orchestra have been introduced in recent years as surveyed by Elsts et. al. [6].

Orchestra defines two types of operation modes: receiver-based, and sender-based. In receiver-based Orchestra, time slot is determined based on the hash of the receiver's MAC address. These slots are more efficient in terms of energy consumption but transmitters compete for the time slot. In the sender-based mode, all the receivers switch to receiver mode at the time indicated by the sender's address incurring extra energy consumption.

By default, each node in the network is assigned three slotframes, in a predetermined order of priority:

- For sending EB from each node to its children, each node is assigned *one* slot in a slotframe with an arbitrary length.
- For broadcast packets (usually used for RPL's operation), each node is assigned *one* slot in a slotframe with an arbitrary length.
- For the application data both toward the node's parent and children, *multiple* dedicated slots are assigned.

To handle the overlaps, the length of these 3 slotframes is selected to be mutually prime. The length of the slotframes determines the traffic capacity, latency, and energy consumption of the nodes as shorter slotframes have their slots repeat more often. Figure 2 demonstrates an example Orchestra schedule and its default slotframes.

III. Related Work

The adoption of TSCH has spurred a variety of research projects and extensions to support better connectivity for mobile nodes. This section provides an overview of the challenges and state-of-the-art in supporting mobility and hybrid scheduling in a TSCH network. First, we review the studies that analyzed the impact of mobility on the performance of TSCH networks. Next, we discuss the existing TSCH schedulers that accommodate mobility, which include centralized, distributed, and autonomous schedulers. Finally, we explore the application of hybrid schedulers in adapting TSCH networks to dynamic environments.

A. Impact of Mobility on TSCH Networks

The impact of mobility on TSCH networks has been studied using simulations and analytical modeling. Nidawi et. al. [7] compared TSCH to LLDN under interference and mobility of the nodes using simulations and showed that TSCH can prolong joining of nodes to the network due to limited time allocated for sending EBs and RPL broadcast packets. A major theme in the literature is focusing on faster joining time for the mobile nodes. Another important aspect is the fluctuations in the data traffic pattern after the joining the network. Both of these challenges should be primarily handled by the scheduler.

Even without mobile nodes, bootstrapping a network running RPL and TSCH can be immensely long [8]. RPL's DIO packets are scheduled by the *trickle* algorithm, which adapts the rate of sending DIO packets to the dynamics of the radio environment. This also means that the control traffic will be considerably large when bootstrapping the network and then scale down to a few packets per hour, at run-time. In autonomous schedulers, the collisions between control traffic and data packets is limited and can be modeled [9] but these collisions can impact the joining time of the mobile nodes. Hence, different schedulers perform differently under presence of mobile nodes [10]. Comparing 3 well-known TSCH schedulers namely Minimal Scheduling Function (MSF), Orchestra and Alice showed that at least 20% of the packets are dropped for all of the nodes.

B. TSCH schedulers with mobility support

Hermeto et al.'s survey [11] classifies schedulers with mobility support into two categories. The first category applies to networks where a group of static nodes serves as the infrastructure, and only leaf nodes can be mobile. The second category allows all nodes to be mobile, but few studies have been conducted in this category due to the challenges it poses [12].

Many mobility solutions assume a static infrastructure to support wearable nodes. INSTANT [13] is one such approach, implementing a reliable anycast mechanism that distributes acknowledgments over time to avoid collisions. The authors demonstrate that three acknowledgments can be sent within a standard 10 ms TSCH timeslot. However, neighbor discovery in INSTANT requires periodic beacon transmission, which significantly increases the number of transmitted control packets. Despite this, INSTANT outperforms Orchestra in terms of delay, fairness, and energy consumption.

Tavallaie et. al. [14] proposed the Distributed Traffic-aware Scheduling Function (DT-SF), which integrates mobility and queue backlog into slot scheduling to address traffic imbalance in wireless networks. They model the problem as a Mixed-Integer Convex Programming (MICP) problem and minimize a defined utility function. The relaxed version of the problem can be solved by using the method of Lagrange multipliers, and the Branch-and-Bound algorithm can be used to generalize the solution for the version with integrality constraints.

FTS-SDN [3], MMF-SDN [15], and SDMob [16] are based on a Software Defined Network (SDN) controller which is integrated with the low-power network. FTS-SDN reserves one cell from each mobile node to each static node, clustering nodes to reduce excessive reservations. MMF-SDN reduces reservations by scheduling mobile nodes as multicast traffic sources, merging cells with the same traffic source to shorten the slotframe. SDMob uses the centralized controller to track the mobile nodes in real-time and pro-actively fix the routing links.

Haxhibeqiri et. al [17] propose allowing mobile nodes to roam between infrastructure nodes with low-rate upstream traffic, agreeing upon a dedicated cell with one of the gateways, and having all other nodes reserve the same cell for the mobile node. Pettorali et. al. [18] propose a scheduling function, Shared Downlink-Dedicated Uplink (SD-DU), assigning shared cells for downlink data transmission and dedicated cells for upward links. The scheduling function considers a tunable parameter to determine how many mobile nodes share a cell for upward communication, with the worst-case scenario being all mobile nodes in range of the same border router.

C. *Adaptive hybrid scheduling*

As mentioned by recent surveys [19], adaptive TSCH schedulers are gaining more attention and hybrid schedulers are one of the best options to improve the performance of autonomous schedulers. Following this trend, On-Demand TSCH (OST) scheduler [20] introduced a hybrid mechanism that autonomous cells are helped by occasional distributed negotiations. OST uses two modes of provisioning, namely: (i) periodic provisioning and (ii) on-demand provisioning where additional slots are allocated to support traffic bursts.

Another hybrid scheduler [21] that integrates a centralized controller that is based on CoAP. HPS [22] and A3 [23] are recent works that integrate autonomous scheduling with a distributed negotiation. The design choice to merge autonomous schedulers with distributed negotiation requires the in-network nodes to maintain a high level of information regarding the condition of the network.

Despite the recent developments, hybrid schedulers have not been studied in the context of mobility and most of the existing mobility solutions rely on centralized TSCH schedulers. In our view hybrid mobility support can be very beneficial to keep the queue stability with minimum negotiation overhead and duty-cycle increase under a dynamic traffic pattern. In this paper we aim at filling this gap by designing a queue-aware hybrid TSCH scheduler.

IV. Method

This section addresses the design of *Forte* and its components. First, we present the details of the interactions between the centralized controller and the autonomous operation of the network. Next, the queue model and the Lyapunov optimization are presented. We extend Contiki-NG's implementation of TSCH [24] with *Forte*.

A. *Hybrid operation*

The design of *Forte* aims at achieving flexible scheduling while minimizing negotiation costs and optimizing duty-cycle. The hybrid approach meets these design goals because negotiations between in-network nodes and the controller utilize the basic connectivity provided by Orchestra and non-storing RPL. We chose non-storing mode of RPL and receiver-based Orchestra as they offer the lowest level of overhead compared to other alternatives (the storing mode RPL and sender-based scheduling).

Figure 3 illustrates an example of a *Forte* cell being allocated. All nodes maintain queues for transmission to each of their neighbors. *Forte* nodes monitor these queues, and when certain conditions (outlined in the queue model subsection) are met, they request the controller to allocate or deallocate a number of cells.

The controller is aware of the network's real-time topology through RPL's DODAG and can determine the initial schedule provided by Orchestra. The initial schedule provided by Orchestra (represented by orange cells in the figure) offers each node one RX cell. Upon receiving a request for more cells, the controller allocates a number of unused cell (represented by green shading in the figure) that have not been allocated to other nodes, either through Orchestra or by the controller. The controller then notifies the requesting node and its parent of its decision. Negotiation with the centralized controller is based on CoAP confirmable messages to ensure successful cell installation.

B. *Queue model*

Now we present how *Forte* nodes decide to request for allocation or deallocation of cells. We study the system in discrete time where each increment consists of one slotframe. Denote $f_{(x(t)} \leq 0$ as the amount of data traffic successfully transmitted towards a neighbor (x) during the slotframe t, $Q_x(t)$ as the queue associated with outgoing packets towards neighbor x, $MaxQ_x$ the maximum queue size, $A_x(t)$ as arrivals including packets generated by the node itself or packets to be relayed. The queue dynamics are defined as follows:

$$0 \leq Q_x(t) \leq MaxQ_x \tag{2}$$

$$Q_x(t+1) = |Q_x(t) - f_x(t)|_+ + A_x(t) \tag{3}$$

In the above equations the plus operator is defined as $|x|_+ = \max(0, x)$.

Forte nodes adapt the future capacity of the link by selecting the action $\alpha_x(t)$. For instance, nodes decide to ask the centralised controller for one more ($\alpha_x(t) = 1$), or fewer cells ($\alpha_x(t) = -1$). The other option is to maintain the same schedule by not sending any request ($\alpha_x(t) = 0$).

$$f_x(t) = f_x(t-1) + \alpha_x(t) \tag{4}$$

Nodes will decide on a vector $\alpha(t) = (\alpha_{x1}, ...\alpha_{xn})$ for all the neighbors. For simplicity, we assume the time required for negotiation with the server is negligible.

Next, we model the cost associated with the decision vector. The cost of negotiation with the centralized controller depends on the distance of the nodes and the quality of the links. For nodes that are placed far from the root node or have unreliable links, it is more costly to negotiate. We model this cost by computing the Rank value of node x based on the rank of

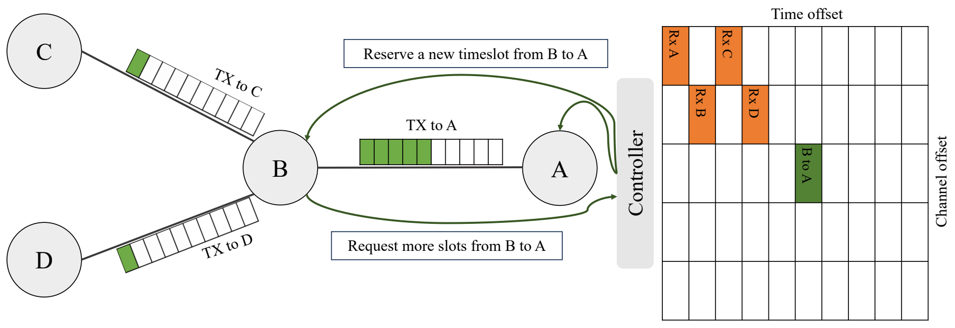


Fig. 3. An example of *Forte*'s operation: Node B detects that its transmission queue toward Node A is becoming congested and requests additional communication resources from the centralized controller. The controller allocates a new cell that does not interfere with previously assigned cells, whether allocated by Orchestra or through centralized control. In the example, cells assigned through autonomous operation are shaded in orange, while those assigned by the centralized controller are shaded in green.

its neighbors ($y \in \mathcal{N}_x(t)$) and the available information from RPL.

$$Rank_x(t) = \begin{cases} \min\limits_{y \in \mathcal{N}_x(t)} (p_{x,y}(t) + Rank_y(t)) & x \notin \mathcal{R} \\ \text{RootRank}_x & x \in \mathcal{R} \end{cases}, \tag{5}$$

We apply quadratic Lyapunov function to the queue backlog, so that larger queue backlogs will have higher impact in the decision.

$$L\left(\Theta\left(t\right)\right) = \frac{1}{2}\sum_{i=1}^{n} Q_i^2(t) \tag{6}$$

Next, we calculate the queue drift. Intuitively, by minimizing drift, our algorithm aims at keeping all the queue backlogs small. For proof on how minimizing the Lyapunov drift results in queue stability please refer to the literature [25].

$$\Delta(\Theta(t)) \stackrel{\triangle}{=} L\left(\Theta\left(t+1\right)\right) - L\left(\Theta\left(t\right)\right) \tag{7}$$

To achieve a trade-off between queue stability and the negotiation overhead, we use the concept of drift-plus-penalty. In other words, nodes try to minimize a weighted sum of the Lyapunov drift and penalty. The penalty includes (i) a one time negotiation cost which is associated with the rank of node and (ii) the energy consumption associated with the decision (the third term in the optimization below). We assign a different weight (V_1 and V_2) for each of these costs that can be tuned based on the requirements of the network. L_{UC} represents the length of unicast slotframe size.

$$\min_{\alpha(t)} \Delta(\Theta(t)) + V_1 \cdot Rank_x(t) + V_2 \cdot (DutyCycle + \frac{\alpha(t)}{L_{UC}})$$

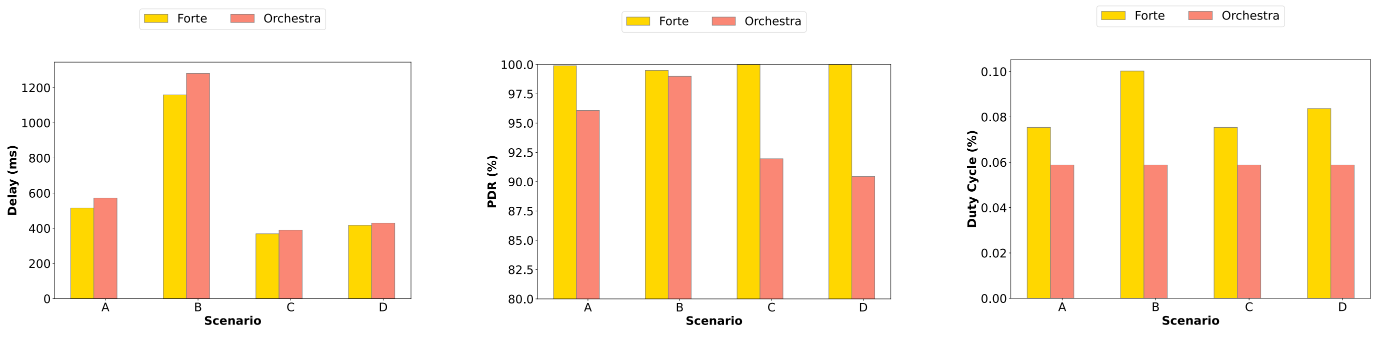
By optimizing the aforementioned objective function, we keep the queues stable (reduce packet loss and improve delay) while also reducing the overhead of negotiation and duty cycle. The in-network nodes use a 'brute-force' approach to solve this optimization problem and find the best action ($\alpha(t)$) among all possible candidates, assuming the queues are independent.

V. Evaluation

In order to demonstrate the efficiency of *Forte*, we have conducted simulations using Contiki-NG under different scenarios. We analyze Packet Delivery Ratio (PDR), average delay of selected traffic, and radio duty-cycle of all nodes. We compare *Forte* with receiver-based *Orchestra* under various conditions, including varying traffic and node mobility. Both *Orchestra* and *Forte* are configured with identical slotframe sizes (17 for unicast, 31 for broadcast, and 397 for EB). Further evaluations, such as scalability analysis, remain a future work due to size constraints.

We organized the evaluation considering the following 4 scenarios:

- Scenario A: This scenario includes 8 stationary nodes transmitting randomly at a rate of 30 pkt/min of upward traffic. We set the rate to slightly surpass the rate of packets that nodes that are closer to the root node. Thus they need additional cells to aggregate the traffic from other nodes.
- Scenario B: In this scenario, a set of 8 stationary nodes transmit an increasing rate of traffic upward in the mesh. The rate starts at 15 pkt/min and every 50 packets the rate is doubled. This setting puts increasing pressure on the nodes that are close to the root node. This scenario is designed to assess how fast can the controller allocate necessary timeslots.
- Scenario C: This scenario includes a single mobile node that roams around while sending a traffic of 30 pkt/min. 8 stationary nodes send a lightweight traffic of 2 pkt/min. In this setting, assigning enough cells to the high traffic demands of the mobile node is crucial for the network to function properly. When the mobile node moves from the proximity of its parent and a parent switch happens, the new parent needs more TX time slots to keep its queues stable.
- Scenario D: For the last scenario, we increase the number of mobile nodes (2 nodes, each sending traffic at 15 pkt/min) to analyze the impact of the mobile nodes on each other. The traffic path that we are interested in is



(a) (b) (c)

Fig. 4. *Forte* outperforms *Orchestra* by demonstrating lower delay (a) and higher PDR (b) for a selected traffic flow. This is achieved by a rather small increase in the average duty-cycle (c) of all the nodes.

from the mobile nodes to the root node. This experiment focuses on the required communication resources for properly sending the data packets. However, another important aspect of mobility solutions is the handover delay which depends on the communication resources that are reserved for control packets (RPL and EB). Handling the impact of handovers is out of scope of this paper but presents an interesting future work.

We measured the *Packet Delivery Ratio* (PDR) and mean end-to-end delay of the target traffic toward the root node and the duty-cycle of the nodes over time. Figure 4 illustrates the measurements from the simulations for the different scenarios. In all scenarios *Forte* outperforms the benchmark in terms of delay and PDR of the selected traffic. In scenarios that includes mobility (C, and D), *Forte*'s advantages are more tangible in terms of PDR while in scenario A and B the high load of traffic that is incurred on the network causes delay rather than packet loss and *Forte* is beneficial in reducing delays.

In scenario A, by only increasing the average duty-cycle from 5.8% to 7.8%, *Forte* achieves a PDR of 100% and improves average delay by 100 ms. The only way that *Orchestra* could have achieved the same performance as *Forte* would be by increasing the duty-cycle of all the nodes together at least to 11.6% (double). In Scenario B, nodes increase their traffic rate and at the final stage of the scenario the traffic is as high as 8 nodes at the rate of 120 pkt/min. *Forte* allocates extra cells only when needed which helps to reduce the average duty-cycle.

Regarding the two mobile nodes (D), a key challenge for receiver-based *Orchestra* is that transmitters share the same cell for upward traffic. An alternative is switching to sender-based mode that requires changing the RPL operation mode to storing mode, which would introduce additional overhead. *Forte* circumvents this by assigning dedicated cells for mobile nodes' traffic on demand. *Forte* achieves 100% packet delivery with only a 1% increase in duty-cycle, indicating its effectiveness in supporting multiple mobile nodes.

VI. Conclusion

Autonomous schedulers provide lightweight but reliable operation for TSCH networks. To support modern applications schedulers need to be able to adapt to the traffic demand. This paper shows how an autonomous scheduling techniques can be improved by integrating a centralized controller. The motivation behind our scheduler, Forte, is to minimize the overhead of centralized negotiation and at the same time benefit from the flexibility that it brings. Our results show that with a very low increase in duty-cycle of specific nodes, Forte succeeds in improving delay and reliability of the network under mobility and evolving traffic.

When supporting connectivity of mobile nodes, Forte prioritizes adapting the schedule to maintain queue stability over other aspects of mobility, such as handoff procedures. Forte primarily adds unicast slots, possibly overlooking broadcast packets that are typically used for routing and node joining.

For future work, we plan to leverage hybrid scheduler to enhance handoff processes and optimize rate of control packets. Also, further evaluations such as scalability analysis, and physical testbed remain a future work due to size constraints.

References

[1] I. of Electrical and E. Engineers, "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 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, 2012.

[2] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 2015, pp. 337–350.

[3] L. L. Bello, A. Lombardo, S. Milardo, G. Patti, and M. Reno, "Experimental assessments and analysis of an sdn framework to integrate mobility management in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5586–5595, 2020.

[4] N. Papadis and L. Tassiulas, "Payment channel networks: Single-hop scheduling for throughput maximization," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 900–909.

[5] P. Thubert, "An architecture for IPv6 over the time-slotted channel hopping mode of IEEE 802.15. 4 (6TiSCH)," *tech. rep., RFC 9030*, 2021.

[6] A. Elsts, S. Kim, H.-S. Kim, and C. Kim, "An empirical survey of autonomous scheduling methods for TSCH," *IEEE Access*, vol. 8, pp. 67 147–67 165, 2020.

[7] Y. Al-Nidawi, H. Yahya, and A. H. Kemp, "Impact of mobility on the iot mac infrastructure: Ieee 802.15. 4e tsch and lldn platform," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 478–483.

[8] C. Vallati, S. Brienza, G. Anastasi, and S. K. Das, "Improving network formation in 6TiSCH networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 98–110, 2018.

[9] I. Rabet, H. Fotouhi, M. Alves, J. P. Champati, J. Gross, M. Vahabi, and M. Björkman, "A Stochastic Network Calculus Model for TSCH Schedulers," in *IEEE Symposium on Computers and Communications (ISCC)*, Paris, June 2024.

[10] C. Orfanidis, A. Elsts, P. Pop, and X. Fafoutis, "TSCH Evaluation under Heterogeneous Mobile Scenarios," *IoT*, vol. 2, no. 4, pp. 656–668, 2021.

[11] R. T. Hermeto, A. Gallais, and F. Theoleyre, "Scheduling for IEEE802. 15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey," *Computer Communications*, vol. 114, pp. 84–105, 2017.

[12] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of manet and wsn in iot urban scenarios," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3558–3567, 2013.

[13] A. Elsts, J. Pope, X. Fafoutis, R. J. Piechocki, and G. Oikonomou, "Instant: A TSCH Schedule for Data Collection from Mobile Nodes." in *EWSN*, 2019, pp. 35–46.

[14] O. Tavallaie, J. Taheri, and A. Y. Zomaya, "Design and optimization of traffic-aware TSCH scheduling for mobile 6TiSCH networks," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 234–246.

[15] F. Orozco-Santos, V. Sempere-Payá, J. Silvestre-Blanes, and T. Albero-Albero, "Multicast scheduling in SDN-wise to support mobile nodes in industrial wireless sensor networks," *IEEE Access*, vol. 9, pp. 141 651–141 666, 2021.

[16] I. Rabet, S. P. Selvaraju, H. Fotouhi, M. Alves, M. Vahabi, A. Balador, and M. Björkman, "SDMob: SDN-Based Mobility Management for IoT Networks," *Journal of Sensor and Actuator Networks*, vol. 11, no. 1, p. 8, 2022.

[17] J. Haxhibeqiri, A. Karaağaç, I. Moerman, and J. Hoebeke, "Seamless roaming and guaranteed communication using a synchronized single-hop multi-gateway 802.15. 4e TSCH network," *Ad hoc networks*, vol. 86, pp. 1–14, 2019.

[18] M. Pettorali, F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "Mobility Management in Industrial IoT Environments," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2022, pp. 271–280.

[19] A. Tabouche, B. Djamaa, and M. R. Senouci, "Traffic-aware reliable scheduling in TSCH networks for industry 4.0: A systematic mapping review," *IEEE Communications Surveys & Tutorials*, 2023.

[20] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "OST: On-demand TSCH scheduling with traffic-awareness," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 69–78.

[21] A. Karaagac, I. Moerman, and J. Hoebeke, "Hybrid schedule management in 6TiSCH networks: The coexistence of determinism and flexibility," *IEEE Access*, vol. 6, pp. 33 941–33 952, 2018.

[22] A. Tabouche, B. Djamaa, and M. R. Senouci, "Hps: A hybrid proactive scheduler with adaptive channel selection for industrial 6tisch networks," *Ad Hoc Networks*, p. 103527, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870524001380

[23] S. Kim, H.-S. Kim, and C.-k. Kim, "A3: Adaptive autonomous allocation of TSCH slots," in *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021)*, 2021, pp. 299–314.

[24] S. Duquennoy, A. Elsts, B. Al Nahas, and G. Oikonomo, "TSCH and 6TiSCH for Contiki: Challenges, design and evaluation," in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2017, pp. 11–18.

[25] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Springer Nature, 2022.