Parameterized Slot Scheduling for Adaptive and Autonomous TSCH Networks

(Invited Paper)

Jinhwan Jung†, Daewoo Kim†, Joonki Hong†, Joohyun Kang‡, and Yung Yi†
†Department of Electrical Engineering, KAIST, Korea
‡Department of Electrical and Electronic Engineering, Yonsei University, Korea
Email:{jhjung,dwkim,joonki}@lanada.kaist.ac.kr, jkcoin@yonsei.ac.kr, yiyung@kaist.edu

***Abstract*—Internet of things (IoT) technologies have been widely used in various applications, especially in industrial field, which often require energy efficiency, high reliability, and low delay. Recently, Time Slotted Channel Hopping (TSCH) that operates based on TDMA with channel hopping has been standardized to achieve such multiple goals. Based on TSCH protocol, the nodes are scheduled at slots by scheduling algorithm, where there is a trade-off between energy efficiency and contention among the senders. In this paper, we propose a novel slot scheduling mechanism that works on TSCH, called *PAAS* (Parameterized Adaptive and Autonomous Scheduling), which minimizes the energy consumption with the reasonably low latency while guaranteeing the reliability. PAAS is an autonomous and distributed algorithm and it works adaptively to traffic intensity, Slotframe length, and reliability requirements. To evaluate PAAS, we implement PAAS in Contiki OS and perform extensive simulations using Cooja simulator where we confirm that energy consumption is reduced by up to 63 % comparing to existing scheduling algorithms with 99.997 % reliability.**

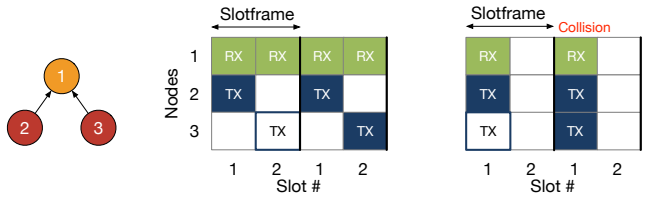
I. Introduction

A. Motivation

As the Internet of Things (IoT) comes into the spotlight, a variety of applications emerge, such as environmental monitoring, surveillance, intrusion detection and industrial automation systems, each of which has diverse traffic intensity/patterns, and target performance metrics. Since IoT sensors are typically powered by battery, energy efficiency is clearly one of the key design goals, yet other important performance metrics, e.g., high reliability and stringent delay, are becoming more important, especially in the industrial IoT applications.

In this paper, we focus on the MAC layer for IoT data delivery, where contention-based 802.15.4 standard protocols [1] (also called ZigBee) have been popularly considered. However, such contention-based protocols have limitation in providing the guarantee of the afore-mentioned multiple performance goals [2]. To tackle this challenge, more scheduling-based MAC protocols have recently considered, where Time Slotted Channel Hopping (TSCH) [3] is gaining significant attentions, standardized and proposed by IEEE 802.15.4e. The main design direction of TSCH is to use slotted medium access with channel hopping based on a given slot schedule (or simply a schedule). Thus, once a "good" schedule is constructed, hopefully in a distributed manner, depending on the pattern and the intensity of offered traffic, it is expected to achieve such multiple goals simultaneously with high possibility.

[0]This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. B0717-18-0034, Versatile Network System Architecture for Multi-dimensional Diversity)



(a) Topology (b) Reliability & latency (c) Energy efficiency

Fig. 1: Two scheduling examples in TSCH. A schedule for high reliability/latency in (b), and for high energy efficiency in (c).

An array of slot scheduling algorithms for TSCH have been proposed [2], [4]–[6] ranging from centralized to autonomous ones. The centralized scheduler in [4] is designed to provide optimality in terms of minimizing energy consumption while satisfying given reliability and delay bound, if the scheduler is aware of the traffic pattern that is assumed to be deterministic. However, it is not practical, because traffic is inherently random in many applications and also a centralized scheduling is not scalable, often requiring a lot of message passing. A decentralized and autonomous scheduler has been proposed in e.g., [6], determining a schedule in a distributed manner. However, the protocol in [6] does not explicitly consider offered traffic intensity and its pattern, which clearly need to be considered for better guarantee of energy efficiency, reliability, and latency. We refer the readers to Section I-C for more list of related work.

B. Main Contribution

In TSCH, a slot schedule refers how we place each transmitter and receiver in which slot, where a scheduling pattern is repeated over a sequence of slots, called *Slotframe*. Note that since TSCH is designed for IoT applications, it is designed on the basis of the assumption that the offered traffic is reasonably low. Thus, it is often allowed for multiple transmitters to be scheduled at the same slot, with expectation that they do not contend each other severely. The key idea of this paper is to construct a slot schedule adapting to offered traffic load, in order to choose a good trade-off point between different performance metrics, where our proposed protocol behaves in a parameterized manner.

We present the main contribution of this paper using a simple example, shown in Fig. 1. We consider a simple 3-

node network, where we have one receiver (node 1) and two transmitters (nodes 2 and 3) with two slots in one Slotframe. Assume that two transmitters generate packets with some probability, say p, during every Slotframe, and with such probability p, two packets are generated at node 2 over two consecutive Slotframes, but only one packet is generated at node 3 at the second Slotframe. Then, we compare two slot schedules in (b) and (c), respectively, where in (b), two transmitters are placed for different slots, but in (c), they are scheduled in one common slot. Then, in (b), two transmissions from nodes 2 and 3 are all collision-free, whereas in (c), on the second Slotframe, we have a collision between two transmissions $2 \rightarrow 1$ and $3 \rightarrow 1$. A collision in TSCH launches a backoff mechanism, delaying such a collided transmission, thus having negative impact on reliability and latency. However, in (c), since node 1, which is a receiver, has only to wake up once per Slotframe, the schedule is more energy efficient comparing to one in (b) where node 1 needs to be awake all the slots inside a Slotframe. Deciding which schedule is a good one actually depends on how high the traffic intensity (p in our example) is. When p is high, one naturally chooses the schedule in (b), but for small p, assigning the same slot to multiple transmitters just like in (c) may not generate serious concerns on collision, but with high energy efficiency of receivers. The main contribution of this paper lies in developing a new scheduling algorithm, called PAAS (Parameterized Adaptive and Autonomous Scheduling) in TSCH, which realizes the idea introduced in our example, so as for PAAS to adapt to offered traffic conditions and autonomously determine a schedule that jointly considers energy efficiency, reliability and latency.

C. Related Work

Prior to TSCH, various TDMA based standards such as WirelessHART [7], ISA100.11a [8] and other scheduling algorithms for low-power wireless networks have been proposed, e.g., [9]–[13], just to name a few. With focus on TSCH, a centralized scheduler, called TASA, and a decentralized scheduler, called DeTAS, for TSCH, have been proposed in [4], [5]. When all traffic information is available (e.g., the central coordinator knows when and how data packets are generated), TASA and DeTAS propose optimal scheduling solutions in terms of energy consumption with high reliability and low latency. However, due to the centralized aspects of TASA and DeTAS (which collects the traffic information at a centralized coordinator), they suffer from the lack of flexibility and scalability. Wave [2] is another distributed scheduler for TSCH but it also requires repeated control packet exchanges, which may not be desirable for high energy efficiency.

The authors in [6] propose an autonomous scheduling protocol, called Orchestra for TSCH, which is designed for flexible and scalable scheduler as much as CSMA based ones with reliable communication. Different from afore-mentioned schedulers requiring the knowledge of traffic patterns, Orchestra is traffic-agnostic, where each node builds its own schedule autonomously without any negotiation and control packet exchange. While Orchestra is designed to flexible and scalable, it is not fully optimized in achieving energy-efficiency, latency, and reliability, since it does not adaptively utilize the traffic information in constructing a schedule, as exemplified in the previous subsection. This paper improves Orchestra by applying a measurement-based traffic intensity (which is possible in many IoT applications) when we construct a schedule, so that we inherit the autonomous and distributed feature of slot scheduling from Orchestra, and further add adaptivity so as to achieve better energy efficiency with almost no extra cost.

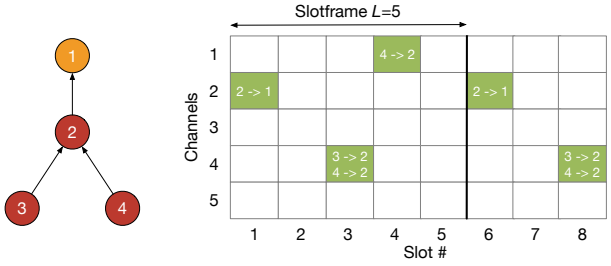


Fig. 2: Slot scheduling with channel hopping in TSCH.

II. Background: TSCH and Orchestra

This section presents a primer of TSCH [3] and Orchestra [6], which helps in understanding our proposed PAAS. The IEEE 802.15.4e [3] has been proposed, defining Time Slotted Channel Hopping (TSCH) as a new MAC protocol for highly reliable communication with low latency. In TSCH, by transmitting Enhanced Beacon (EB) periodically, every node in the network advertises its network information. TSCH has the feature of Time Division Multiple Access (TDMA) with channel hopping. All nodes in a TSCH network should be synchronized globally by transmitting and receiving packets and ACKs with its time sources as defined in the standard [3]. Time is divided into slots, at each of which one of transmitting, receiving or sleeping in terms of a node's action which is assigned. A transmitter and a receiver perform channel hopping slot by slot as defined in [3], so that they can encounter at the same channel in given slot (Fig. 2). Slotframe is the sequence of slots and during the network lifetime Slotframe is repeated, with its corresponding slot schedule being possibly changed. Each slot can be indexed by time offset and channel offset, i.e., the location within a Slotframe and the frequency to which it is assigned. In TSCH, a scheduling algorithm determines which link that is a pair of a transmitter and a receiver is assigned to which slot. The scheduler can assign a link or links to a slot, so that if only a single link is assigned, then such a slot is contention-free, otherwise a slot may have collisions, in which case a backoff mechanism runs, i.e., a node with a collided transmission re-attempts transmission with a same packet a few Slotframes after the previous attempted Slotframe. The TSCH standard does not specify a particular slot scheduling mechanism for which an algorithm such as Orchestra [6] has been proposed in literature.

Orchestra [6] is an autonomous scheduling protocol for TSCH. Each node builds its local schedule without any signal-

ing overhead, but only based on routing information (e.g., RPL [14]). In Orchestra, they propose two types of slots to deliver application packets[1]: (i) Receiver-based Shared Orchestra Slot (RBS) and (ii) Sender-based Shared Orchestra Slot (SBS) (see [6] for more details).

III. PAAS

We now present our slot scheduling algorithm, PAAS (Parametrized Adaptive and Autonomous Scheduling) in Algorithm 1. Prior to the detailed description, we first provide its overall framework.

A. Framework

PAAS consists of the following three phases: (i) Initialization, (ii) Adaptation, and (iii) Recovery. For ease of exposition of this section, we comment that we will use n-PBS as a slot scheduling mechanism whose details will be explained after this subsection, where n is the parameter which trades off multiple performance metrics.

(i) Initialization phase: We assume that a routing tree is built by the RPL protocol and each node periodically broadcasts DIO messages to update and maintain the routing tree. As a node, say v, joins a TSCH-running network in response to the existing nodes' enhanced beacons, it schedules a TX slot as 1-PBS to its parent that transmits enhanced beacon and schedules a RX slot per child only when a new node joins the TSCH network as a child of v. As time advances, node v is aware of the stable information on the volume of traffic (relayed from its children) from measurements and the number of its children, say K. Then, it moves on to Adaptation phase.

(ii) Adaptation phase: Using the knowledge of the traffic intensity, each node v except for leaf nodes determines the value of n in n-PBS following our design philosophy, where we will elaborate how to choose n in the next subsection. The parameter n is properly chosen so as to minimize energy consumption with some reliability guarantee, as exemplified in Section I. In n-PBS, each node v adaptively schedules its TX and RX slots depending on the chosen n by transmitting or receiving DIO packets which include the information about slot schedule (see Section III-B for details), so that at most n transmitters are assigned to a single slot in fully distributed manner. They keep staying in Adaptation phase, until the changes of topology or average traffic intensity are made. If v detects the change of topology or traffic intensity, it goes to Recovery phase.

(iii) Recovery phase: Node mobility or significant link quality change leads the routing topology to be reconstructed. If the routing tree or the traffic intensity is changed, nodes run recovery mechanism, where they start from running 1-PBS scheduling. During that time, nodes measure traffic intensity and the number of their child K again, as in Initialization phase, then moving to Adaptation phase for finding a more efficient schedule.

[1]In [6], two more slot types are proposed, but we exclude them in our discussion, because it is designed for broadcast and is a special case of SBS.

Algorithm 1: PAAS algorithm for each node v

Input: traffic intensity p and child list C of node v, length of Slotframe: L

Output: Scheduled TX and RX slots

1. Choosing the value of n
Given reliability requirement δ, chooses n as:

$$n = \left\lceil \min\left\{ f^{-1}(\delta), \frac{1}{p} \right\} \right\rceil. \quad (1)$$

2. Schedule RX slot(s)
Set $LIST$ to be the empty list of node IDs.
2-1. Puts IDs of every n-th node in C into $LIST$
2-2. Transmits DIO including $LIST$
2-3. Builds its RX slot as:
for i: 1 to $\left\lceil \frac{K}{n} \right\rceil$ **do**
Builds its RX at the slot of Hash($LIST[i]$) % L with *channelOffset* = Hash($LIST[i]$) % N_c, where N_c is the number of available channels

3. Schedule a TX slot
3-1. Upon receiving DIO from its parent, obtains $LIST$
3-2. **for** j: $ID(v)$ until find j in $LIST$ **do**
if j is in the received $LIST$ **then**
Builds its TX slot at Hash(j) % L with *channelOffset* = Hash(j) % N_c and **break**
else $j = (j+1)$ % $n_{\max}$, where $n_{\max}$ is the maximum node ID.

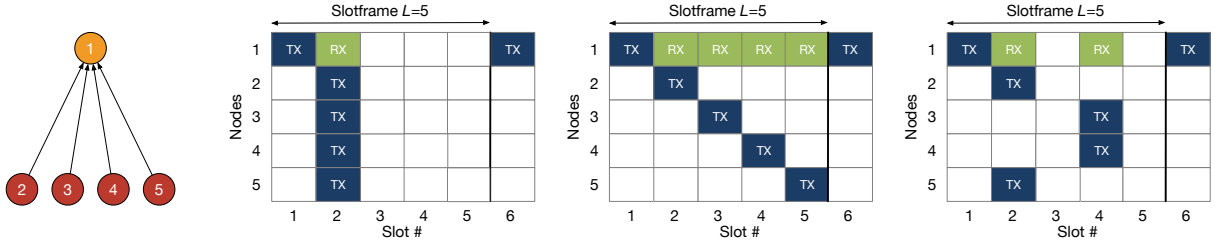
B. Algorithm Description

We now describe our scheduling algorithm PAAS which finds a slot schedule in Adaptation phase in response to the measured traffic intensity and the constructed routing tree, presented in Algorithm1. We present the description from the perspective of an arbitrary node v. Given the traffic intensity p and the child list C of v, PAAS outputs each node v's schedule for TX and RX in a fully distributed manner. By node v's schedule, we mean that the offsets of slot and channel for transmission and reception, where note that there can be a multiple of RX slots. We now elaborate on how PAAS works.

S1. In Step 1, once a traffic intensity p is measured, node v chooses the parameter n, which corresponds to the maximum number of transmitters that can be scheduled in the same slot, such that n is as large as possible while satisfying given reliability requirement δ (see Section III-C for details).

S2. In Step 2, using the chosen n and the child list C, node v schedules its RX slot(s). To assign at most n transmitters into a RX slot, node v makes a list $LIST$ that contains IDs of the subset of nodes in C such that it chooses every n-th node in C (Step **2-1**)[2]. Then, node v embeds the information of $LIST$ into DIO messages (Step **2-2**) and builds its RX slots at Hash(ID) % L based on IDs in $LIST$ with *channelOffset*, where Hash can be any hashing function (Step **2-3**), so that v has $\left\lceil \frac{K}{n} \right\rceil$ RX slots in the Slotframe and informs its children of its RX schedules.

[2]In our implementation, using the sorted list of node IDs, v chooses every n-th node sequentially



(a) Topology (b) ∞-PBS (c) 1-PBS (d) 2-PBS

Fig. 3: Scheduling example of ∞-PBS, 1-PBS and 2-PBS.

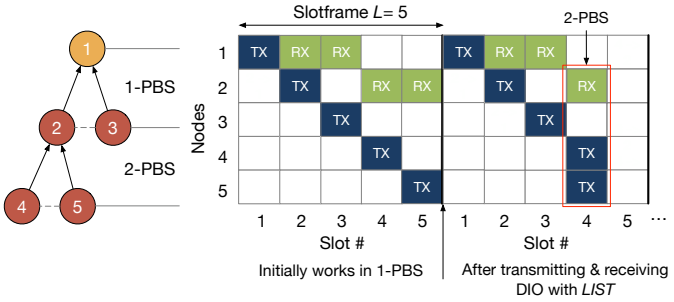


Fig. 4: An example of PAAS slot scheduling with a simple tree having height 2 and two child nodes for nodes 1 and 2. Initially all nodes are scheduled as 1-PBS. After transmitting and receiving the DIO includes $LIST$, the node 2 reschedules its RX slot and node 4 and 5 reschedule its TX slots as 2-PBS.

S3. In Step **3-1**, upon receiving a DIO packet that includes $LIST$ from its parent, node v can reschedule its TX slot by choosing one of RX slots of its parent included in $LIST$. To this end, node v tries to find $ID(v)$ in $LIST$, where $ID(v)$ is the unique ID of node v. If $ID(v)$ is in $LIST$, v schedules its TX slot at Hash($ID(v)$) % L with *channelOffset*, otherwise v finds another ID in $LIST$ by running the loop (Step **3-2**). For example, when $LIST$ includes IDs: 1, 3 and 5, among nodes from 1 to 6, nodes 1 and 2 are scheduled at slot #1 and nodes 3 and 4, and nodes 5 and 6 are scheduled at slot #3 and #5, respectively.

Example. Fig. 4 shows an example of a slot schedule with a tree with five nodes with IDs: 1,2,3,4 and 5. Once the network is initialized, all nodes work in 1-PBS. By the above rules, if node 2 chooses $n = 2$, it transmits a DIO that includes $LIST$ with ID: 4, so that the receiver (i.e., node 2) and senders (i.e., nodes 4 and 5) are scheduled at slot #4 to communicate. Assuming that node 1 chooses $n = 1$, they do not need to change schedules.

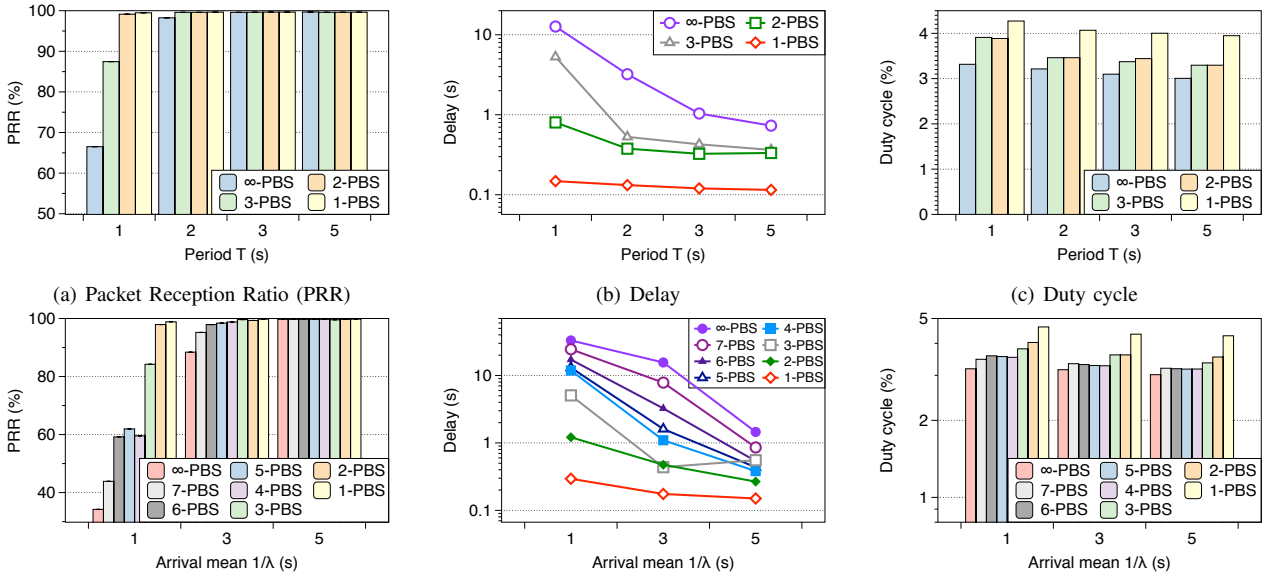
Algorithm 1 produces a slot schedule that may vary depending on the traffic intensity p, e.g., ∞-PBS, 1-PBS, or 2-PBS, as exemplified in Fig. 3 of one parent and 4 children. As we see, in ∞-PBS, many nodes are scheduled as transmitters in the same slot which have higher contention degree among nodes 2, 3, 4, and 5, yet with small energy consumption of the receiver node 1, whereas in 1-PBS, all transmitters are scheduled at different slots with less contention degree at the cost of higher energy consumption of node 1. 2-PBS is a choice between those two extremes. We comment that when a multiple of transmitters are scheduled in the same slot with their intended receiver, they are all scheduled in the same channel, as directed by the TSCH specification. Thus, in the example of Fig. 3, 2 channels for ∞-PBS, 5 channels for 1-PBS, and 3 channels for 2-PBS can be used.

Note that we allocate only one slot for node v's TX (to its parent). This is based on the assumption that the offered traffic is sufficiently low or data fusion is made, such that even relay nodes that is close to a base station, i.e., the nodes in small depth in the routing tree, can process and relay the data from their children without blowing up their queues. This is not always possible as the offered load increases with data fusion, in which case it is necessary to schedule multiple TX slots, which is left as a future work.

C. Rationale of PAAS

The key step of PAAS is how to choose the parameter n in (1) of Algorithm 1. We now present the rationale behind it. Our choice of n in (1) is motivated by a simple model and its analysis, as elaborated in what follows: Let L denote the length of Slotframe which is the number of slots in one Slotframe. Since the entire routing tree is a collection of subtrees, each of which consists of one parent and multiple children, say K children indexed by $\{1, 2, ..., K\}$, we focus on one subtree only with uplink traffic (i.e., from children to the parent) for simplicity, which, however, provides enough implication on how to choose n. In each slot, the parent wakes up and waits for an incoming packet only when the slot is scheduled as RX slot and at most n transmitters can wake up to transmit a packet only if they have some packets at scheduled slot. Define the traffic intensity (p) by the probability that a node generates a packet in one Slotframe. Assume that each child possess homogeneous traffic intensity, and the number of children is smaller than the length of Slotframe, i.e., $K < L$.

The underlying goal of PAAS lies in minimizing energy consumption while satisfying the reliability requirement which are quantified by collision probability in a single slot. The number n of TX slots of transmitters which are scheduled at the same time is an important parameter that trades off energy efficiency and collision probability. As n grows, a receiver consumes less energy but more collision occurs due to high contention and this results in low reliability and high delay. Thus, we suitably choose n under the given collision constraint



(a) Packet Reception Ratio (PRR) (b) Delay (c) Duty cycle

(d) Packet Reception Ratio (PRR) (e) Delay (f) Duty cycle

Fig. 5: Micro benchmark for Periodic and Poisson traffic with 1 receiver and 4 or 8 senders. It shows PRR, delay and duty cycle performances of n-PBS from 1-PBS to ∞-PBS.

δ by considering the number of children K and the traffic intensity p. In our design, we choose n to be:

$$n = \left\lceil \min\left\{ f^{-1}(\delta), \frac{1}{p} \right\} \right\rceil, \tag{2}$$

where $f(n)$ is a mapping from our parameter n to the collision probability for a given traffic intensity p. To derive $f(n)$, we consider the probability that more than one transmitters, which are scheduled at the same slot, try to transmit a packet in the same slotframe. Then, we have the following $f(n)$:

$$f(n) = 1 - (np + 1 - p)(1 - p)^{n-1} \tag{3}$$

In (2), choosing n comes from two requirements in relation to collision probability and stability. The energy consumption mainly comes from the receiver's idle listening. Since the receiver's energy consumption is a decreasing function of n, we can minimize the energy consumption by setting n as large as possible. However, there is an upper limit due to reliability constrains $f(n) < \delta$. Thus, we set n as a maximum number which is less than $f^{-1}(\delta)$ for given δ. Secondly, in terms of stability, the condition $n < \frac{1}{p}$ is necessary, as it ensures that the traffic arrival rate does not exceed the packet service rate. Due to space limitation, we focus on slot scheduling and omit other engineering tasks that makes our idea a working protocol. We refer the readers to our technical report for more details [15].

IV. Implementation and Evaluation

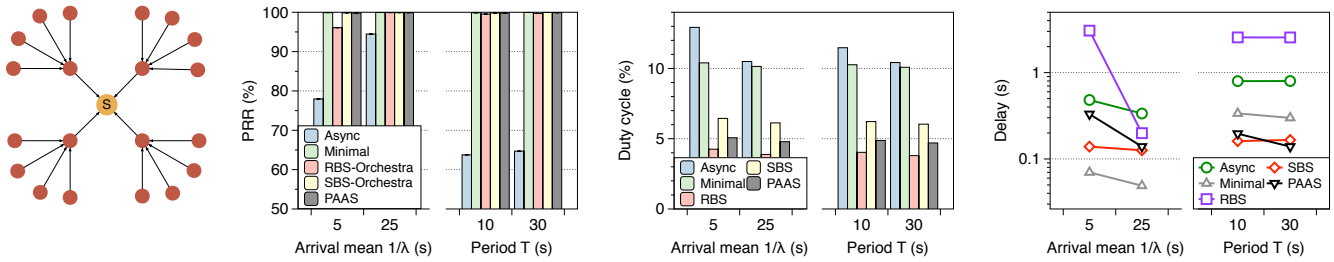
A. Implementation

We implement our scheduler PAAS on Contiki OS [16], which is an open source operating system for IoT devices. We add and modify our scheduler based on the implementation of TSCH + Orchestra [17]. We choose the objective function of RPL as MRHOF [18] that uses the hop count and ETX metrics with hysteresis property, so that the network topology changes infrequently. Using the same settings for the enhanced beacon and RPL control packets in [17], we implement a new type of unicast slot n-PBS and PAAS. Since we embed the information of slot schedules to a reserved space in the DIO, there is no extra overhead for our implementation. We also evaluate PAAS using Cooja simulator in Contiki OS to test various scenarios under a controlled environment.

B. Evaluation

We evaluate the performance of n-PBS slot type and our proposed scheduler PAAS. In this evaluation, two representative application scenarios of data collection are considered: (i) Periodic monitoring, which generates data packets periodically with period T, and (ii) event notification, which follows a Poisson process with arrival rate λ. To evaluate the performance of PAAS scheduler, we consider three metrics; (i) Packet Reception Ratio (PRR) for reliability, (ii) end-to-end delay for latency, and (iii) average duty cycle (i.e., fraction of one period in which node is active) over all nodes to quantify energy efficiency. We first show the performance of n-PBS as varying n from 1 to ∞ (i.e., the maximum number of senders assigned to the same slot) to see the impact of parameter selection in a simple subtree with one parent and four or eight children. Second, we evaluate PAAS in a full 4-ary tree with height 2 (see Fig.6(a)), by comparing with asynchronous MAC (i.e., CX-MAC, which is variation of [19]), Minimal configuration [20] and Orchestra [6] with RBS or SBS. We run simulations for each environment with multiple random seeds until each sender generates at least 1,000 packets.

Impact of parameter selection. Fig. 5 shows the reliability, the delay and the duty cycle of n-PBS. In both subtrees with four and eight children, Figs. 5(a) and 5(d) show the packet



(a) **4-ary TREE:** 21 nodes (b) Packet Reception Ratio (PRR) (c) Duty cycle (d) End-to-End Delay

Fig. 6: Evaluation result of PAAS for Poisson and Periodic traffic.

reception ratio in T =1, 2, 3 and 5 sec and $1/\lambda$ =1, 3 and 5 sec, where the Slotframe length is 17 (the black error bars correspond to the standard deviation). As the parameter n in n-PBS grows, the energy consumption decreases, however, reliability and delay performances degrade. For example, when the traffic is generated periodically with the period of 1 second, 1-PBS achieves 99.485% of reliability, however, ∞-PBS achieves 66.52% with consuming only 78% energy, compared to 1-PBS case. However, 2-PBS achieves comparable reliability (i.e., 99.145%) to that in 1-PBS with relatively small energy consumption (91% of 1-PBS). From the result of PAAS, a proper parameter to guarantee 1% collision probability is 2 in this setting. Thus, if we choose the parameter n as 2, we achieve minimizing energy consumption with 99% of reliability. The delay performance has similar pattern with reliability performance, where the delay performance improves as n decreases. This trend does not change for different traffic patterns of Periodic monitoring and event notification, as shown in Fig. 5.

Performance comparison. Under topology of the **4-ary TREE** (Fig. 6(a)), we show the performance of PAAS compared to tested protocols, see Figs. 6(b), 6(c) and 6(d), where the Slotframe length is 11. Asynchronous algorithm shows poor reliability with very high energy consumption. Even though Minimal shows high reliability with reasonably low delay, it consumes a large amount of energy. Comparing Orchestra and PAAS, since PAAS finds a good trade-off between RBS and SBS of Orchestra, PAAS shows similar reliability and delay performances (i.e., up to 99.997% and 0.139 ms) to SBS and lower energy consumption (i.e., up to 26% lower than SBS). In other words, PAAS is much better than RBS in terms of the reliability and the delay performances, while the only energy consumption is slightly more than RBS (i.e., up to 28% higher than RBS).

V. Conclusion

We proposed a Parameterized Adaptive and Autonomous Scheduling (PAAS) in TSCH for IoT data delivery in Low power and Lossy Networks. We parameterized n, which is the number of simultaneous transmission under the same reception, and defined a new type of unicast slot: n-PBS, which allows at most n transmitters to contend within the single slot. We implemented PAAS in Contiki OS, which schedules n-PBS slots by measuring traffic intensity in adaptive and autonomous manner, and evaluated the superior performance of PAAS to other competitive algorithms.

References

[1] LAN/MAN Standards Committee and others, "IEEE Standard for Local and metropolitan area networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE Computer Society Approved*, 2011.

[2] 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.

[3] 802.15.4e Task Group, "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.

[4] 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 *Proc. of IEEE PIMRC*, 2012.

[5] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, "Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things," in *Proc. of IEEE WoWMoM*, 2013.

[6] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. of ACM SenSys*, 2015.

[7] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying wireless technology in real-time industrial process control," in *Proc. of IEEE RTAS*, 2008.

[8] ISA, "Wireless system for industrial automation: process control and related applications," ANSI/ISA-100.11a-2011.

[9] Y. Wu, J. A. Stankovic, T. He, and S. Lin, "Realistic and efficient multi-channel communications in wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2008.

[10] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[11] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Transactions on Mobile computing*, vol. 11, no. 1, pp. 86–99, 2012.

[12] R. Soua, P. Minet, and E. Livolant, "MODESA: an optimized multi-channel slot assignment for raw data convergecast in wireless sensor networks," in *Proc. of IEEE IPCCC*, 2012.

[13] M. Nobre, I. Silva, and L. A. Guedes, "Routing and scheduling algorithms for WirelessHARTNetworks: a survey," *Sensors*, vol. 15, no. 5, pp. 9703–9740, 2015.

[14] T. Winter and P. Thubert, "RPL: IPv6 routing protocol for low-power and lossy networks," *IETF RFC6550*, 2012.

[15] "Technical report," https://lanada.kaist.ac.kr/pub/paas.

[16] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Proc. of IEEE LCN*, 2004.

[17] S. Duquennoy, A. Elsts, A. Nahas, and G. Oikonomou, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *Proc. of IEEE DCOSS*, 2017.

[18] O. Gnawali, "The minimum rank with hysteresis objective function," *IETF RFC6719*, 2012.

[19] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *Proc. of ACM SenSys*, 2006.

[20] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15. 4e (6TiSCH) configuration," *IETF RFC8180*, 2017.