2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS) | 978-1-6654-0386-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/RTAS52030.2021.00033

APaS: An Adaptive Partition-Based Scheduling Framework for 6TiSCH Networks

Jiachen Wang[†*], Tianyu Zhang[‡], Dawei Shen[§], Xiaobo Sharon Hu[¶], Song Han[†],
[†]Dept. of Computer Science and Engineering, University of Connecticut, Storrs, CT, 06269
[†]Email: {jc.wang, song.han}@uconn.edu
[‡]Dept. of Computing, The Hong Kong Polytechnic University, Hong Kong
[‡]Email: tianyu1.zhang@polyu.edu.hk
[§]School of Computer Science and Engineering, Northeastern University, Shenyang, China
[‡]Email: 1610547@stu.neu.edu.cn
[¶]Dept. of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, 46556
[§]Email: shu@nd.edu

***Abstract*—The past decade has witnessed the rapid development of real-time wireless technologies and their wide adoption in various industrial Internet-of-Things (IIoT) applications. Among those wireless technologies, 6TiSCH is a promising candidate as the de facto standard due to its nice feature of gluing a real-time link-layer standard (802.15.4e, for offering deterministic communication performance) together with an IP-enabled upper-layer stack (for seamlessly supporting Internet services). 6TiSCH's built-in random slot selection scheduling algorithm, however, often leads to large and unbounded transmission latency, thus can hardly meet the real-time requirements of IIoT applications. This paper proposes an adaptive partition-based scheduling framework, APaS, for 6TiSCH networks. APaS introduces the concept of resource partitioning into 6TiSCH network management. Instead of allocating network resources to individual devices, APaS partitions and assigns network resources to different groups of devices based on their layers in the network so as to guarantee that the transmission latency of any end-to-end flow is within one slotframe length. APaS also employs a novel online partition adjustment method to further improve its adaptability to dynamic network topology changes. The effectiveness of APaS is validated through both simulation and testbed experiments on a 122-node multi-hop 6TiSCH network.**

***Index Terms*—6TiSCH, partition-based scheduling, end-to-end latency, dynamic topology**

I. Introduction

Internet of Things (IoT) is a fast-growing domain that promises ubiquitous connection to the Internet, turning common objects into connected devices [1], [2]. The industrial subset of IoT, namely Industrial IoT (IIoT), aims at creating a unified sensing, computing and control framework to interconnect industrial assets with information systems and business processes, which helps streamline the manufacturing process and lead to optimal industrial operations [3]. Many industries – including but not limited to chemical process control, automotive and aerospace manufacturing [3]–[5] – will benefit from this revolution.

IIoT systems are typically deployed in distributed industrial environments, supporting a dense array of sensing and actuation devices. In the past decade, we have witnessed the rapid development of real-time wireless networking (RTWN) technologies and their wide adoption in various IIoT applications [6]–[11]. This is mainly due to their great advantages over their wired counterparts on easier deployment, enhanced mobility, and reduced maintenance cost. Among the many emerging RTWN technologies, 6TiSCH [12] has been receiving increasing attention in recent years. It has a nice feature of gluing a real-time link-layer operational technology (OT) standard (802.15.4e), for offering deterministic communication performance, together with an IP-enabled upper-layer Information Technology (IT) stack, for supporting seamless Internet services. Fig. 1 gives an overview of the 6TiSCH stack architecture and example device and gateway platforms used in our 122-node 6TiSCH network testbed.

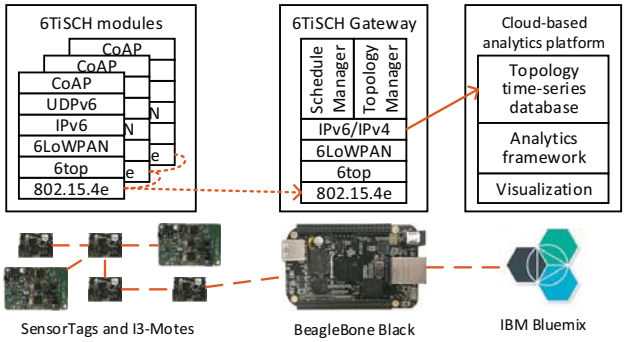


Fig. 1. Overview of the 6TiSCH architecture and hardware in our testbed.

Although 6TiSCH has high potential to become the de facto standard for real-time wireless edge networks in IIoT, its network management techniques are still rudimentary and under development. Most of the 6TiSCH networks currently deployed in industrial fields adopt a randomized packet scheduler at the data link layer. This randomized scheduler allocates network resources (in the form of cells to be described in detail later) randomly to the requesting communication tasks. Such an approach, although can lead to good network resource utilization, may cause high transmission latency for multi-hop flows and even violate their end-to-end timing requirements.

We use actual measured data obtained from example 6TiSCH networks to show the impact of the randomized scheduler on end-to-end latencies. Fig. 2 illustrates two

*The first two authors have equal contribution to this work.

978-1-6654-0386-3/21/$31.00 ©2021 IEEE
DOI 10.1109/RTAS52030.2021.00033

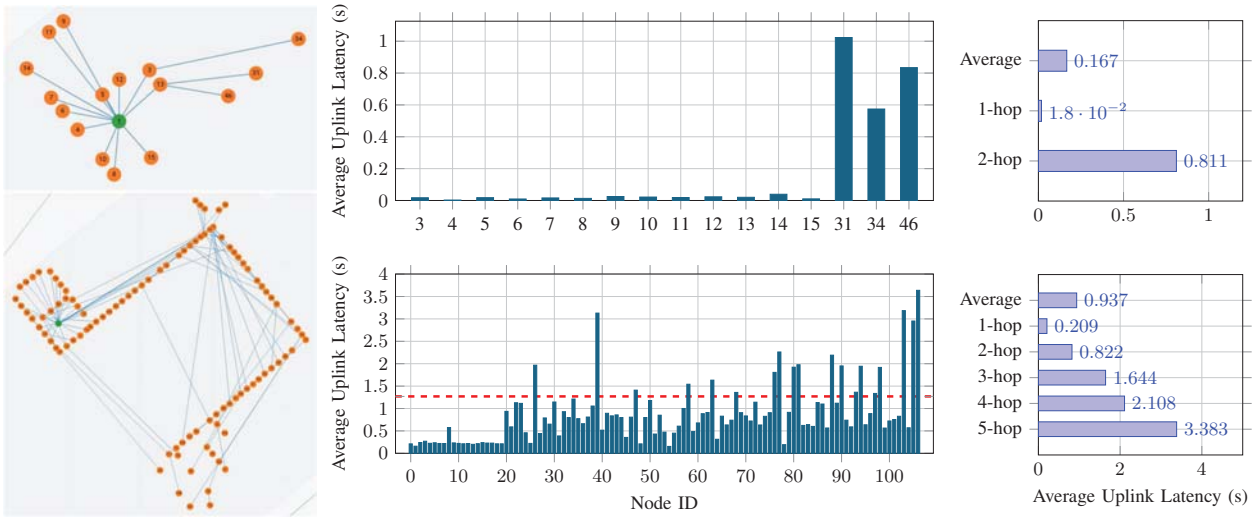


Fig. 2. Average uplink latency in two multi-hop 6TiSCH networks with the built-in randomized scheduler.

6TiSCH networks that we created in our testbed, one with 17 nodes organized into a 2-hop network and the other with 101 nodes forming a 5-hop network topology. In both networks, each device is configured to publish its sensor readings to the gateway periodically, with period set to be 1.27 second (*i.e.*, the slotframe length of the network). From the collected statistics on the uplink transmission latency, we have the following observations. (i) The end-to-end latency can be rather long even for very few hops. In the 2-hop 17-node network, although the average uplink transmission latency is only 0.167 second, the average latency of the 2-hop flows is already close to one second. (ii) The uplink transmission latency grows quickly along with the increase in the network size and hop counts. In the 5-hop 101-node network, the average uplink transmission latency of the 5-hop flows can reach 3.38 seconds which is almost three times larger than the designated timing requirement.

The above limitation of the randomized scheduler is exacerbated by the fact that 6TiSCH networks are typically deployed in industrial environments where interference and disturbance happen throughout the network lifetime. Those harsh environments cause the network nodes to frequently change their associated parents to seek for better communication links and thus change the network topology. Although several methods have been proposed in the literature to minimize the latency of multi-hop flows in 6TiSCH networks (see Section VIII for a detailed discussion), they are mainly designed for constructing communication schedules for static networks. They either cannot handle dynamic topology updates in a timely fashion, or incur significant network overhead for updating the schedules.

In this paper, we tackle the problem of link scheduling at data link layer for 6TiSCH networks by introducing an adaptive partition-based scheduling framework, APaS. APaS aims to guarantee the end-to-end packet transmission latency in multi-channel multi-hop 6TiSCH networks, even in the presence of frequent network topology changes. APaS employs the concept of resource partitioning. Specifically, it divides a slotframe into multiple partitions, and assigns the partitions to specific links based on the links' types and distances to the gateway. Each partition contains a group of cells to not only ensure that the latency requirements by the current workload are satisfied but also provide reservation for future demands due to topology changes. Our main contributions are summarized below.

1) We introduce a novel partition-based, link-type guided scheduling scheme at data link layer to help satisfy the end-to-end real-time requirements of multi-hop flows. The scheme divides time slots into different partitions to schedule links of different types (uplink, downlink, broadcast) and hop counts to the gateway. As long as a link is assigned in the right partition, the transmission latency of end-to-end flows is guaranteed to be within the length of a single slotframe.
2) To improve the network scalability and support more devices, we extend the baseline partition-based scheme to perform further partitioning in the channel dimension to allow channel reuse among different partitions.
3) We propose an efficient online partition adjustment method to adjust allocations and boundaries of the partitions for adapting to topology changes.
4) We implement APaS on a 122-node 6TiSCH network, and validate the effectiveness of APaS through both simulations and testbed experiments.

The remainder of the paper is organized as follows. Section II describes the system model and presents the problem statement. Section III introduces the baseline design of the adaptive partition-based scheduling framework, APaS. Section IV presents a two-dimensional partitioning scheme to further increase the network scalability. Section V details the dynamic partition adjustment method to improve the adaptability of APaS. Section VI and VII evaluate the performance of APaS via simulations and real-world testbed experiments, respectively. Section VIII summarizes the related work. Section IX concludes the paper and discusses future work.

II. System Model and Problem Statement

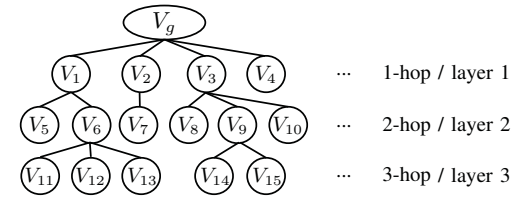
In this section, we present the 6TiSCH network model and describe the problem to be studied in this work.

A. Network Model

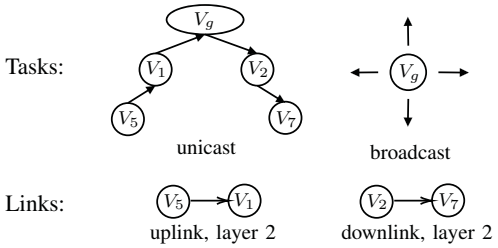
We adopt a typical RTWN model, in which sensors and actuators are wirelessly connected to a controller node (the gateway) either directly or through one or multiple relay nodes. The network topology is modeled as a tree $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where node set $\mathbf{V} = \{\{V_0, V_1, \cdots, \}, V_g\}$ and root node V_g represents the gateway. At any given time, each node is only associated with one parent node but can have multiple child nodes. If two nodes are directly connected (without any intermediate nodes) and one node has fewer hops to the root node than the other, we say the former is the parent of the latter and the latter is a child of the former. Link $e_{i,j} \in \mathbf{E}$ represents the directed wireless communication between nodes V_i and V_j, where V_i is the sender and V_j is the receiver. Each link is associated with two attributes, $type$ and $layer$. The type of link $e_{i,j}$ is uplink (downlink) if V_i is the child (parent) and V_j is the parent (child). The layer of a link is the child node's hop count to the gateway. We use $e(U, l)$ or $e(D, l)$ to denote an uplink or a downlink in layer l, respectively. Fig. 3(a) shows a 6TiSCH network topology with 16 nodes and three layers.

An end-to-end (e2e) communication task (or simply task) in a 6TiSCH network can be either a broadcast task or a unicast task. A broadcast task is originated from the gateway and is responsible for broadcasting network-wide configuration information to all the nodes. A unicast task typically originates at a sensor node. Following real-life RTWN settings, we assume that each unicast task *periodically* samples the environment and passes sensor readings along a pre-defined uplink routing path to the gateway for data collection and control decision-making. The generated control signals at the gateway are then forwarded along a pre-defined downlink routing path to an actuator node for execution. The information transmitted for one instance of a task is referred to as a packet. See Fig. 3(b) for the task related concepts and an example task.

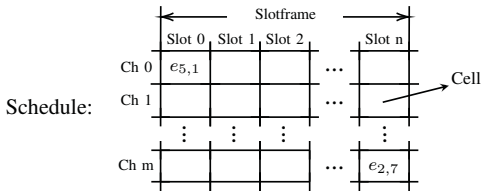
In our network model, the 6TiSCH network employs a Time-Division Multiple Access (TDMA) based data link layer and adopts a *centralized* link-based scheduler to allocate network resource for individual links [13]. We use the concept of *cell* (denoted as $x_{s,c}$) to represent the basic resource unit that can be allocated to individual links, where s is the time slot offset and c is the channel offset. Consecutive time slots in a 6TiSCH network are grouped into a slotframe. The assignment of the cells to individual links in the slotframe defines the communication schedule of the network and the schedule repeats every slotframe during the course of the network operation. When a device joins the network and a task is created, the scheduler allocates specific cells to the links along the routing path of the task; when a node leaves the network, the assigned cells are revoked[1]. We use



(a) A 6TiSCH network topology with 16 nodes and three layers.



(b) Example for task, communication, and links.



(c) Examples for cell, slotframe, and schedule.

Fig. 3. Important concepts of network resource management in 6TiSCH.

$A(x_{s,c}) = e_{i,j}$ to denote that cell $x_{s,c}$ is allocated to link $e_{i,j}$ and use $AL(x_{s,c}) = (U, l)$ to denote that $x_{s,c}$ is allocated to an uplink in layer l. If cell $x_{s,c}$ is not assigned to any link in the communication schedule, we say it is an idle cell. Fig. 3(c) illustrates the cell and schedule concepts in our network model.

B. Problem Statement

A key performance metric of 6TiSCH networks is the e2e packet transmission latency which is defined as the time duration between the generation of the packet at the sensor node's application layer and the reception of the control signals at the actuator's application layer. We assume that all tasks have the same e2e latency requirement that is bounded by the slotframe length[2]. Our work aims to guarantee the e2e transmission latency in multi-hop 6TiSCH networks, even in the presence of frequent network topology changes. Our work uses the following two assumptions.

Assumption 1. Instead of directly focusing on the e2e application layer latency, we consider the e2e MAC layer latency, which is defined as the time duration between the transmission of the packet at the sensor node's MAC layer and the reception of the packet at the actuator node's MAC layer. We make this assumption because the data link layer scheduler is not aware of the generation time of the packet at the application layer which can be arbitrary, and the e2e MAC layer latency is a

[1]Note that when a node leaves the network, all its descendants rejoin the network by choosing a new parent. This will incur topology change(s).

[2]The APaS framework can be readily extended to be applied in the scenario where the latency requirements of the e2e tasks are diverse.

more precise performance metric to evaluate the effectiveness of data-link layer scheduling schemes.

Assumption 2. We assume that for each task, the number of cells required by each link along its routing path has been determined and each packet is assigned with sufficient number of cells to transmit within each slotframe. We make this assumption because extensive studies (*e.g.* [14], [15]) have been conducted on how to decide the proper number of cells for individual links in static networks to reduce the queuing delay. Thus, this work focuses on (i) how to appropriately allocate the determined number of cells for individual links in a slotframe to guarantee the e2e MAC layer latency, and (ii) how to adjust the allocation with moderate network overhead in the presence of network topology changes.

Overall, the design objectives of our proposed scheduling framework for 6TiSCH networks include: (i) guarantee the e2e MAC layer latency of each packet to be within one slotframe length; (ii) achieve high network scalability to support more devices and tasks running in the network; and (iii) adapt to network topology changes in an online and efficient manner, so as to satisfy the e2e MAC layer latency requirements.

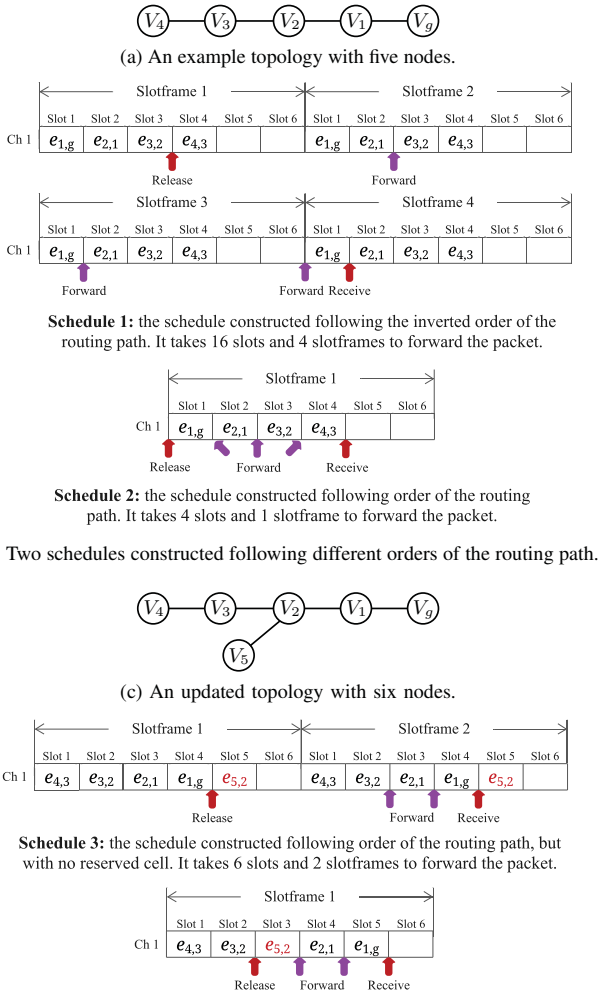
To achieve these design objectives, we introduce an adaptive partition-based scheduling framework, APaS, which consists of three key components. First, a baseline partition-based scheduler is designed to achieve objective (i), where cells in the schedule are allocated to links in groups according to their types and layers. Second, the baseline design is extended to a two-dimensional partition-based method to perform further partitioning in the channel dimension to support channel reuse by different partitions to realize objective (ii). Finally, a dynamic partition adjustment strategy is developed in response to online dynamic topology changes to achieve objective (iii). Details of these innovations are presented in Section III, Section IV and Section V, respectively.

III. Baseline Partition-based Scheduling Approach

As described in the introduction section, a major limitation of the randomized scheduler in 6TiSCH networks is that it may cause high e2e transmission latency for multi-hop flows and violate their e2e timing requirements. In this section, we first present a motivating example to show that an employed cell allocation strategy (*i.e.*, schedule) can significantly impact the e2e latency, especially in the presence of topology changes. Based on two important observations made from this example, we then describe a baseline design of the proposed partition-based, link-type guided scheduling scheme.

A. Motivating Example

6TiSCH is a representative type of TDMA-based multi-channel multi-hop real-time wireless networks. In such networks, each packet has to wait for the allocated cell(s) for each hop, and the e2e latency of the packet is an accumulation of the waiting time of all its transmissions on the routing path from the sensor node to the actuator node. In our study, we observed that the following two key factors can affect the transmission



(a) An example topology with five nodes.

Schedule 1: the schedule constructed following the inverted order of the routing path. It takes 16 slots and 4 slotframes to forward the packet.

Schedule 2: the schedule constructed following order of the routing path. It takes 4 slots and 1 slotframe to forward the packet.

(b) Two schedules constructed following different orders of the routing path.

(c) An updated topology with six nodes.

Schedule 3: the schedule constructed following order of the routing path, but with no reserved cell. It takes 6 slots and 2 slotframes to forward the packet.

Schedule 4: the schedule constructed following order of the routing path with one cell reserved. It takes 3 slots and 1 slotframe to forward the packet.

(d) Two schedules constructed with and without reserved cells.

Fig. 4. Comparison of e2e packet transmission latencies for four different schedules. The latency in each schedule is indicated by the number of time slots between the "release" red arrow and the "receive" red arrow. The purple arrows indicate the slots where forwarding takes place.

latency of a packet: i) the sequence (or ordering) of the cells allocated to the links along the routing path of the packet, and ii) the positions of these cells in a slotframe. In the following, we will use an example to illustrate how the two factors affect the e2e transmission latency.

For simplicity of discussion, we consider a single-channel 6TiSCH network with a slotframe size of 6 and an uplink packet transmission with routing path $V_4 \rightarrow V_3 \rightarrow V_2 \rightarrow V_1 \rightarrow V_g$ (see Fig. 4(a)). If the communication schedule is constructed following an inverted order of the packet routing path (Schedule 1 in Fig. 4(b)), 4 slotframes are needed to transmit the packet from the sensor (V_4) to the gateway (V_g). More specifically, after the first-hop transmission via cell $x_{4,1}$ allocated to link $e_{4,3}$ in the first slotframe, the second-hop transmission has to wait for the next available cell $x_{3,1}$ allocated to link $e_{3,2}$ in the next slotframe. Similarly, the third-hop and fourth-

hop transmissions are transmitted in the cells $x_{2,1}$ and $x_{1,1}$ in the third and fourth slotframes, respectively. Thus, it takes 4 slotframes to complete the transmission of the packet and the e2e latency is 16 slots. However, if we allocate the cells following the sequence of the links along the routing path (Schedule 2 in Fig. 4(b)), 4 slots within one slotframe are sufficient. This leads us to the following observation.

Observation 1 (Cell allocation sequence)**.** *If the cells assigned to the links of a packet are allocated in the slotframe following the packet's routing sequence (i.e., the order of the links along the packet's routing path starting from the sensor node), the packet experiences a shorter e2e transmission latency.*

Besides the cell allocation sequence in a slotframe, another key factor that affects the e2e transmission latency of a packet is the network topology change, which in turn may change the routing path of the packet. In 6TiSCH networks, a topology change usually happens during the network operation when new node(s) join the network, old node(s) leave the network, or existing node(s) change their parents.

Fig. 4(c) shows a scenario when a new node V_5 joins the network and chooses node V_2 as its parent. Assume that a packet is generated at node V_5 with node V_g set as its destination. If Schedule 3 as shown in Fig. 4(d) is adopted, there is no idle cell between cells $x_{2,1}$ and $x_{3,1}$ which are allocated to link $e_{3,2}$ and link $e_{2,1}$, respectively. In this case, we have to allocate the first idle cell ($x_{5,1}$) in the slotframe to link $e_{5,2}$. This schedule violates the routing sequence of the packet, thus does not satisfy Observation 1. The e2e transmission latency of the packet from node V_2 to V_g is 6 slots. By contrast, if we leave cell $x_{3,1}$ idle before node V_5 joins the network, we can use Schedule 4 (as shown in Fig. 4(d)) after V_5 joins the network. Since Schedule 4 satisfies Observation 1, *i.e.*, all the cells follow the routing sequence of the packet even after the topology change, the e2e packet transmission latency is reduced to 3 slots. This example leads us to the following observation.

Observation 2 (Idle cell reservation)**.** *Idle cells should be judiciously reserved during the network operation so as to help satisfy Observation 1 when topology changes occur.*

Although reserving idle cells may negatively impact the e2e transmission latency of the existing packets, it can significantly improve the e2e latency when new devices join the network in the future. The above observations provide useful guidelines to reduce the e2e transmission latency for both existing and newly joining devices. Then the question is how to utilize these observations to construct the communication schedule, especially when considering potential future topology changes.

B. Design of the Baseline Partition-based Scheduler

We now present an adaptive partition-based scheduling scheme to allocate cells to links for a given network topology. The key idea behind the scheduling scheme is to construct schedules that satisfy Observation 1& 2. Before we go into the details, we first introduce the concept of compliant schedule.

Fig. 5. Slotframe layout of the base-line partition-based scheduler with one broadcast partition and 3 partitions for uplinks and downlinks each.

Definition 1 (Compliant Schedule)**.** *A schedule S is a compliant schedule if it satisfies the following three conditions.*

Condition 1. *Cells allocated to all the uplinks are placed before the cells allocated to all the downlinks in the slotframe. That is, $\forall AL(x_{s_1,c_1}) = (U, l_1), AL(x_{s_2,c_2}) = (D, l_2)$, it holds that $s_1 < s_2$.*

Condition 2. *Among all the cells allocated to the uplinks, the ones allocated to any links with larger layer indices are scheduled before those cells allocated to the links with smaller layer indices in the slotframe. That is, $\forall AL(x_{s_1,c_1}) = (U, l_1), AL(x_{s_2,c_2}) = (U, l_2)$, if $l_1 > l_2$, it holds that $s_1 < s_2$.*

Condition 3. *Among all the cells allocated to the downlinks, the ones allocated to any links with smaller layer indices are scheduled before those cells allocated to the links with larger layer indices in the slotframe. That is, $\forall AL(x_{s_1,c_1}) = (D, l_1), AL(x_{s_2,c_2}) = (D, l_2)$, if $l_1 < l_2$, it holds that $s_1 < s_2$.*

Based on the definition of compliant schedule, we have the following lemma.

Lemma 1. *The e2e transmission latency of each packet in a compliant schedule is bounded by one slotframe length.*

The proof of Lemma 1 is straightforward and its detail is omitted due to the page limit. According to Assumption 2 in the problem statement, the number of cells needed by each link along a packet's routing path is known and can be accommodated in one slotframe. Further, according to Definition 1, all the cells allocated to the links in the network are compliant with their routing sequences. These two facts together guarantee that the e2e transmission latency of each packet does not exceed one slotframe length.

Based on Lemma 1, we propose a baseline partition-based, link-type guided scheduling scheme that constructs a compliant schedule by partitioning each slotframe in the time dimension. Specifically, we first divide each slotframe into three *super-partitions* in the time dimension. The three super-partitions are allocated to uplinks (U), downlinks (D) and broadcast tasks (B), respectively, and scheduled in the slotframe according to Condition 1 in Definition 1. The uplink and downlink super-partitions are then further divided into L *partitions* where L is an estimate of the maximum number of layers in the network. Uplink and downlink partitions are scheduled in the slotframe according to Condition 2 and Condition 3 in Definition 1, respectively. That is, the uplink/downlink partitions are arranged in the descending/ascending orders of their associated layers. Fig. 5 shows an example of the partitioned slotframe with one broadcast super-partition and 3 partitions for uplinks and downlinks each. In the rest of the paper, when it is not necessary to distinguish between partition and super-partition, we simply refer to both as partition.

Note that, initially we divide the slotframe equally among all the partitions. If the total number of cells required by the task set exceeds the capacity of the allocated partition, the online partition adjustment method (to be elaborated in Section V) is triggered to adjust the size and boundaries of the partition to adapt to the change(s) from either the network topology or the task specification. Below, we give the formal definitions of uplink/downlink/broadcast partitions.

Definition 2 (Uplink/Downlink Partition)**.** *An uplink (downlink) partition, denoted as $P_{U,l}$ ($P_{D,l}$), is a set of consecutive time slots in the slotframe. Each cell in $P_{U,l}$ ($P_{D,l}$) can only be allocated to an uplink (downlink) in layer l.*

Definition 3 (Broadcast Partition)**.** *A broadcast partition, P_B, is a set of consecutive time slots in the slotframe. Each cell in P_B can only be allocated to broadcast tasks.*

After a slotframe is partitioned, the next question is how to allocate the cells in each partition to the corresponding links, *i.e.*, constructing a compliant schedule. Since each node in a 6TiSCH network is equipped with a single omni-directional antenna, a node can only receive (send) a packet from (to) *one* child node through an uplink (downlink) within one time slot. Thus, a cell allocation should satisfy the following constraint.

Constraint 1. *Cells with a same slot offset cannot be allocated to links with a same sender or receiver.*

Given that the number of slots in each partition is limited and more slots may be needed when the network topology or task specification changes, we desire to use the minimum number of slots in a partition $P_{U/D,l}$ to satisfy the required number of cells for all uplinks/downlinks in layer l. We thus formulate the following slot minimization problem.

Problem 1 (Slot Minimization)**.** *Given partition $P_{U/D,l}$ in a 6TiSCH network with M channels and N uplinks/downlinks in layer l, determine a cell allocation (i.e. schedule S) that can provide the number of cells required by the N links while using the minimum number of slots in $P_{U/D,l}$ (denoted as ρ).*

We introduce an optimal solution below to solve Problem 1. For simplicity of presentation, we describe the solution for the case where all the N links are uplinks and each link requires one cell. The solution can be readily extended to the cases where all links are downlinks or each link requires an arbitrary number of cells. Let α be the maximum number of links associated with a same receiver. Let $\beta = \lceil N/M \rceil$ which denotes the minimum number of slots that needs to be allocated to all the links if Constraint 1 is not considered. We first claim that $\rho = \max\{\alpha, \beta\}$. Below, we describe how to allocate the cells using ρ slots and prove its optimality.

Optimal Cell Allocation Policy (OCAP). (1) Group N links into link groups, where all the links in a link group, LG_k ($1 \leq k \leq I$), share a common receiver (*i.e.*, parent node) and I is the total number of link groups. **(2)** For each LG_k ($1 \leq k \leq I$) in the decreasing order of the number of links in LG_k, allocate the links to the cells in $P_{U,l}$ from right to left in the time dimension (*i.e.*, in a decreasing order of the slot offset), and from top to bottom in the channel dimension. Once ρ cells in a channel are occupied and there are still links in LG_k to be handled, these links are allocated to the cells in the next channel again from right to left in the time dimension.

It is easy to see that following OCAP, the number of slots used by all the links is equal to ρ according to its definition. The lemma below guarantees that OCAP satisfies Constraint 1.

Lemma 2. *Links associated with the same receiver are always assigned to cells with different slot offsets under the optimal cell allocation policy, OCAP.*

Proof. For an arbitrary group LG_k, according to the definition of ρ, the number of links in LG_k is less than or equal to ρ, *i.e.* $|LG_k| \leq \rho$. There are two cases when assigning cells to links in LG_k.

Case 1: All the links in LG_k are assigned to cells in the same channel. Obviously, in this case, all the links in LG_k are assigned to different slots.

Case 2: Links in LG_k are assigned to cells in two different channels. Suppose s^* and s' are the slot offsets of the right partition boundary and the cell allocated to the first link in LG_k, respectively. According to OCAP, links in LG_k are assigned to cells in a decreasing order of their slot offsets (*i.e.*, $s', s'-1, \dots$). Once the cell with slot offset $s^* - \rho + 1$ in the current channel is occupied by a link in LG_k (*i.e.*, hitting the left boundary of the partition), the remaining links in LG_k are assigned to cells in the next channel starting from slot offset s^*. Since $|LG_k| \leq \rho$, the slot offset of the cell allocated to the last link in LG_k must be larger than s'. That is, all the cells allocated to links in the next channel are to the right of the cells in the previous channel. □

Theorem 1. *OCAP uses the minimum number of slots to allocate N links in $P_{U/D,l}$ while satisfying Constraint 1.*

Proof. According to Lemma 2, OCAP satisfies Constraint 1. Next, we prove that ρ, the number of slots allocated by OCAP to N links in $P_{U/D,l}$, is the minimum number of slots required. If $\rho = \alpha$, the number of allocated slots cannot be less than ρ since all α links associated with the same receiver must be assigned into cells with different slot offsets. If $\rho = \beta$, the conclusion directly holds according to the definition of β. □

The computational complexity of OCAP is $O(n^2)$ where n is the number of links. Thus, the computation overhead of OCAP is scalable considering that the algorithm runs on the gateway. Fig. 6 shows an example schedule constructed by the proposed partition-based scheduler for the 16-node 6TiSCH network as shown in Fig. 3(a). As the network consists of 3 layers, 3 uplink and downlink partitions are created to allocate cells for uplinks and downlinks in the corresponding layers. Each partition contains 4 slots. As can be observed from Fig. 6, the generated schedule is a compliant one, and the e2e transmission latency of all the packets are guaranteed to be within one slotframe length according to Lemma 1.

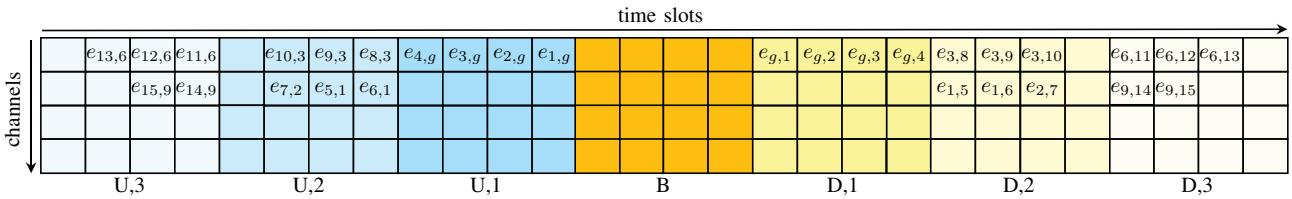


Fig. 6. A schedule constructed by the baseline partition-based scheduler for the 16-node 6TiSCH network in Fig. 3(a). Assume each link needs one cell.

IV. Two-Dimensional Partition-based Scheduling

The baseline partition-based scheduler presented in Section III-B can guarantee that the e2e transmission latency of all packets are bounded by one sloftframe length, if the two assumptions in the problem statement are satisfied. However, the baseline scheduler may have rather low cell utilization, which leads to low schedulability especially as the network size grows. In this section, we first discuss the root cause of low cell utilization and then propose a novel two-dimensional (2D) partition-based scheduling approach to address this issue.

A. Cell Utilization Challenge

In the baseline partition-based scheduler, partitioning is done along the time dimension. Hence all the cells along the channel dimension in the same time slot belong to the same partition, which can lead to some cells not usable even when other partitions need more cells. Consider the example in Fig. 6, where all the cells in the first channel in partition $P_{U,1}$ are allocated. Suppose a new cell is requested by some uplink in layer 1 due to either a new device joining or a new task. It is clear that this request cannot be accommodated in $P_{U,1}$ since all the links in layer 1 share the same receiver (gateway). This limitation is mainly caused by Constraint 1 which states that links with a same sender or receiver cannot be assigned in the cells with a same slot offset. Though the cells in the lower two channels in $P_{U,2}$ could be used for this new request, the baseline schedule restricts these cells for the second-layer uplinks only.

In general, in the baseline partition-based scheduler, once the number of cells requested by the links with a same sender (receiver) reaches the corresponding partition size, no further cell request from the same sender (receiver) can be accommodated even when idle cells still exist in that partition or other partitions. This leads to low cell utilization. The problem becomes more severe for higher-layer partitions since links in these layers tend to have fewer packets to transmit (as these links are shared by fewer routing paths) so most of the cells in such partitions are wasted. Given that 6TiSCH networks often need to deal with topology changes (such as new nodes joining or existing nodes changing their parents/children), this cell utilization challenge can be a large obstacle to handling topology changes under the baseline partition-based scheduler, which significantly reduces the scalability of the system.

To address this low utilization challenge, we note that according to OCAP, cells in each partition are allocated in a top-down fashion in the channel dimension. Thus, idle cells in the bottom channels can be allocated to links in other

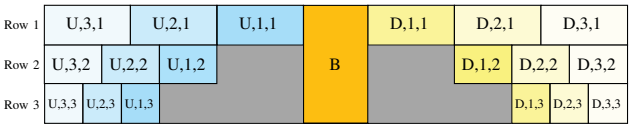


Fig. 7. Slotframe layout of the 2D partition-based scheduler with 3 rows. Each row has 3 sub-partitions for both uplinks and downlinks.

partitions. In Fig. 6, for example, if a link in $P_{U,1}$ requests an additional cell, this link can be allocated to cells in the bottom channels in $P_{U,2}$. This insight motivates us to perform judicious partitioning not only in the slot dimension but also in the channel dimension. Then the question is how to guarantee the transmission latency bound when designing such a 2D partition-based schedule.

B. Two-Dimensional Partition-based Scheduling Approach

We introduce a novel 2D partition-based scheduling approach to overcome the low cell utilization challenge. The approach is built on two key ideas: i) let links in lower layers (*i.e.*, layers closer from the gateway and having smaller layer index) borrow cells from higher layer partitions; and ii) the lower the layer, the more cells it can borrow. it is easy to see that the ideas are derived from the discussions in Section IV-A. The 2D partition-based scheduler consists of four major parts: (1) divide a slotframe into multiple *rows* in the channel dimension, where each row consists of a group of channels; (2) divide the slotframe in each row into partitions; (3) assign links to the partitions in each row; (4) allocate cells in each partition to specific links. The objective of this 2D partition-based scheduler is to improve the network scalability while ensuring that the constructed schedule is still a compliant schedule. Below we describe the scheduler in detail.

The first two parts of the scheduler are to create partitions in both the time and channel dimension. The resulting partition layout should help the scheduler to eventually generate a compliant schedule. Thus, the partition layout must satisfy Condition 1 in the compliant schedule definition. That is, cells allocated to all the uplinks are scheduled before the cells allocated to all the downlinks in the slotframe. The following process achieves the goal outlined above.

2D Partition Layout Generator (2D-PLG). (1) Divide a slotframe into R rows according to the network specification, including the max hop count, slotframe length, partition size and the topology. (2) For the first row, create partitions by applying the baseline partition-based scheduler. (3) For the immediate next row, the uplink partitions start from the beginning of the slotframe and ends at the left boundary of

$P_{U,1}$ in the previous row. Similarly, the downlink partitions start from the right boundary of $P_{D,1}$ and ends at the end of the slotframe. The *relative* sizes (i.e., the number of time slots) of the partitions in the current row are the same as those in the previous row. (4) Repeat (3) for the remaining rows.

Fig. 7 illustrates an example slotframe layout obtained from applying 2D-PLG. Here 3 rows are considered and $(U/D, l, r)$ represents a uplink/downlink partition allocated for layer-l links in row r. The sizes of the partitions within each row are the same. We use $P_{U/D,l,r}$ ($l \in [1, L], r \in [1, R]$) to denote an uplink (downlink) partition where r is the row index. (Note that some cells (*e.g.*, the grey ones in Fig. 7) do not belong to any partitions under the 2D partition-based scheduler. Those cells will be used as buffer to be assigned to links in the presence of dynamic topology change(s). This will be elaborated in the next section.) It is easy to see that the partition layout obtained by 2D-PLG indeed satisfies Condition 1.

Next, we present the details of the third part in the 2D partition-based scheduler, *i.e.*, assigning links to the partitions in each row. Condition 2 and Condition 3 in the compliant schedule definition collectively specify some requirements when assigning links to partitions. Though they are appropriate conditions for the baseline scheduler, they are over-constraining for the 2D case. Hence, we first discuss how to relax these conditions while still guaranteeing the e2e transmission latency bound. Then, we introduce our link to partition assignment policy that satisfies the relaxed conditions.

Consider two links of the same type (*e.g.*, uplink) but in different layers. The two may or may not be on the same routing path from a sensor node to the gateway for a packet. Links $e_{11,6}$ and $e_{5,1}$ in Fig. 3a belong to the latter as they are on the routing paths of different packets. According to Observation 1, the cells assigned to the links of a same packet should be assigned in the slotframe following the packet's routing sequence, but this is not required for the cell sequence for different packets. Thus, there is no need to impose any constraint on the cell sequence for the links on the routing paths of different packets. Based on this observation, we relax Condition 2 and Condition 3 as follows.

Condition 4. *For any two cells allocated to two different uplinks, if the two uplinks are on a same routing path to the gateway for a packet, the cell allocated to the uplink in a higher layer should be scheduled before the cell allocated to the uplink in a lower layer.*

Condition 5. *For any two cells allocated to two different downlinks, if the two downlinks are on a same path from the gateway for a packet, the cell allocated to the downlink in a lower layer should be scheduled before the cell allocated to the downlink in a higher layer.*

To develop a link assignment policy satisfying the above two conditions, we make use of the "branch" concept in the tree structure, and specifically refer to the collection of all nodes/edges having a common layer-1 edge as a branch.

Algorithm 1 2D Partition-based Scheduler

```
Divide the slotframe into R rows in the channel dimension according to the network specification;
Create partitions for the 1^st row by applying the baseline partition-based scheduler;
r ← 2;
while r ≤ R do
    P(U, L, r) starts from the beginning of the slotframe and P(U, 1, r) ends at the left boundary of P(U, 1, r − 1);
    P(D, 1, r) starts from the right boundary of P(D, 1, r − 1) and P(D, L, r) ends at the end of the slotframe;
    The relative sizes of P(U/D, l, r)(1 ≤ l ≤ L) are the same as those of P(U/D, l, r − 1)(1 ≤ l ≤ L);
    r ← r + 1;
end while
while unassigned branches exist do
    Select the branch B_m with the maximum number of links;
    if P(U/D, 1, 1) is empty then
        Assign e(U/D, 1) in B_m to P(U/D, 1, 1);
    else if current row r is unfilled and has enough capacity then
        Assign e(U/D, 1) in B_m to P(U/D, 1, r);
    else
        Assign e(U/D, 1) in B_m to P(U/D, 1, r_s) where r_s is the unassigned row with the smallest index;
    end if
    Assign all the links in the higher layers (i.e., l > 1) in B_m into the corresponding partitions in the same row as e(U/D, 1);
end while
```

Observe that by our definition, the links not belonging to the same branch will never have shared routing paths, and thus they do not need to satisfy Condition 4&5. The key idea of our link assignment policy is to assign all links in a same branch to a same row, thus allow links in different branches to share slots if needed. The detail of the policy is given below.

Link to Partition Assignment. (1) Select the branch with the maximum number of links among the unassigned branches. (2) Assign the layer-1 link of this branch to the corresponding partition in row-1 if this is the first link assignment, or in the current unfilled row if the row has enough capacity, or in the unassigned row with the smallest index. (2) Assign all the links in the higher layers (*i.e.* $l > 1$) in this branch to the corresponding partitions in the same row as the layer-1 link. (3) Repeat (1) and (2) until no more unassigned branches.

The last part of our 2D partition-based scheduler is allocating cells to links. This step can simply use the optimal cell allocation policy, OCAP. The four parts collectively ensure that any generated schedule is a compliant one. Hence, the 2D partition-based scheduler can guarantee that the e2e transmission latency is bounded within one slotframe length. Alg. 1 summarizes the 2D partition-based scheduler.

V. Dynamic Partition Adjustment

The 2D partition-based approach enables 6TiSCH to accommodate more nodes and tasks in the network, and thus improve the system scalability. However, when the network size grows or the network topology changes, it is still possible that the cell allocation requests from some new or existing links cannot be fulfilled in their corresponding partitions. This will cause

the e2e transmission latency of some packets to exceed the slotframe length and degrade the overall network performance. In these cases, dynamic partition adjustment is desired to adapt the schedule to make it complaint again.

The communication schedule of a 6TiSCH network may need to be changed in three different scenarios depending on if the cell allocation requests from packets are updated.

Scenario 1: The cell allocation request decreases. This case happens when some existing node leaves the network or the data rate of some communication task decreases.

Scenario 2: The cell allocation request increases. This case happens when some new node joins the network or the data rate of some communication task increases.

Scenario 3: The cell allocation request does not change but some nodes change their parents.

In the first scenario, the partition-based scheduler can readily release the corresponding cells in the slotframe. Handling the network dynamics in Scenario 2 and 3 is more complicated since both scenarios require on-line cell and/or partition adjustments in the current schedule. In the following, we first describe our solution for handling Scenario 2 by assuming that a cell allocation request from an uplink $e(U,l)$ is to be satisfied. If $e(U,l)$'s corresponding partition is not fully used, the request is simply fulfilled by one of the cells in the partition. On the other hand, if $e(U,l)$'s corresponding partition, $P_{U,l,r}$, is full, we perform partition boundary adjustment to enlarge $P_{U,l,r}$ whenever possible. This dynamic partition boundary adjustment approach needs to decide when and how to adjust, and we discuss the details below.

Regarding "how", the partition boundary adjustment is realized in two ways depending on whether the entire neighbor partition needs to be shifted, or only the boundary of the neighbor partition needs to be changed. For the former case, the partition adjustment information is broadcast in a Beacon packet to all the nodes within that partition. Upon receiving such information, those nodes update their slot offsets in the schedule. On the other hand, the latter case indicates that idle slot(s) exist in the neighbor partition which is able to accommodate $e(U,l)$ (*e.g.*, the leftmost slot in $P_{U,2}$ in Fig. 6). Since the partition boundary information is only known by the scheduler, we can directly allocate $e(U,l)$ into the idle cell in the neighbor partition.

Regarding when to perform dynamic partition adjustment, we do not perform the adjustment immediately when new nodes join in the network or additional cell request exists for transmitting packets. Instead, in order to keep the run-time overhead under control, we perform partition adjustment either periodically or based on certain loading condition. For this "delayed" adjustment to work, we need to temporarily allocate the new links or new requests somewhere before the next partition adjustment. We make use of the unassigned partition, *i.e.*, the cells do not belong to any partition $P_{U/D,l,r}$ (*e.g.*, the grey blocks in Fig. 7) for this purpose. That is, we assign the new links to the unassigned partition. This assignment may cause unbounded transmission latency for packets with links assigned in the unassigned partitions. While, if we choose a random cell in another partition for such temporary cell allocations, that partition's capacity would be affected and may not be able to serve future cell allocation requests, even causing a cascading effect.

The partition adjustment to respond to Scenario 3 can be easily achieved based on the above operations for Scenario 1 and 2. Specifically, we first move the cells allocated to the links impacted by the topology change to the temporary partition. Then the situation becomes the same as Scenario 2 and we can follow the aforementioned process to make the adjustment.

VI. Simulation Studies

We have performed extensive experiments to evaluate the performance of the APaS framework. In this section, we present our key findings in the simulation studies. We further implemented APaS on a 122-node 6TiSCH network testbed and summarize our performance evaluation in Section VII.

A. Simulation Setup

In the simulation studies, we compare APaS with the baseline randomized scheduler (RS), and two state-of-the-art low latency scheduling functions designed for 6TiSCH networks, LLSF [16] and LDSF [17]. RS simply allocates a random cell for the requesting link. LLSF takes the following two steps for cell allocation: (i) for each 1-hop link, it allocates a random cell; (ii) for each multi-hop link, it allocates the next available cell after the cell allocated for its previous link on the routing path. LDSF is a block-based scheduling framework, and it divides each slotframe into small blocks which repeat over time. Similar to LLSF, links in LDSF are assigned to consecutive blocks to minimize the e2e latency.

Our simulation studies aim to evaluate the e2e latency of flows with different hop counts, and exam whether their transmission latency bounds can be guaranteed. We use the following two performance metrics in our studies.

E2e Latency (EL): EL is defined as the e2e MAC layer latency of a packet. To better control the hop count of a packet, we set its sensor and actuator to be a same device.

Success Ratio (SR): SR is defined as the fraction of packets whose e2e transmission latencies are within one slotframe length over all the packets transmitted in the network.

We randomly generate 200 network topologies. The number of nodes in the network is selected from the uniform distribution over $\{20, 40, 60, \ldots, 160\}$. 25 random topologies are generated for each selected network size. The network topology is formed in the same way as in the 6TiSCH network initialization phase. Specifically, we first distribute the gateway and nodes on a 25×25 grid randomly and let the gateway broadcast Beacons. When nodes within the communication range of the gateway (set to 5 unit distance) receive the Beacons, they choose the gateway as their parents and broadcast Beacons to allow other nodes to join the network. This process repeats until all nodes join the network and each node originates a task. The slotframe lengths of all the networks are set to 127 slots and each slot length is 10 milliseconds. We

enable all the 16 channels in 6TiSCH and cell reuse is only allowed for broadcasting Beacons to save slots.

B. Simulation Results

We conduct two sets of experiments in our studies. In the first set of experiments, we compare the performance of APaS, RS, LLSF and LDSF in static network settings. Fig. 8 compares the average EL and average SR among all the schedulers by varying the network size from 20 to 160. Each point in the figure represents the average value of 25 trials with different topologies. We observe from the results that RS suffers a high average EL especially when the network size is large. On the other hand, the average ELs of APaS, LDSF and LLSF are all within 1 slotframe length. Although LLSF achieves the lowest average EL, its SR drops to 80% when the network size increases to 160. LDSF demonstrates a similar trend with LLSF on both EL and SR since it also allocates a random cell for each 1-hop link and the available cells in the subsequent blocks for links corresponding to the following hops. Due to the block-based design, in LDSF each link can only be allocated to its corresponding blocks and thus LDSF suffers a larger EL compared to LLSF. By contrast, APaS can always achieve 100% SR and guarantee that its ELs are bounded by one slotframe length.

In the second set of experiments, we compare the performance of APaS, LLSF and LDSF on handling dynamic network topology changes. We randomly create a 100-node network and generate 50 interfering events in the network at randomly selected times and locations. The duration of each event is set to be small enough so that no concurrent events will appear in the network. During an interfering event, each node within the interference range (set to 2.5 unit distance) of the interferer will choose another neighbor node located outside the interference range as the new parent for seeking better connectivity. At the bottom of Fig. 9, we show the number of nodes that change their parents during each interfering event. The top two subfigures in Fig. 9 illustrate the dynamic changes of EL and SR in responding to the 50 interfering events under APaS, LLSF and LDSF, respectively. It can be observed that both EL and SR of LLSF and LDSF degrade significantly when the network topology changes. However, APaS can maintain a stable EL and guarantee a 100% SR during the simulation.

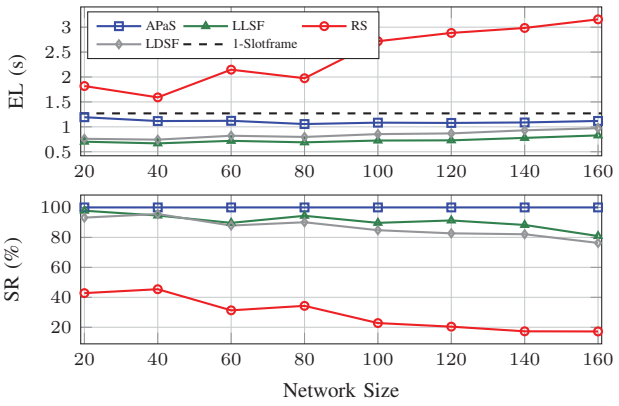


Fig. 8. Comparison of APaS, RS, LLSF and LDSF in static network settings.

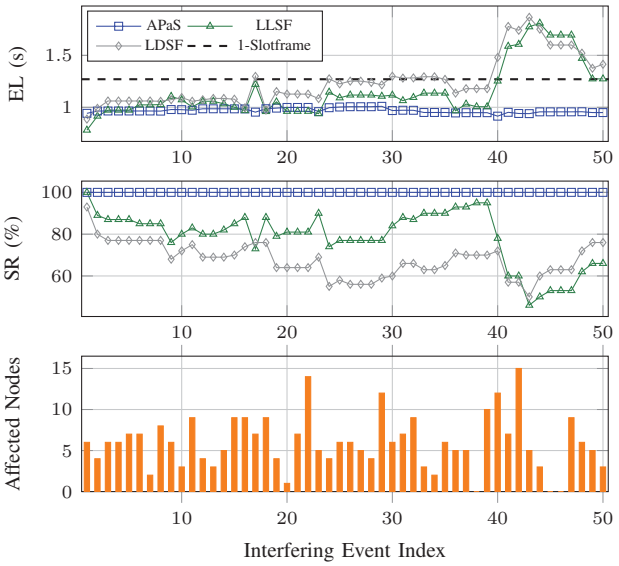


Fig. 9. Comparison of APaS, LLSF and LDSF with dynamic topologies. If the number of affected nodes is 0, it means no nodes are in the interference range of the randomly generated interferer, and no topology change happens.

VII. Testbed Implementation and Evaluation

A. Testbed Implementation and Deployment

We implemented the 6TiSCH stack and the gateway software on COTS hardware and established a 122-node full-blown 6TiSCH network testbed. The 6TiSCH stack is implemented on TI CC2650 SensorTag device [18] (see Fig. 10(a)). We use TI-RTOS as the real-time operating system to run multiple tasks on the devices. For the gateway, we use a Beagle Bone Black (BBB) embedded Linux system to serve the boarder router and a SensorTag device to serve the Access Point (AP) (see Fig. 10(a)). We mount the gateway and the devices at designated locations in our testing environment (see Fig. 10(b)). The SensorTag device provides multiple on-board sensors. These sensor readings are collected through a CoAP task and updated through the gateway to the cloud-based network management system of our testbed (see Fig. 10(c)).

Similar to the simulation studies, we set the slotframe length to 127 slots in the experiments, and each slot is 10 milliseconds. We enable all the 16 channels in 802.15.4e, and cell reuse is only allowed for Beacons to save slots.

B. Performance Evaluation

We performed extensive experiments on the testbed to compare the performance among APaS, LLSF and RS, under both static and dynamic network settings.

Uplink Latency (UL): After the sensor data is subscribed from the gateway, each device generates a CoAP packet every 10 seconds. We record the packets' MAC layer transmission time at the device and the MAC layer reception time at the gateway to measure the uplink latency.

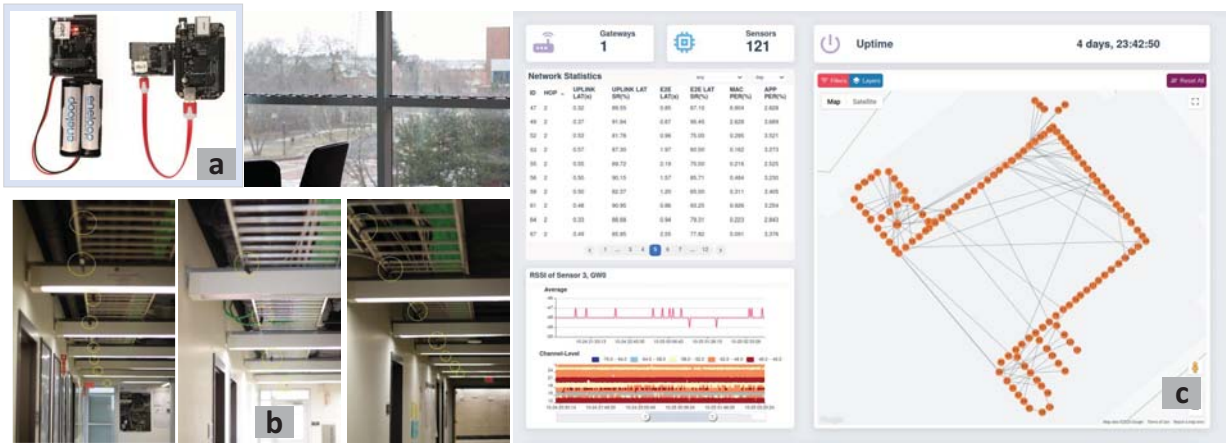
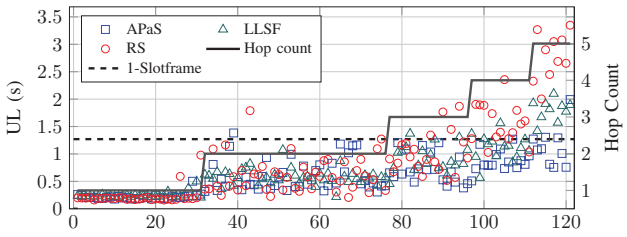


Fig. 10. (a) Hardware platforms for the device (SensorTag) and gateway (BBB+SensorTag); (b) the testing environment where 122 devices and one gateway are mounted; (c) the cloud-based network management system for sensor data collection, topology visualization and network health monitoring.

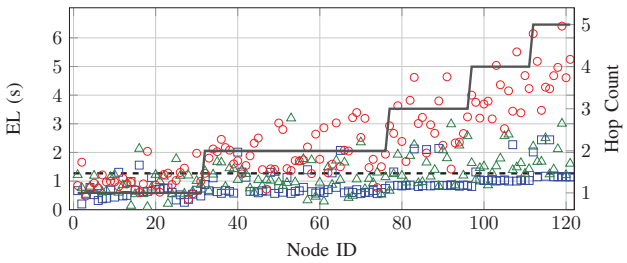
TABLE I
AVERAGE UPLINK AND E2E LATENCY COMPARISON BY LAYERS.

Layer	Average UL (s)			Average EL (s)		
	APaS	LLSF	RS	APaS	LLSF	RS
1	0.224	0.246	0.221	0.629	0.868	1.025
2	0.609	0.585	0.614	0.799	1.141	1.943
3	0.742	0.917	1.005	1.058	1.539	2.909
4	0.881	1.106	1.556	1.110	1.562	3.948
5	1.070	1.796	2.777	1.496	1.865	4.922
Overall	0.604	0.717	0.873	0.894	1.248	2.362
SR	85.7%	80.5%	78.1%	87.3%	70.3%	39.9%

E2e Latency (EL): Each device sends a CoAP packet to the gateway and then to another device every 30 seconds. Similar to the simulation studies, both source and destination are set to the same device. We record the MAC layer transmission and reception time at the device to measure the e2e latency.



(a) Average uplink latency of each device.



(b) Average end-to-end latency of each device.

Fig. 11. Comparison of uplink latency and e2e latency for all the devices under the three schedulers: APaS, RS and LLSF.

To evaluate the performance of all the methods in a static network, we collect 2-hour data trace when the network is running stably, including 87,120 uplink packets and 29,040 e2e packets. Table I summarizes the average uplink and e2e latency for the devices in each layer of the network, and the overall success ratio of all the packets. Although the transmission and reception of 1-hop packets are in the same time slot and their ideal MAC-to-MAC latency should be zero, as there is queuing delay and packet loss in the testbed, some 1-hop packets may also need one or more slotframes to finish the transmission. Thus, the average uplink latency of layer-1 devices of the three schedulers are all around 0.2 seconds. For higher layer packets, the performance of RS degraded quickly as the layer increases (*e.g.*, layer-5's average uplink and e2e latency of RS could reach 2.777s and 4.922s, respectively). APaS and LLSF are less sensitive to the hop counts. However, the average latency of multi-hop packets under the LLSF scheduler is much higher than that under APaS. This is because it allocates cells as close as possible, and thus there is no reserved space to allocate new cells along the routing path when the traffic changes. When the network size increases, LLSF is more likely to suffer from queuing delay than APaS.

Since more than half of the devices in the testbed are in layer 1 and 2 (hop distribution is shown in Fig. 11), the overall average uplink latency of the three schedulers are close, and the overall average e2e latency of LLSF and RS are near one or two slotframes length. But the difference in SR is significant. For the e2e latency, APaS can guarantee 87.3% packets to be finished within one slotframe, while the success ratios of LLSF and RS are only 70.3% and 39.9%, respectively. Fig. 11 compares the average uplink and e2e latency for every device under the three schedulers. Both the uplink and e2e latency of LLSF and RS are increasing linearly with the hop counts, while the performance of APaS is stable and most of packets can be finished within 1 slotframe.

Note that the ELs of LLSF, RS and APaS all increase in the testbed experiments compared with those in the simulation evaluations (the SRs decrease accordingly). The main reason

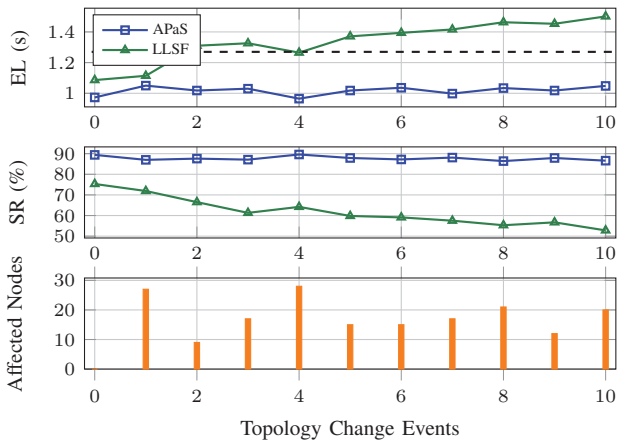


Fig. 12. Comparison of APaS and LLSF with dynamic topology changes.

lies in the system assumptions made in the simulation experiments, *i.e.*, reliable links and sufficient cells without queuing delay. These two assumptions typically do not hold in the testbed and hence lead to increased ELs.To mitigate the impact of packet loss and queuing delay, we allowed retransmission of packets upon link failure and allocated certain number of redundant cells in the slotframe for such retransmissions for all the methods in the testbed implementation. Benefiting from the compliant schedule design of APaS, these redundant cells allocated in the idle slots reserved within each partition can considerably help reduce the e2e latency. However, LLSF does not make idle slot reservation along the packet routing path and redundant cells have to be randomly allocated into idle slots. This is the reason why APaS outperforms LLSF in the testbed experiments.

To evaluate the performance of all the methods in handling network topology changes, we randomly select nodes in the network to restart, which forces the descendants of the selected nodes to rejoin the network through new parents. We generate these topology change events every 40 minutes and collect the performance data for 30 minutes after the network becomes stable for each event. For fair comparison, we use the same random number generator seed and guarantee that the topology change events are the same for all the methods. Fig. 12 shows the average EL and SR of APaS and LLSF on handling 10 consecutive network topology change events. (We omit the results of RS in the figure since the performance of RS is significantly lower than both APaS and LLSF.) The results show that compared with LLSF, APaS maintains stable EL and SR values in the presence of topology changes. By contrast, the EL of LLSF gradually increases and exceeds one slotframe length after the third topology change event. In the meantime, the SR of LLSF decreases from 75% to 52% as well. This is because LLSF does not have the built-in functions for handling network topology changes. After a node (re)joins the network after the topology change, LLSF cannot guarantee that the updated schedule is compliant. By contrast, APaS judiciously reserves idle slots within each partition along the tasks' routing paths, and thus can adapt to topology changes.

VIII. Related Work

In recent years, significant research efforts have been made in designing data link layer scheduling methods for 6TiSCH networks. [12] applies the Minimal Scheduling Function (MSF) as a bootstrap mechanism [19] for 6TiSCH networks based on random cell selection. [20] proposes 6TiSCH operation sublayer protocol (6p) where neighbor devices are able to negotiate the communication schedule locally. The 6TiSCH community standardizes several negotiation-based distributed scheduling functions using 6P transaction, including Scheduling Function Zero (SF0) [21] and Scheduling Function One (SF1) [22]. SF0 enables each node to dynamically adjust the amount of resources between itself and its neighbors based on the current resource allocation. On the other hand, SF1 is an end-to-end resource scheduler with hop-by-hop reservation in a distributed manner. [23] proposes a distributed scheduling policy based on PID control, enabling each node to determine the cell allocation based on bandwidth estimation using a PID controller. However, the above scheduling approaches have the following two drawbacks: 1) transmission conflicts and possible interference among nodes are not considered; 2) end-to-end packet transmission latency can be large and unbounded under a distributed scheduling policy.

Several works in the literature studied the end-to-end latency optimization problem in 6TiSCH networks. Based on SF0, [16] develops an advanced scheduling method called LLSF to minimize the transmission latency by allocating the cells for the packets along their routing paths. However, LLSF does not make any cell reservation and thus suffers high latency in the presence of network dynamics. [24] proposes a localized scheduling algorithm enabling slot reservation for nodes in each layer, but does not support on-line adjustment. [15] proposes the Traffic Aware Scheduling Algorithm (TASA) to reduce transmission latency for 802.15.4e networks. A distributed version of TASA, called DeTAS, is also proposed in [25]. However, both approaches require that the topology under study is static and known a priori. [17] proposes a block-based scheduling framework, called LDSF, to allocate retransmission cells in the slotframe to improve reliability. However, LDSF's low latency and reliability are achieved by adding a large number of redundant cells for each flow, and thus unnecessarily sacrifice the network utilization.

IX. Conclusion and Future Work

In this paper, we propose an adaptive partition-based scheduling framework called APaS for 6TiSCH networks. APaS aims to guarantee the e2e packet transmission latency in multi-channel multi-hop 6TiSCH networks, and handle frequent network topology changes by employing a novel online partition adjustment method. We implement APaS on a 122-node 6TiSCH network and validate its effectiveness through both simulations and testbed experiments. As the future work, we will extend APaS to a distributed version, support tasks with diverse e2e latency requirements, and apply machine learning techniques to handle different kinds of network dynamics and further improve APaS's scalability.

X. Acknowledgement

This research is partially supported by National Science Foundation under awards CCF-2028875 and CCF-2028879. We thank Dr. Tao Gong for assistance with the methodology discussion and experimental testbed setup.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[4] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[5] H. P. Breivold and K. Sandström, "Internet of things for industrial automation–challenges and technical solutions," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*. IEEE, 2015, pp. 532–539.

[6] T. Zhang, T. Gong, C. Gu, H. Ji, S. Han, Q. Deng, and X. S. Hu, "Distributed dynamic packet scheduling for handling disturbances in real-time wireless networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2017, pp. 261–272.

[7] T. Zhang, T. Gong, Z. Yun, S. Han, Q. Deng, and X. S. Hu, "Fd-pas: A fully distributed packet scheduling framework for handling disturbances in real-time wireless networks," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2018, pp. 1–12.

[8] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu, "Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2502–2517, 2018.

[9] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu, "Fully distributed packet scheduling framework for handling disturbances in lossy real-time wireless networks," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 502–518, 2021.

[10] T. Gong, T. Zhang, X. S. Hu, Q. Deng, M. Lemmon, and S. Han, "Reliable dynamic packet scheduling over lossy real-time wireless networks," in *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, 2019.

[11] T. Zhang, T. Gong, X. S. Hu, Q. Deng, and S. Han, "Dynamic resource management in real-time wireless networks," in *Wireless Networks and Industrial IoT*. Springer, 2020, pp. 131–156.

[12] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: deterministic IP-enabled industrial internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, 2014.

[13] D. De Guglielmo, G. Anastasi, and A. Seghetti, "From ieee 802.15. 4 to ieee 802.15. 4e: A step towards the internet of things," in *Advances onto the Internet of Things*, 2014.

[14] P. Djukic and S. Valaee, "Delay aware link scheduling for multi-hop tdma wireless networks," *IEEE/ACM Transactions on networking*, vol. 17, no. 3, pp. 870–883, 2008.

[15] 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 *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC)*. IEEE, 2012, pp. 327–332.

[16] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana, "LLSF: Low latency scheduling function for 6TiSCH networks," in *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2016, pp. 93–95.

[17] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "Ldsf: Low-latency distributed scheduling function for industrial internet of things," *IEEE internet of things journal*, vol. 7, no. 9, pp. 8688–8699, 2020.

[18] "SimpleLink multi-standard CC2650 SensorTag kit." [Online]. Available: http://www.ti.com/tool/TIDC-CC2650STK-SENSORTAG

[19] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) configuration," Internet Requests for Comments, RFC Editor, BCP 210, May 2017.

[20] Q. Wang, X. Vilajosana, and T. Watteyne, "6TiSCH operation sublayer (6top) protocol (6P)," RFC 8480, Nov. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8480.txt

[21] D. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura, "6TiSCH 6top scheduling function zero (SF0)," Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-6top-sf0-05, Jul. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-6top-sf0-05

[22] S. Anamalamudi, B. L. (Remy), M. Zhang, A. R. Sangi, C. E. Perkins, and S. Anand, "Scheduling function one (SF1): hop-by-hop scheduling with RSVP-TE in 6TiSCH networks," Internet Engineering Task Force, Internet-Draft draft-satish-6tisch-6top-sf1-04, Oct. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-satish-6tisch-6top-sf1-04

[23] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne, "Distributed PID-based scheduling for 6TiSCH networks," *IEEE Communications Letters*, 2016.

[24] I. Hosni, F. Théoleyre, and N. Hamdi, "Localized scheduling for end-to-end delay constrained low power lossy networks with 6TiSCH," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016, pp. 507–512.

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