LLSF: Low Latency Scheduling Function for 6TiSCH Networks

Tengfei Chang, Thomas Watteyne, Wang Qin, Xavier Vilajosana

LLSF: Low Latency Scheduling Function for 6TiSCH Networks

Tengfei Chang*,†, Thomas Watteyne†, Qin Wang*, Xavier Vilajosana‡
*University of Science and Technology, Beijing, China
†Inria-Paris, EVA Team, France
‡Universitat Oberta de Catalunya, Barcelona, Spain

***Abstract*—The 6TiSCH working group is standardizing the low-power wireless protocol stack for the Industrial IoT. The default scheduling function (SF0) standardized by 6TiSCH uses simple random slot selection. This paper proposes the Low Latency Scheduling Function (LLSF), a new scheduling function which daisy-chains timeslots rather than picking them randomly. We implement LLSF in OpenWSN and evaluate its performance experimentally. LLSF yields 82.8% lower end-to-end latency on a 5-hop path than SF0, at no extra costs.**

***Keywords*-6TiSCH, Time Synchronized Channel Hopping, Scheduling Function, 6top, Latency.**

I. Introduction

The IETF 6TiSCH [1] standardization working group defines how to combine the industrial performance of IEEE802.15.4e [2] with the ease of use of IPv6. As the core of 6TiSCH, the Time Synchronized Channel Hopping (TSCH) mode of IEEE802.15.4e requires a protocol to manage the schedule of the nodes in network. 6TiSCH defines the 6top Protocol (6P) [3], which allows neighbor nodes to directly negotiate with one another to modify their communication schedules locally. Yet, 6P does *not* define the "policy" (i.e. the algorithm) which decides (`req1`) when a node should schedule/un-schedule timeslots to its neighbor and (`req2`) which timeslots to pick. The scheduling function zero (SF0) [4], defined by 6TiSCH, is the simplest possible policy. However, SF0 *only* focuses on `req1` ("*when should a node schedule/unschedule timeslots to its neighbor*"), not on `req2` ("*which timeslots should it pick*").

This paper introduces a new Scheduling Function, called "Low-Latency Scheduling Function" (LLSF). When node `A` schedules a timeslot to node `B`, rather than picking the timeslot randomly, it daisy-chains the timeslots used in a multi-hop path. LLSF reduces the end-to-end latency on a 5-hop path by 82.8% compared to SF0, with *no* additional overhead.

II. Low Latency Scheduling Function (LLSF)

Fig. 1 illustrates LLSF. `F` monitors the number of packets it generates, and determines that an additional (transmission) timeslot needs to be scheduled to `E`. At the first hop, LLSF selects a cell *randomly* among the unscheduled timeslots in the slotframe. As part of the 6P negotiation, `E` installs the same timeslot (as a reception slot) as `F`. Now, `E` needs to schedule an additional (transmission) timeslot to `D`. Fig. 2 depicts the schedule of `E` at that point in time. Each cell in the figure represent a timeslot, with the first timeslot on the left. Node `E` has 3 reception slots from `F` (slot 2, 5 and 97).

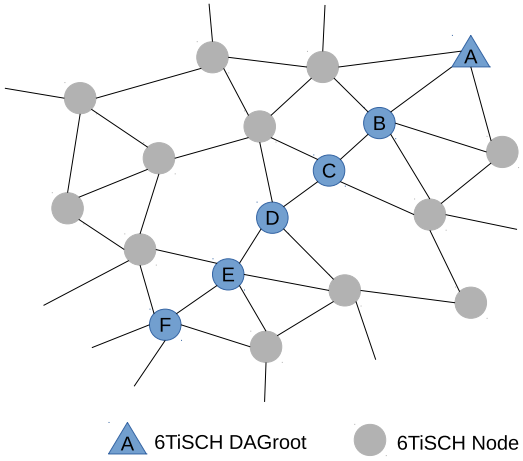


Fig. 1. A generic TSCH network rooted in DAGroot `A`. Without loss of generality, we use the highlighted multi-hop path to illustrate how LLSF works.

LLSF schedules the transmission slot in a 3-step process: **(1)** for each reception slot from the previous hop, determine the number of slots (the "gap") between that and the previous reception slot from the same neighbor (in Fig. 2, the gap between slots 5 and 97 is 91 slots wide). **(2)** pick the slot which has the largest gap to its left (slot 97 in Fig. 2). **(3)** the new transmission slot to the next hop is the closest unused slot to the right of the selected reception slot (slot 99 in Fig. 2).

Assuming at some point, `E` determines that one of the (transmission) timeslots to `D` needs to be unscheduled. Fig. 3 depicts the schedule of `E` at that point in time. Node `E` has 3 reception slots from `F` (slot 2, 5 and 97) and 4 transmission slots to node `D` (3, 6, 95, 99).

LLSF unschedules the transmission slot in a 2-step process: **(1)** for each transmission slot to the next hop, determine the number of slots (the "gap") between that and the previous reception slot from the same neighbor. For example, in Fig. 3, the gap between slots 5 (a reception slot) and 95 (a transmission slot) is 88 slots wide. **(2)** remove the transmission slot which has the largest gap to its left. In Fig. 2, slot 95.

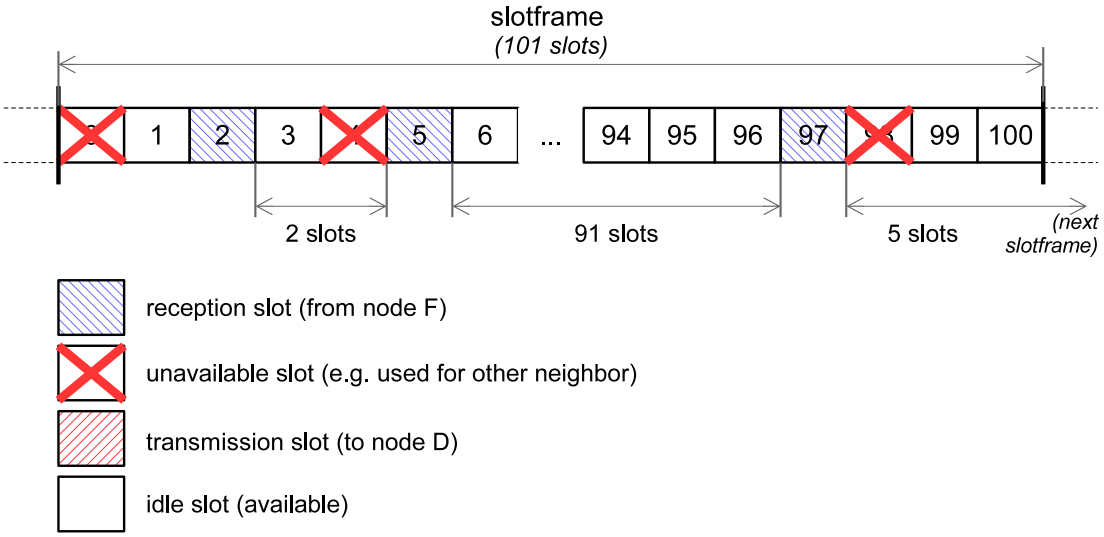


Fig. 2. The TSCH schedule of node E when LLSF *schedules* an transmit slot to node D.

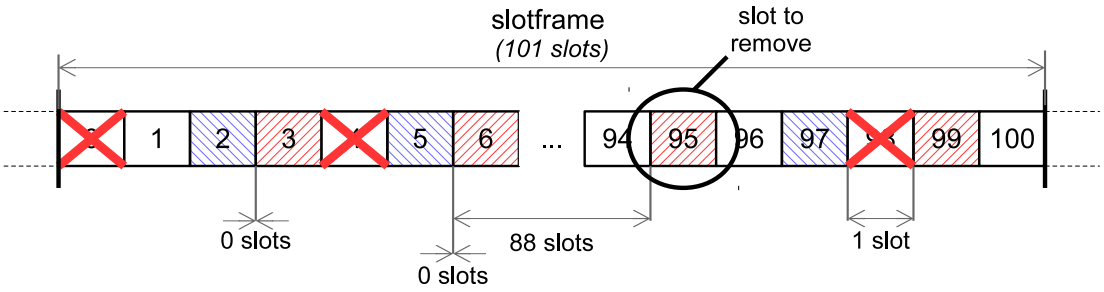


Fig. 3. The TSCH schedule of node E when LLSF *unschedules* an transmit slot to node D.

III. Evaluation

We implement LLSF in OpenWSN [5], the de-facto open-source implementation of the 6TiSCH protocol stack. We isolate the 5-hop path depicted in Fig. 1 and measure latency on a 6-node linear topology network with OpenSim, the network emulator of OpenWSN. Node A is the DAGroot of the network, node F is the one which generates traffic that travels to the DAGroot, all other nodes are just relaying data and do not generate their own application traffic.

In a TSCH network, each timeslot is associated with a unique ever-incrementing index called "Absolute Slot Number" (ASN). When generating a packet, F writes the current ASN into the packet payload. When any of the nodes in the network receives the packet, it retrieves the ASN from the packet, calculates the duration the packet has been in the network for (in timeslots) by subtracting that from the ASN when the packet was received, and writes that on its serial port.

Since subsequent packets would follow the same schedule and hence have strong correlation in latency, we choose to repeat the complete run for only one packet. At each run, F only sends one application packet to A. We repeat 100 runs for each scheduling function (SF0 and LLSF) and for each slotframe length (101, 67 and 31 slots), resulting in 600 runs. A different random number generator seed is for each run.

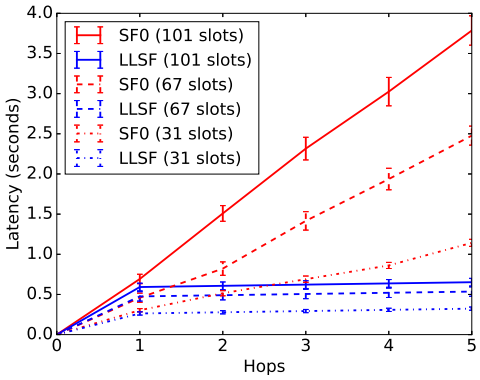
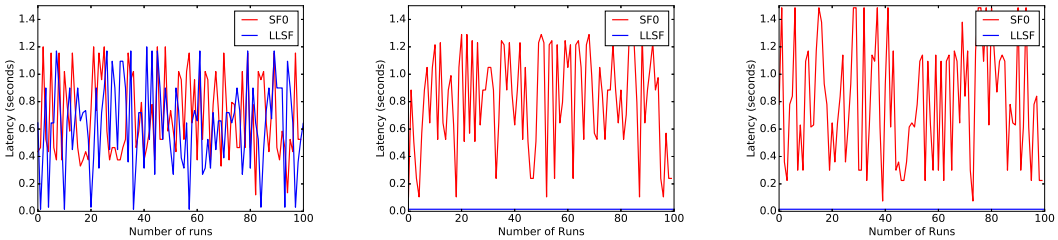


Fig. 5. Average accumulated latency using SF0 and LLSF at each hop with slotframe length of 31, 67 and 101 timeslots.

Fig. 4 shows the latency of each of the 100 packets when using both SF0 and LLSF with a slotframe of 101 timeslots. It shows how, at the first hop (Fig. 4(a)), the latency for the SF0 and LLSF is equivalent, as the data packet is generated randomly, and is buffered for an equivalent amount of time.



(a) F →E latency (first hop) (b) E →D latency (second hop) (c) D →C latency (third hop)

(d) C →B latency (fourth hop) (e) B →A latency (fifth hop)

Fig. 4. Per-hop latency when using SF0 and LLSF with a slotframe of 101 timeslots. The latency for LLSF is 15 ms for all hops except the first.

For all subsequent hops, the latency of SF0 remains the same as in the first hop, however, the average latency of LLSF drops down to 15 ms (a single slot), as receive/transmit slots are scheduled back-to-back.

Fig. 5 shows the latency an average packet accumulates as it traverses the network. At the first hop, SF0 and LLSF account for the same latency, but after the first hop, the cumulative latency of LLSF plateaus and increments only by 15 ms per slot. Fig. 5 depicts these results for 3 slotframe lengths (101, 67 and 31 timeslots). Note that only a single timeslot is scheduled by SF0 or LLSF per slotframe. For SF0, reducing the length of the slotframe from 101 to 31 slot reduces the latency by 2.65 s. For LLSF, the same change in slotframe length only reduces the latency by 0.33 s, allowing LLSF to operate with a much lower overhead (i.e. a 101-timeslot slotframe) with only $<$500 us impact on end-to-end latency.

Fig. 6 summarizes the information from Fig. 5. The average 5-hop latency using SF0 with slotframe length of 31, 67 and 101 timeslots is 1.14 s, 2.48 s and 3.79 s, respectively. The average 5-hop latency using LLSF with slotframe length of 31, 67 and 101 timeslots is 0.32 s, 0.54 s and 0.65 s, respectively. This translates into a latency reduction of 71.9%, 78.2% and 82.8% when using LLSF instead of SF0.

IV. Conclusion

This paper proposes a scheduling function which daisy-chains timeslots along a multi-hop path, so that when a node receives a packet in a slot, it can immediately retransmit it in the next slot. We call this the Low Latency Scheduling Function (LLSF).

The measurement campaign shows that, for a 5-hop scenario, the end-to-end latency when using LLSF is 82.8% lower than when using SF0. LLSF allows the slotframe to remain long, yielding both low-power operation and low latency.

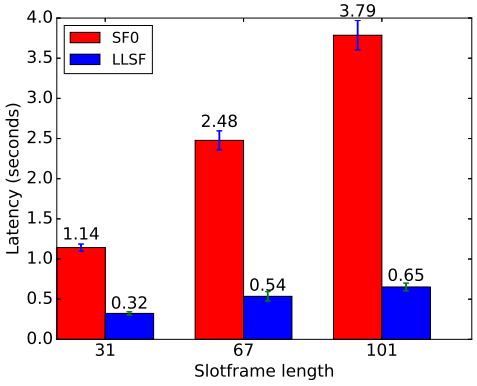


Fig. 6. Average end-to-end latency using SF0 and LLSF with slotframe length of 31, 7 and 101.

References

[1] P. Thubert, T. Watteyne, M. R. Palattella, X. Vilajosana, and Q. Wang, "IETF 6TSCH: Combining IPv6 Connectivity with Industrial Performance," in *International Workshop on Extending Seamlessly to the Internet of Things (esIoT)*, Taiwan, 3-5 July 2013.

[2] *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.

[3] Q. Wang, X. Vilajosana, and T. Watteyne, *6TiSCH Operation Sublayer (6top)*, IETF Std. draft-wang-6tisch-6top-sublayer-04 [work-in-progress], 3 November 2015.

[4] D. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura, *6TiSCH 6top Scheduling Function Zero (SF0)*, IETF Std. draft-dujovne-6tisch-6top-sf0-00 [work-in-progress], 19 October 2015.

[5] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: A Standards-Based Low-Power Wireless Development Environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, August 2012.