Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Layered autonomous TSCH scheduler for minimal band occupancy with bounded latency

Andreas Ramstad Urke*[1,2] | Øivind Kure[3] | Knut Øvsthus[1]

[1]Department of Computer science, Electrical engineering and Mathematical sciences, Western Norway University of Applied Sciences, Bergen, Norway
[2]Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, Trondheim, Norway
[3]Faculty of Mathematics and Natural Science, University of Oslo, Oslo, Norway

Correspondence
Andreas Ramstad Urke, Norwegian University of Science and Technology, Trondheim, Norway. Email: andrerur@stud.ntnu.no

Summary

Robustness and bounded latency are among the key requirements in wireless industrial networks. A typical traffic patterns is convergecast, where sensors sends data towards a sink, resulting in a funneling effect with increased traffic intensity. This letter proposes the Layered autonomous Time Slotted Channel Hopping (TSCH) scheduler for convergecast traffic, which addresses the funneling effect while reducing the band occupancy. It divides the slotframe into layers where all nodes have one timeslot reserved for forwarding its own traffic. Layers are organized to allow spatially reuse such that traffic belonging to a node may be forwarded at multiple hops simultaneously. This is achieved autonomously by exploiting a node's knowledge of the routing topology. We evaluate the Layered scheduler through theoretical analysis and simulations in Cooja. Results show that Layered attains bounded latency even when nodes are utilizing all their available resources. This is achieved with significantly reduced band occupancy compared to Escalator, an autonomous scheduler for convergecast traffic. The performance is traded-off by increased maximum latency.

KEYWORDS:
IIoT, CPS, DetNet, TSN, Time-Slotted Channel Hopping (TSCH), Autonomous scheduling, 6TiSCH

1 | INTRODUCTION

The visions of Cyber-Physical Systems and Industrial Internet of Things have resulted in significant research towards meeting industrial requirements in wireless networks. Key requirements are high reliability and deterministic bounded latency[1].

The Media Access Control (MAC) layer has significant impact on meeting these requirements, and a trend can be seen towards utilizing Time Slotted Channel Hopping (TSCH). This MAC combines channel-hopping, to increase resilience against wireless fading events, with a possibility to allocate timeslots in a contention-free manner. TSCH is used in industrial standards such as WirelessHART and the IEEE 802.15.4-2015. Furthermore, it is selected as the MAC in IETFs ongoing work on 6TiSCH - an IPv6-enabled stack for industrial wireless low-power networks[2].

Nodes in a TSCH network operate according to a schedule implemented as one or more *slotframes*. These repeat over time and dictate if a node is allowed to transmit or receive. Figure 1 shows an example network with an accompanying schedule. Time is divided into timeslots on the horizontal axis, while the available channels are shown in the vertical. A *cell* is defined by a pair of timeslot- and channel-offset coordinates, and it allows for transmitting or receiving one frame with an optional acknowledgment.

Key in TSCH is the establishment and maintenance of the slotframe content, i.e. how cells are allocated among nodes in the network. This is achieved by a *scheduler*, which may operate in a centralized, collaborative, or autonomous fashion. With an autonomous scheduler, nodes create their schedules independently without any information exchanged between schedulers

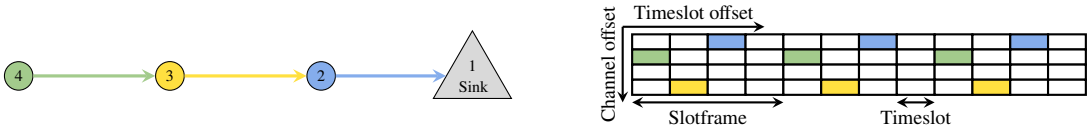


FIGURE 1 Simple wireless network topology with example TSCH schedule

on the nodes. This simplifies the configuration, avoids signaling overhead, and increases the fault tolerance since no signaling is needed to utilize new links. However, adaption to heterogeneous traffic is typically challenging. In the state-of-the-art autonomous scheduler Orchestra[3], nodes allocate cells based on their own and neighbors ID. The timeslot offset of cells are calculated autonomously by a hash algorithm with the input based on either the receiver or sender ID. In the sender-based mode, a contention-free schedule is possible if the hash output is unique.

1.1 | Problem statement and contribution

Common industrial applications such as monitoring have a convergecast traffic-pattern where sensors periodically send data to a sink, normally relayed through other sensors. This results in increased traffic intensity closer to the destination, i.e. a *funneling effect*. Industrial applications typically require reliability and bounded latency, thus the TSCH schedule needs to take this effect into account in order to avoid packet loss or queuing which increases the latency. Most autonomous schedulers rely on contention-based approaches to address such traffic heterogeneity, resulting in non-deterministic performance[4].

We designed a scheduling algorithm emphasizing spatial reuse while addressing the funneling effect. Radio channels may be expected to be a scarce resource. In an industrial environment there will be multiple isolated networks, allocated in separate channels to avoid fate sharing if overloaded and eased planning and configuration. Additionally, in a challenging RF environment it is to be expected that channels may be unusable or blacklisted.

The contribution of this letter is as follows:

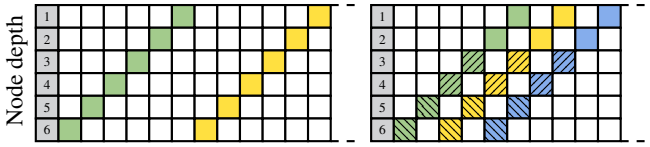
- We propose the *Layered autonomous scheduler* which focus on limiting band occupancy through spatial reuse while maintaining a bounded latency. To our knowledge it is the first autonomous scheduler to do so.
- We confirm the feasibility of the Layered scheduler through theoretical analysis and simulations, and show how the band occupancy is lower compared to Escalator, a state-of-the-art scheduler designed to mitigate the funneling effect.

2 | RELATED WORK

In order to mitigate the funneling effect in a deterministic fashion we need to ensure there are sufficient resources for every node's traffic from source to destination. To describe the different autonomous approaches, we first introduce an intuitive solution: A cell can be allocated at every hop dedicated to a particular node's traffic, as illustrated in Figure 2a. This yields a contention-free schedule with guaranteed delivery within one slotframe (assuming perfect links). The immediate issue with this approach is the slotframe length which equals the maximum allowed hop count times the number of supported nodes. Both the Layered scheduler and related work utilizes the essence of this intuitive solution yet differ in the way they reduce the slotframe length.

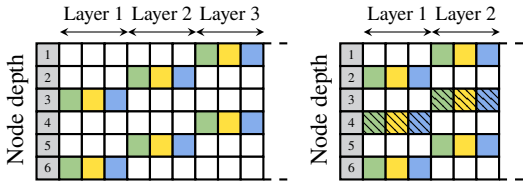
Escalator[5] is the only autonomous scheduler targeting convergecast while guaranteeing sufficient and contention-free resources for the traffic[4]. It solves the previously described slotframe length problem by placing the strings as close as possible to each other, as illustrated in Figure 2b. This allows *traffic from multiple different nodes to be forwarded in the same timeslot*. Contention is avoided by employing a different channel for every two hops, as indicated by the patterned cells. However, this *greatly increases* the band occupancy: For every two hops supported, one additional channel is occupied. The position of each node's string in the slotframe can be calculated by the node ID, i.e. Escalator assumes a collision free hash algorithm to translate IDs into an offset. Thus to implement the schedule, a node is only required to know the ID of any nodes in its sub-tree (to learn which strings he is part of), and his own depth (to learn where in the string he is).

Several collaborative schedulers target bounded latency. These differ from autonomous schedulers as they rely on neighbors negotiating which cells to utilize. A notable example can be found in[6] which similar to our proposal utilizes the node depth and spatial reuse to optimize for delivery within one slotframe.



(a) Intuitive solution **(b)** Escalator

FIGURE 2 Solutions for funneling effect. Colors denote from which node traffic originated. Pattern indicate channel. Depth = hops from root.



(a) Single-channel **(b)** Two channels

FIGURE 3 Transmission-cells in Layered schedule. Colors denote from which node traffic originated. Pattern indicate channel.

3 | LAYERED SCHEDULER

The Layered scheduler solves the aforementioned slotframe length problem by dividing the string of cells into pieces and have the different pieces overlap. This results in *traffic belonging to one node being forwarded simultaneously at multiple hops*, as illustrated in Figure 3a. Since the packets originated from the same node and belong to the same flow, spatial reuse may be employed, reducing the band occupancy compared to Escalator. This is implemented by dividing the slotframe into *layers*. Each layer corresponds to a depth from the root and contains enough cells for all nodes' traffic to be transmitted to the next layer/hop. Thus, as a layer is passed, all packets on the corresponding depth are forwarded to the next.

As the distance between two nodes in terms of hops increases, it is assumed a point is reached where they no longer interfere. Two nodes which are beyond this distance, may use the same cell, i.e. layer. The actual boundary depends on the routing protocol objective function and local RF conditions. Layered therefore allows its utilized boundary to be configured at deployment. An example schedule utilizing one channel and three layers is shown in Figure 3a. This yields a spatial re-use at every third hop. If this distance is insufficient, additional layers may be added, thus allowing a trade-off between resilience against interference and slotframe length. However, the schedule in Figure 3a is not optimal in terms of latency as transmissions are not done at every other hop simultaneously. The optimum may be reached by employing an additional channel, as depicted in Figure 3b. This allows a reduction to two layers, with spatial reuse occurring at every fourth hop, i.e. increased resilience as compared to the single-channel setup. If this distance is insufficient, additional channels may be employed. Consequently, we can trade band occupancy for increased resilience, as opposed to latency in the single-channel configuration.

The resulting slotframe length is the product of the number of nodes N supported and the number of layers L:

$$SF_{len} = N * L + CS$$

Where CS is the number of common slots (described later). Note that L = 2 for multi-channel. The length of each layer would equal the number of nodes supported in the network.

TABLE 1 Calculation of cell coordinates for node n_s

(a) Formulas

Layer

$layer(n) = L - ((depth_n - 1) \bmod L)$

Timeslot

$TS(n_i, n_j) = H(n_i) + ((layer(n_j) - 1) * N) + CS_{passed}$

Channel

$CH(n) = \left\lfloor \frac{depth_n - 1}{L} \right\rfloor \bmod CH$

(b) Cell calculations

	Timeslot	**Channel**
Unicast TX upward	$TS(n_o, n_s)$	$CH(n_s)$
Unicast RX upward	$TS(n_o, n_c)$	$CH(n_c)$
Broadcast TX upward	$TS(n_s, n_c)$	$CH(n_c)$
Broadcast RX upward	$TS(n_p, n_s)$	$CH(n_s)$

(c) Symbols

n_o, n_s	Node originating traffic, scheduling node
n_c, n_p	Child of n_s (towards n_o), parent of n_s
N, CH	Number of nodes, channels supported
L	Number of layers
$depth_n$	Num hops from root for node n
H	Hash function yielding unique timeslots
CS_{passed}	Common slots passed thus far in the SF

In the following we describe how the schedule is built. A formal description on how to identify the coordinates of each cell can be found in Table 1. Common broadcast slots are added in fixed positions before deployment.

3.1 | Unicast upward traffic

Similar to Escalator, the Layered scheduler assumes a node knows 1) Its current depth, and 2) The node ids in its sub-tree. The RPL routing protocol in the 6TiSCH stack fulfills these requirements when downward routing is enabled. Note that any objective function may be utilized as long as the depth is included in the RPL DIO packet. Additionally, the hash function which maps a node ID to timeslot (H in Table 1) must have a collision-free output. This may be solved by enforcing a maximum number of nodes supported before deployment, which we argue is reasonable in a managed industrial network. This same requirement is found in Escalator and sender-based collision-free Orchestra.

In RPL, every node transmits a Destination Advertisement Object (DAO) packet towards the root to establish downwards routing. The scheduler can exploit this to insert cells for upward unicast traffic: Upon receiving a DAO, it extracts the originating node's ID. This, along with its own depth, is sufficient to calculate the cell to be added, following the formulas in Table 1.

Using node 2 in Figure 1 for an example: Upon receiving a DAO originating from node 4, it would first add an RX cell such that it can receive traffic forwarded via node 3. The cell timeslot offset is calculated according to "Unicast RX upward" in Table 1 and require 1) The originating node ID n_o which is fetched from the DAO, and 2) The child's (node 3) layer $layer_{n_c}$ which is known from the RPL rank. Thus, node 4 would add a RX cell at offset 4 in the layer below its own. Similarly, it would add a TX cell at its own layer, to forward the traffic from node 4 to the sink, according to the "Unicast TX upward" equations. With this executed at each node along the DAO path, traffic from node 4 will have a dedicated cell at every hop all the way to the sink.

3.2 | Downward traffic, RPL traffic, and beacons

Layered utilize only one slotframe to accommodate all traffic. Thus, for RPL packets, common broadcast slots (CS) are inserted at fixed timeslots before deployment. For every CS, the remaining cells in the slotframe must be shifted to the right. An example can be seen in Figure 5b in Section 4.2. This makes the timeslot offset calculation only slightly more complex as seen by the adding of CS_{passed} in Table 1. However, it avoids doubling of the maximum latency due to slotframe collisions, as in Escalator.

A node knows the cell for its own traffic at the layer below himself is not in use. This is therefore utilized for transmitting TSCH beacons and downward traffic. The child node, knowing its parent ID and layer, adds a corresponding RX cell in his own layer. The coordinates for these cells can be derived using the definitions for downward cells in Table 1.

4 | SIMULATION AND RESULTS

We evaluate and verify the feasibility of the Layered scheduler using Contiki-NG[1] v4.4 simulator Cooja which has an implementation of both Orchestra and the 6TiSCH stack. The following performance indicators are used: 1) Application-layer end-to-end latency, 2) End-to-end Packet Delivery Ratio (PDR) for reliability, 3) For energy: Radio duty cycle, i.e. the ratio between enabled radio time and the total time, and lastly 4) Band occupancy, the amount of cells which a scheduler *potentially* may add to the schedule. Since we focus on industrial applications we are interested in bounds and not average performance. Thus, for each run we measure the overall PDR and the 99.9 percentile of latency. We next calculate the 95 % confidence interval of the 95 % percentile of these metrics across all runs, and present the conservative interval bound (lower for PDR, upper for latency).

TABLE 2 Theoretical comparison

	Escalator	Layered
Min. num. channels	$\frac{hops_{max}}{2}$	1
Slotframe length	$N * 2$	$N * L + CS$
Bounded latency	✓	✓(w/hop limit)
Max. latency without retransmissions, in timeslots	$(SF_{len} + hops_{max}) * 2$	$SF_{len} + (hops_{max} - 1) * N + \left\lceil \frac{hops_{max}}{L} \right\rceil * CS$
Latency per retrans.	SF_{len}	SF_{len}

TABLE 3 Simulation settings

Parameter	Setting
Mote	Cooja
Physical layer	802.15.4, 2.4 GHz, 250 kbps
Orchestra mode/hash	Sender based, collision-free
Layered num. layers/channels/CS	2/2/3
Layered num. nodes supported	49
TSCH pending bit & RPL probing	Disabled
RPL objective function	OF0 with hop count
Payload	UDP, 25 bytes

[1] http://contiki-ng.org

We simulate a converged monitoring application where sensor data is sent at a fixed interval to the sink. Each node randomly selects a starting point within the slotframe. To ensure representative results, we repeat the simulations 60 times with different seeds. The transmission interval is set according to nodes' capacity: Slotframe length is always set to 101 timeslots, thus when a node has one cell allocated for its traffic, a 100 % utilization yields an interval of 1010 ms, 50 % yields 2020 ms interval, and so on. We utilize the simple Unit Disk Graph Medium radio propagation model, with 100 % link quality. Evaluation in more realistic environments are deferred to future work, see discussion in Section 5. Table 3 shows the key simulation parameters utilized. The TSCH queue module was modified such that a cell would only be utilized for traffic from the intended node.

4.1 | Results

As a baseline we employ a simple chain topology with one sink and two nodes. Figure 4b shows the resulting latency and PDR as the interval increases according to the node's allocated resources. We added Orchestra to this scenario in order to showcase the funneling effect with an even allocation of resources among nodes. Note how latency increases and PDR decreases at 50 % capacity since the funneling effect overwhelms the node closest to the sink which also forwards traffic from its child.

Layered is able to retain both latency and PDR even when capacity is fully utilized by the nodes. Note that the latency stays below the theoretical maximum derived in Table 2. Reviewing the queue utilization further shows it does not grow significantly and is always less than 30 %. Layered mitigates the funneling effect by adding cells accommodating all traffic end-to-end. This can be seen in Figure 4a which shows excerpts from the Layered slotframe for this setup. Note how the node closest to the sink has an additional cell compared to Orchestra. This accommodates forwarding traffic from the child to the sink.

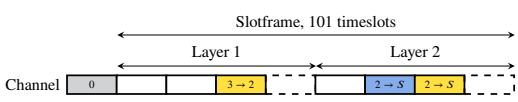
4.2 | Spatial reuse and scalability

Next we assess Layered's ability to retain performance in larger topologies where spatial reuse is employed. We employ a 5 x 5 grid as well as an 8-hop linear topology, as depicted in Figure 5c and 5a, respectively. The traffic pattern is the same as in the previous scenario, yet the queue size is raised from 8 to 16 and 64 such that the increased amount of cumulative traffic can be accommodated. Orchestra is omitted from the results as the trend from the previous scenario was continued yet amplified by the additional nodes, with latency and PDR greatly suffering already at 20 % utilization.

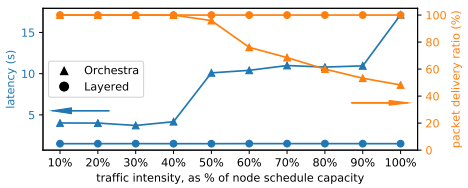
To illustrate the spatial reuse in Layered, an excerpt from the schedule for the 8-hop topology is shown in Figure 5b. It shows all cells related to the traffic from node 8 and 9. As intended, there is spatial reuse of cells in all layers and channels. The performance for this topology is so similar to the grid simulation, that we omit the results for brevity.

Latency and PDR for the grid topology can be seen in Figure 5d. As in the baseline scenario, Layered is still able to retain performance even when all capacity is utilized. The latency is never higher than 4.56 seconds - in line with the theoretical predictions in Table 2. Layered makes no assumption on the topology - since each layer has enough cells for every node there is no constraint on the distribution of nodes in the network. We therefore argue these results hold for any topology.

The energy consumption is expected to have a linear increase as the traffic intensity grows: The number of cells is the same for all scenarios, yet cell utilization increases. Figure 5e confirms these expectations, showing a consistent increase in duty cycle. Note that the time spent listening increases only by 20 % while the traffic increases ten-fold. This highlights a drawback of fixed allocations such as done by Layered and Escalator. Since the number of allocated cells are not adapted to traffic intensity, energy is wasted on idle listening when traffic is less than the capacity. Orchestra is added as baseline for a setup where nodes have one cell each, yet note that Orchestra has significant packet loss due to insufficient resources at traffic intensities above 10 %.

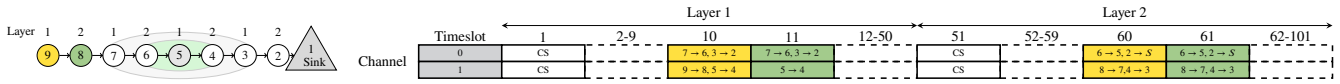


(a) Upwards unicast cells. Color denote from which node the traffic originated. Notation from Figure 1.



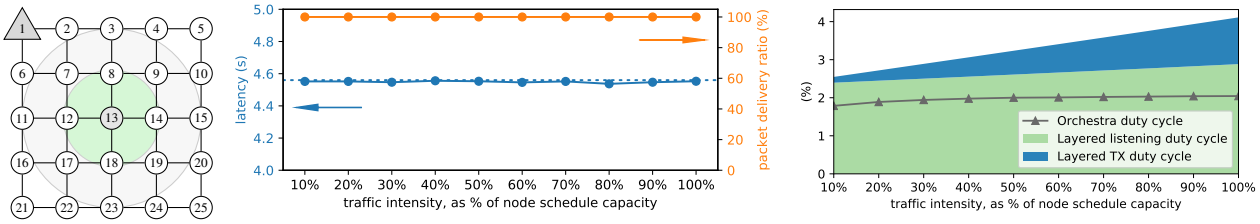
(b) Latency and PDR

FIGURE 4 Slotframe and results for two-hop topology



(a) 8-hop topology. TX- and interference-range in green and grey

(b) Layered schedule for traffic from node 8 and 9 in 8-hop topology. Color denote from which node the traffic originated.



(c) Grid topology

(d) Latency & PDR for grid. Dotted line is theoretical max. from Table 2

(e) Mean duty cycle for grid topology

FIGURE 5 Topology, slotframe and results for 8-hop- and grid-topology

5 | DISCUSSION

We have shown the Layered scheduler achieves bounded latency, similar to Escalator in[5]. The key difference is band occupancy, which for both schedulers can be derived from their design and thus does not require simulations. As highlighted in Table 2, Escalator requires one channel per two hops supported. Layered on the other hand can operate on a single channel, or on a limited set, such as the two channels used in our evaluation. This comes at the cost of increased maximum latency: Assuming no retransmissions, Escalator guarantees delivery in two slotframes. With Layered, the maximum latency depends on the number of layers a packet must traverse, which is limited by the maximum number of hops.

Wireless links are by nature unreliable and transmissions must be expected to fail. There is a wide range of mitigation techniques such as retransmissions, dropping, black-listing, path diversity, network coding, etc. The appropriate solutions is highly dependent on the application. We defer investigation of imperfect links to future work where we intend to combine Layered with multiple mitigation mechanisms in a test-bed evaluation. As shown in Table 2, Layered's long slotframes results in significantly increased latency if retransmissions are used - beyond what is typically acceptable for industrial applications.

Both Escalator and Layered fits convergecast applications which require bounded latency and sufficient resources for all nodes' traffic. However, since Layered limits the band occupancy through its spatial reuse, it will be beneficial in scenarios where frequencies are scarce, significant blacklisting is required, or in multi-network or -scheduler deployments. Lastly if a large number of nodes is required, Layered could accommodate it while retaining throughput by deploying multiple networks.

References

1. Watteyne T, Handziski V, Vilajosana X, et al. Industrial Wireless IP-Based Cyber-Physical Systems. *Proceedings of the IEEE* 2016; PP(99): 1-14. doi: 10.1109/JPROC.2015.2509186
2. Vilajosana X, Watteyne T, Chang T, Vučinić M, Duquennoy S, Thubert P. IETF 6TiSCH: A Tutorial. *IEEE Communications Surveys Tutorials* 2019: 1-1. doi: 10.1109/COMST.2019.2939407
3. Duquennoy S, Al Nahas B, Landsiedel O, Watteyne T. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In: SenSys '15. ACM. ACM; 2015; New York, NY, USA: 337–350
4. Elsts A, Kim S, Kim H, Kim C. An Empirical Survey of Autonomous Scheduling Methods for TSCH. *IEEE Access* 2020; 8: 67147-67165.
5. Oh S, Hwang D, Kim KH, Kim K. Escalator: An Autonomous Scheduling Scheme for Convergecast in TSCH. *Sensors* 2018; 18(4). doi: 10.3390/s18041209
6. Hosni I, Théoleyre F. Self-healing distributed scheduling for end-to-end delay optimization in multihop wireless networks with 6TiSCH. *Computer Communications* 2017; 110: 103 - 119. doi: http://dx.doi.org/10.1016/j.comcom.2017.05.014

How to cite this article: Urke AR., Ø. Kure, and K.Øvsthus (2020), Layered autonomous TSCH scheduler for bounded latency, *Internet Techonology Letters*, *2020*.