Autonomous Traffic-Aware Scheduling for Industrial Wireless Sensor-Actuator Networks

XIA CHENG and MO SHA*, Florida International University, USA

Recent years have witnessed rapid adoption of low-power Wireless Sensor-Actuator Networks (WSANs) in process industries. To meet the critical demand for reliable and real-time communication in harsh industrial environments, the industrial WSAN standards make a set of specific design choices, such as employing the Time Slotted Channel Hopping (TSCH) technique. Such design choices distinguish industrial WSANs from traditional Wireless Sensor Networks (WSNs), which were designed for best-effort services. Recently, there has been increasing interest in developing new methods to enable autonomous transmission scheduling for industrial WSANs that run TSCH and the Routing Protocol for Low-Power and Lossy Networks (RPL). Our study shows that the current approaches fail to consider the traffic loads of different devices when assigning time slots and channels, which significantly compromises network performance when facing high data rates. In this paper, we introduce a novel Autonomous Traffic-Aware transmission scheduling method for industrial WSANs. The device that runs ATRIA can detect its traffic load based on its local routing information and then schedule its transmissions accordingly without the need to exchange information with neighboring devices. Experimental results show ATRIA provides significantly higher end-to-end network reliability and lower end-to-end latency without introducing additional overhead compared with a state-of-the-art baseline.

CCS Concepts: • **Networks** → **Network protocol design**; **Network experimentation**.

Additional Key Words and Phrases: Industrial Wireless Sensor-Actuator Networks, IEEE 802.15.4, Transmission Scheduling, TSCH, RPL

ACM Reference Format:
Xia Cheng and Mo Sha. 2022. Autonomous Traffic-Aware Scheduling for Industrial Wireless Sensor-Actuator Networks. *ACM Trans. Sensor Netw.* X, X, Article X (January 2022), 25 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Industrial Internet of Things (IoT), which underlies the Fourth Industrial Revolution (or Industry 4.0) [27], promises one of the largest potential economic effects of IoT – up to $47 trillion in added value globally by 2025, according to the McKinsey report on future disruptive technologies [37]. Industrial networks, the underlying support of industrial IoT, typically connect sensors, actuators, and controllers in industrial facilities, such as manufacturing plants, steel mills, oil refineries, and infrastructures that implement complex processes. Industrial applications pose unique challenges to networking because of their critical demand for *real-time* and *reliable* communication in harsh industrial environments. Failure to achieve such performance can lead to production inefficiency, safety threats, and financial loss. These demands have been traditionally met by specifically chosen

*Corresponding author

Part of this article was published in Proceedings of the ICNP [5].
Authors' address: Xia Cheng; Mo Sha, Florida International University, 11200 SW 8th St, Miami, FL, 33199, USA, {xcheng, msha}@fiu.edu.

1550-4859/2022/1-ARTX $15.00
https://doi.org/10.1145/1122445.1122456

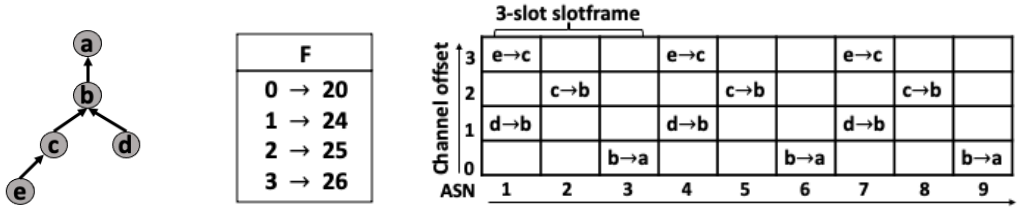
wired solutions, such as HART [20]. However, wired networks are often costly to deploy and maintain in industrial environments and difficult to reconfigure to accommodate new requirements.

IEEE 802.15.4-based Wireless Sensor-Actuator Networks (WSANs) appeal to industrial network designers because they do not require wired infrastructures and can be manufactured inexpensively. Battery-powered wireless modules easily and inexpensively retrofit existing sensors, actuators, and controllers in industrial facilities without the need to run cables for communication and power. To meet the stringent real-time and reliability requirements, the industrial WSAN standards, such as WirelessHART [53], ISA100 [23], WIA-FA [21], and 6TiSCH [22], make a set of specific design choices, such as employing the Time Slotted Channel Hopping (TSCH) technique. Such design choices distinguish industrial WSANs from traditional Wireless Sensor Networks (WSNs), which were designed for best-effort services [35]. A large number of WSANs that implement those standards have been deployed all over the world. For instance, Emerson Process Management, a leading WirelessHART network supplier, has deployed more than 54,835 WirelessHART networks globally and gathered 19.7 billion operating hours of experience [15]. A decade of real-world deployments has demonstrated the feasibility of employing WSANs to achieve reliable low-power wireless communication in industrial facilities.

Recently, WSANs that run TSCH and the Routing Protocol for Low-Power and Lossy Networks (RPL) [48] have been deployed for various applications [12, 41]. Meanwhile, there has been increasing interest in developing new methods, which enable autonomous transmission scheduling for industrial WSANs. For instance, Duquennoy et al. introduced Orchestra [11], which allows each network device to generate its transmission schedule based on its local routing information, and Kim et al. developed ALICE [29], which overcomes Orchestra's limitations and enables the use of all available physical channels in each cell[1] by replacing Orchestra's node-based scheduling with link-based scheduling. To understand the performance of those autonomous transmission scheduling methods, we have performed a series of experimental studies on the FIT IoT-LAB testbed [2]. Our studies show that the current approaches fail to consider the traffic loads of different devices when assigning cells, which significantly compromises network performance when facing high data rates. Therefore, the autonomous scheduling solutions must calculate the traffic loads and assign cells without introducing additional communication. To address such challenges, we develop *ATRIA*, a novel *A*utonomous *Tra*ffi*c*-*A*ware transmission scheduling method for industrial WSANs. The device that runs ATRIA can schedule its transmissions to meet its traffic demand without exchanging information with its neighboring devices. Specifically, each device in the network can detect its traffic load based on its local routing information, select the best-suited slotframe[2] length based on the performance requirements specified by the application, then schedule one or more cells based on its specific traffic load, and adapt the schedule when the traffic demand changes. We have implemented ATRIA under Contiki [10] and evaluated its performance using a network that consists of 50 devices on the FIT IoT-LAB testbed. Experimental results show that ATRIA provides higher end-to-end network reliability and lower end-to-end latency without introducing additional overhead compared with a state-of-the-art baseline.

The remainder of the paper is organized as follows. Section 2 introduces the background of TSCH, RPL, and ALICE. Section 3 presents our experimental study. Section 4 introduces our design of ATRIA. Section 5 presents our experimental evaluation. Section 6 reviews the related work. Section 7 concludes the paper.

[1] A cell denotes the combination of a time slot and a physical channel.
[2] A slotframe consists of a group of successive time slots, which repeats over time.



(a) Network topology. (b) F table. (c) Example transmission schedule.

Fig. 1. Example transmission schedule for a network of five devices, which run TSCH and operate on four channels. One slotframe consists of three slots.

2 BACKGROUND

In this section, we introduce the background of TSCH, RPL, and ALICE.

2.1 TSCH

TSCH was amended into the IEEE 802.15.4e standard [1] in 2012 as a mode to support industrial and embedded applications with stringent performance requirements. TSCH combines time-slotted medium access, multi-channel communication, and channel hopping to provide time-deterministic packet deliveries and combat narrow-band interference and multi-path fading. In a network that runs TSCH, time is divided into slices of fixed length (time slots) that are grouped in a slotframe. All devices are time synchronized using the beacons flooded across the network (e.g., Keep-Alive messages in WirelessHART) and share the notion of a slotframe that repeats over time. Each time slot is long enough to deliver a packet and an acknowledgment between a pair of devices. In each time slot, a set of physical channels can be used, resulting in a matrix-like combination as shown in Figure 1(c). Each cell in Figure 1(c) is identified by its time slot offset and channel offset coordinates. The time slot offset of a cell indicates its position in a slotframe and the channel offset is an index, which maps to one of available physical channels. For a link that is defined as the pairwise assignment of directional communication between two devices, channel hopping is achieved by sending successive packets on different channels in different time slots. A pair of devices identify their communicating channel by computing the following function:

$$channel = F[(ASN + ChannelOffset)\%L_C] \tag{1}$$

where ASN is the absolute slot number, defined as the total number of slots elapsed since the network started, and "%" is the modulo operator. F is the lookup table that maps the channel offsets to their corresponding channels, and L_C is the length of a sequence of available physical channels. In a conventional network that runs TSCH, each device learns the current ASN and the channels used in the network from its neighbors upon joining the network and then uses such information to compute the channel used in each cell.

Figure 1(b) shows an example with four pairs of channel offsets (0, 1, 2, and 3) and channels (channel 20, channel 24, channel 25, and channel 26). Figure 1(c) shows an example transmission schedule, which allows device a in Figure 1(a) to collect data from the rest of the devices every three time slots. For example, device d and e are scheduled to use channel 24 (channel offset 1) and 26 (channel offset 3) to transmit a packet in the first time slot, respectively, while device c and b are scheduled to forward the data in the second and third time slots. The transmission schedule is generated by the scheduling algorithm, which runs on top of TSCH.

2.2 RPL

RPL was developed to support IPv6 and provide resource-constrained devices with multi-hop routing. To address the low-power constraint, RPL constructs a Destination-Oriented Directed Acyclic Graph (DODAG) anchored at a root, typically a border router to external networks. In a DODAG, a device computes its RANK (the logical distance to the root) according to its Objective Function (OF). Minimum Rank with Hysteresis Objective Function (MRHOF) [16] is one of the commonly used OFs, which adopts the Expected Transmission Count (ETX) metric [50] to compute RANK. The routing information including RANK is exchanged by broadcasting DODAG Information Object (DIO) messages. After receiving DIO messages from neighbors, a device can update its RANK and set or change its preferred parent device by sending a Destination Advertisement Object (DAO) message to its selected parent to reduce the logical distance to the router. RPL provides Destination Advertisement Object Acknowledgment (DAO-ACK) as an optional function to enable a device to resend a DAO message to its parent if it does not receive a DAO-ACK from its parent in case of transmission failures. By broadcasting DODAG Information Solicitation (DIS) messages, a device requests routing information from its neighbors. By exchanging DIO and DAO messages, each device sets up its downward and upward routes in the DODAG. RPL supports two modes of operation: storing and non-storing. In the storing mode, devices maintain routing tables for routes locally in a distributed fashion. In the non-storing mode, devices do not maintain the routing states locally.

2.3 ALICE

ALICE schedules transmissions for the networks that run TSCH and RPL and defines three types of slotframes to deliver time synchronization, routing, and application traffic [29]. Enhanced Beacons (EBs) are broadcast by all devices in the time synchronization slotframes and RPL messages are scheduled in the routing slotframes. The unicast upward and downward application traffic uses the application slotframes. When a device is scheduled for multiple types of traffic in a time slot, the device chooses a packet for transmission in this order: time synchronization, routing, and application. The device that runs ALICE can schedule its transmissions autonomously based on its local routing information. Specifically, ALICE assigns one cell for each directional link, uses all available channels, and reschedules transmissions in every application slotframe. Under ALICE, the slot offset $T^k_{m,n}$ of the cell assigned to link $m \rightarrow n$ (from device m to device n) in the k-th unicast slotframe ($k = ASN/L_S$) is calculated as

$$T^k_{m,n} = mod(Hash(\alpha ID(m) + ID(n) + k), L_S) \tag{2}$$

where $Hash(x)$ is a HASH function used to randomize the input value to reduce the conflict between different directional links [51], the coefficient α is used to differentiate traffic directions, e.g., link $m \rightarrow n$ and link $n \rightarrow m$, k is used to provide different input values in different slotframes to avoid the same conflict from happening repeatedly in successive slotframes, and L_S denotes the length of the unicast slotframe. Similarly, the channel offset $C^k_{m,n}$ of this cell is calculated as

$$C^k_{m,n} = mod(Hash(\alpha ID(m) + ID(n) + k), L_C - 1) + 1 \tag{3}$$

where $L_C - 1$ is used because ALICE reserves a physical channel with offset 0 for the time synchronization slotframe and the routing slotframe. To allocate a unique cell for each directional link in a network that consists of N devices and operates on M physical channels, ALICE suggests that the slotframe length should be larger than $(2N - 2)/(M - 1)$.

Fig. 2. Testbed used for our studies. All 50 devices are deployed on the same floor in an office building. The black star indicates the device that serves as the border router (Root) and the read circles denote end devices.

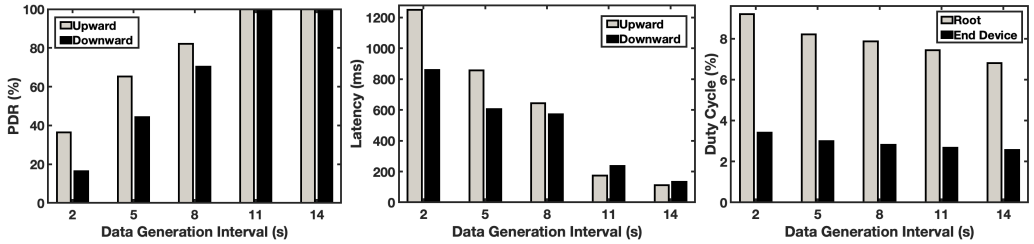
3 EXPERIMENTAL STUDY

We have performed a series of experimental studies to understand the performance of ALICE when the network faces different data rates. We select 50 M3 devices [36] from the FIT IoT-LAB testbed [2] to form a mesh network and run the ALICE implementation provided by Kim et al. [28]. Figure 2 plots the device deployment for our studies. All devices operate on four channels (by default in ALICE) and send packets using the transmission power of -17dBm. We set the routing slotframe length and the time synchronization slotframe length to 19$slots$ and 397$slots$, respectively. Each time slot lasts 10ms.

3.1 ALICE's Performance

We first examine the network performance when we increase the data generation interval of each device from 2s to 14s for both upward traffic and downward traffic, typical data rates for industrial applications [30–32]. We set the length of the unicast slotframe to 43$slots$ (by default in ALICE). Figure 3 plots the end-to-end Packet Delivery Ratio (PDR), the end-to-end latency, and the radio duty cycle under different traffic loads. As Figure 3(a) shows, the PDRs of both upward and downward traffic are 100% when the data generation interval is 11s or 14s. The PDRs drop to 82.2% (upward) and 70.3% (downward), when the data generation interval decreases to 8s. The PDRs further decrease to 36.4% (upward) and 16.4% (downward) when a packet is generated every 2s. These results show that ALICE performs well at low data rates but cannot deliver all packets at high data rates. As Figure 3(b) shows, the end-to-end latency increases significantly when the data generation interval decreases. For example, the latency of the upward traffic increases from 112ms to 1,251ms when the data interval decreases from 14s to 2s. The duty cycle of the Root and the end devices also increases when the traffic load increases, as Figure 3(c) shows.

We also investigate the causes of low reliability and long latency when the network has to deliver data at high rates. Figure 4 plots the Cumulative Distribution Function (CDF) of the cell utilization for all devices when the slotframe length is 43$slots$ and the data generation interval is 2s. As Figure 4 shows, 62.0% of devices use only 21.5% of the cells that are scheduled for upward traffic, while 20.0% of devices have used 86% or more of their allocated cells. Similarly, 26.3% of



(a) PDR. (b) Latency. (c) Duty Cycle.

Fig. 3. Performance when the network has different traffic loads.

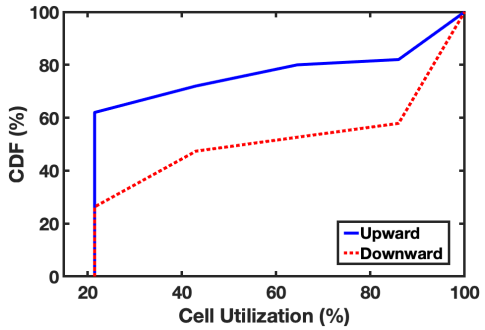


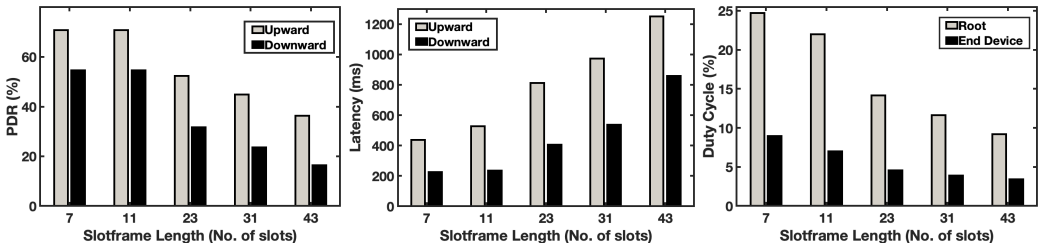
Fig. 4. Cell utilization for upward and downward traffic.

devices use only 21.5% of the cells that are scheduled for downward traffic, while 42.1% of devices have used 86.0% or more of their allocated cells. These results clearly show that many devices with heavy traffic do not have enough cells for packet transmission and retransmission, while the rest have many unused ones. The reason behind that is ALICE fails to consider the traffic demand of each device when scheduling transmissions and assigns a single cell for every directional link in each unicast slotframe. Some devices with high traffic demand do not have enough cells to transmit their data, which results in packet losses and high latency, while some devices with light traffic waste many unused ones.

Observation 1*: Scheduling transmissions without considering the traffic demand of each device leads to poor network performance when the devices generate data at high rates.*

3.2 Impact of Slotframe Length

The slotframe length is a configurable parameter in ALICE. To study the impact of the slotframe length on network performance, we vary the unicast slotframe length and repeat the experiments five times. The data generation interval of all devices is set to 2*s* for both upward traffic and downward traffic in all experiments. Figure 5 plots the end-to-end PDR, the end-to-end latency, and the radio duty cycle when the slotframe lengths are 7*slots*, 11*slots*, 23*slots*, 31*slots*, and 43*slots* (the five default values in ALICE), respectively. As Figure 5(a) shows, the PDRs increase from 36.4% (upward) and 16.4% (downward) to 70.8% (upward) and 54.6% (downward), when the slotframe length decreases from 43*slots* to 11*slots*. This is because ALICE provides more cells for each link in a fixed time period when it uses a smaller slotframe. However, the PDRs of both upward and downward traffic do not continue to increase when the slotframe length further decreases. The PDRs are 70.7% (upward) and 54.5% (downward) when the slotframe length is 7*slots*. This is because



(a) PDR. (b) Latency. (c) Duty cycle.

Fig. 5. Performance when the network uses different slotframe lengths.

the contention for available time slots becomes so severe that many cell allocation failures occur when the slotframe is too small. Figure 5(b) and 5(c) plot the end-to-end latency and the radio duty cycle when the network uses different slotframe lengths. The latency of the upward traffic decreases from 1,251ms to 438ms and the duty cycle of the Root increases sharply from 9.19% to 24.69% when the slotframe length decreases from 43$slots$ to 7$slots$. The results show the tradeoffs between latency and energy efficiency. The reliability and latency can be improved to a certain degree by reducing the slotframe length at the cost of increasing energy consumption.

Observation 2*: Using a smaller slotframe can improve network reliability and latency at the cost of increased energy consumption. However, tuning the slotframe length cannot always help the network achieve desirable performance.*

4 OUR DESIGN OF ATRIA

In this section, we first present an overview of ATRIA and then introduce each of its modules in detail.

4.1 Overview

As Figure 6 shows, operating between RPL and TSCH, ATRIA takes the routing information from RPL, the parameters specified by the application, and the number of transmitted packets in each specified time duration as inputs and generates the transmission schedule for TSCH to execute at runtime. ATRIA inherits the basic slotframe designs for time synchronization, routing, and application traffic and the scheduling priority from ALICE. To avoid introducing additional communication overhead, ATRIA adopts the storing mode of RPL to make use of the local routing information. Therefore, each device that runs ATRIA can generate its transmission schedule and allocate cells for each link based on the medium access control (MAC) addresses stored in the local routing table. There is no need to exchange information with neighbors. Our study in Section 3.1 shows that scheduling transmissions without considering the traffic demand of each link leads to poor network performance when the devices generate data at high rates. Therefore, ATRIA is designed to allocate one or more cells to each link based on its specific traffic demand. To achieve this goal, each device that runs ATRIA first detects the traffic demand of each link, selects the best-suited slotframe length, then schedules one or more cells to each directional link, and adapts the schedule at runtime with four modules:

- **Topology Identifier** is responsible for learning the current network topology from the local routing information.

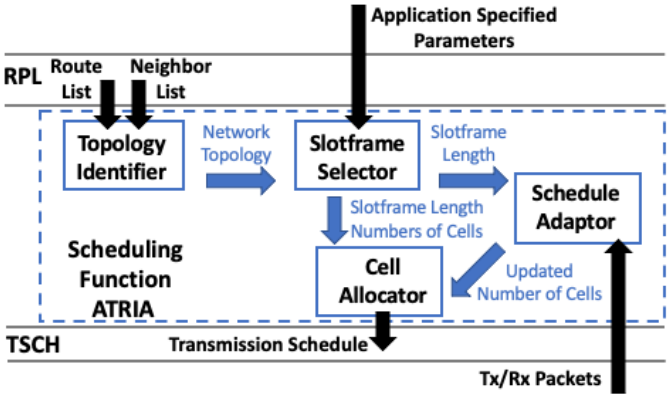


Fig. 6. Overview of ATRIA.

- **Slotframe Selector** computes the best-suited slotframe length based on the network topology learned by Topology Identifier and the application specified parameters including the data generation interval T of each device and the maximum transmission attempts per packet N_R.
- **Cell Allocator** leverages our dual-slotframe design to allocate cells to each directional link according to the slotframe length and the numbers of scheduled cells provided by Slotframe Selector.
- **Schedule Adaptor** adapts to the change of the traffic demand by counting the number of transmitted packets in a specified time duration and adjusting the numbers of scheduled cells accordingly.

We will present our designs of those four modules next.

4.2 Topology Identifier

Topology Identifier runs on each device and identifies the number of descendant nodes it has and the number of descendant nodes belonging to each of its child nodes. Such information is used to select the slotframe length (See Section 4.3) and allocate cells (See Section 4.4).

To collect such information, ATRIA enables the DAO-ACK option in RPL where each device keeps updating its route list that stores the routes to its descendants and the neighbor list that stores its child nodes and their descendants[3]. Topology Identifier detects the descendants of a device by checking its route list and identifies the child nodes of a device and the descendants belonging to each of its child nodes by scanning the neighbor list. Topology Identifier is executed by each device in every slotframe to address the routing list updates resulted from network topology changes.

4.3 Slotframe Selector

Our experimental study in Section 3.2 shows that the selection of the slotframe length significantly affects network performance. As Figure 5 shows, the network that uses a large slotframe suffers poor reliability and large latency. Blindly reducing the slotframe length not only cannot keep improving the reliability but also significantly compromises the energy efficiency. This is because the slot conflict, where a cell is allocated to more than one link, happens frequently when the slotframe length is too small. Slotframe Selector is designed to identify the best-suited slotframe length that allows the network to achieve high reliability with low energy consumption.

[3]The route list is implemented as routelist and the neighbor list is implemented as nbr_routes in Contiki 3.0.

Slotframe Selector runs on each device and takes the topology information provided by Topology Identifier and the application specified parameters (the data generation interval T of each device and the maximum transmission attempts per packet N_R) as inputs. For a network that consists of N end devices and the Root, Slotframe Selector first detects the traffic loads by calculating the number of packets, which are scheduled to be transmitted in a specified time duration. We define the length of such time duration as D and set it to the least common multiple of the data generation intervals used in the network. Please note that all devices in the network derive the same value for D, because they use the same set of data intervals as their input. The number of packets P_j that are scheduled to be transmitted through a directional link j during D is calculated as

$$P_j = \sum_{i=1}^{m+1} \frac{D}{T_i} \tag{4}$$

where T_i denotes the data generation interval of the device i and m denotes the number of the end devices, each of which is the descendant of the sender or the receiver of this link. When m is equal to $N-1$, the directional link is responsible for forwarding the packets between the Root and the rest $N-1$ end devices. Under this extreme topology case, this directional link j is expected to transmit the maximum number of packets P_j^m during D, which is expressed as

$$P_j^m = \sum_{i=1}^{N} \frac{D}{T_i} \tag{5}$$

Meanwhile, the end device that forwards the traffic between the Root and the $N-1$ end devices is responsible for transmitting the maximum number of packets during D. The maximum traffic load of a device P_{max} is

$$P_{max} \simeq 2 * (\sum_{i=1}^{N} \frac{D}{T_i^u} + \sum_{i=1}^{N} \frac{D}{T_i^d}) \tag{6}$$

where T_i^u denotes the interval of the upward data flow from the device i to the Root, and T_i^d denotes the interval of the downward data flow from the Root to the device i.

For a given D, the maximum length of the slotframe is D/S (S denoted as the duration of a time slot). If D/S is no less than P_{max}, the transmission is schedulable by ATRIA. For a given slotframe length, the success rate of allocating cells without introducing any slot conflict depends on the specific cell allocation algorithm. While taking the success rate, D/S, and N_R into account, the following equation can be used to compute the best-suited slotframe length:

$$L_S = R \times D/S \div N_R = \frac{R \cdot D}{N_R \cdot S} \tag{7}$$

where R denotes the success rate of the chosen cell allocation algorithm. We will introduce our cell allocation algorithm in Section 4.4. Because all devices share the same R, D, N_R, and S, they select the same value for L_S, which should be no less than P_{max} to provide enough cells even under the extreme topology case. Then, Slotframe Selector decides the number of cells scheduled for each link according to the number of packets during D. For example, P_j cells are scheduled for the directional link j in each slotframe to deliver P_j packets during D. When the network topology changes, Slotframe Selector only needs to adjust the number of scheduled cells according to the updates of Topology Identifier.

4.4 Cell Allocator

The existing autonomous allocation methods such as ALICE usually assign the slot offsets completely randomly by using HASH functions. Such methods may introduce many slot conflicts.

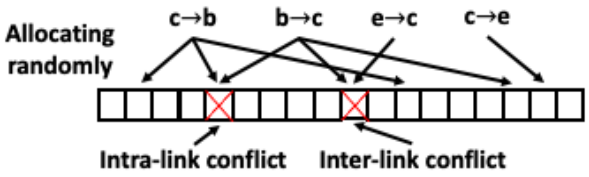


Fig. 7. Example slot conflicts. A slotframe consists of 18 slots. The numbers of cells scheduled for link $c \rightarrow b$, $b \rightarrow c$, $e \rightarrow c$, and $c \rightarrow e$ are three, three, one, and one, respectively.

Figure 7 shows example slot conflicts when using ALICE to allocate one or more cells randomly for each link of device c in the network plotted in Figure 1(a). We assume that L_S is 18. The numbers of cells scheduled for link $c \rightarrow b$, $b \rightarrow c$, $e \rightarrow c$, and $c \rightarrow e$ are three, three, one, and one, respectively. For the second cell scheduled for link $c \rightarrow b$ and the first cell scheduled for link $b \rightarrow c$, we assume that the return values of the HASH operation are 185 and 77, respectively. However, the modulo operation returns the same remainder five for those different inputs when using 18 as the modulus, leading to the conflict for the slot with the offset five. We define such slot conflict between the links that connect the same pair of devices as *intra-link conflict*. Link $b \rightarrow c$ and $e \rightarrow c$ do not connect the same pair of devices and the cells scheduled for them are assigned with the same slot offset 10. We define such slot conflict as *inter-link conflict*. The abovementioned slot conflicts result in an allocation success rate of 75.0%. Cell Allocator performs cell allocation according to the slotframe length selected by Slotframe Selector and the number of cells scheduled for each link provided by Slotframe Selector and Schedule Adaptor. It runs on each device in every slotframe to enhance the success rate while allocating one or more cells to each directional link.

Instead of allocating cells randomly, Cell Allocator first intends to allocate the cells that are scheduled for the two directional links between two devices, e.g., link $a \rightarrow b$ and link $b \rightarrow a$, sequentially in a slotframe. This process is designed to eliminate the intra-link conflict by allocating the cells for the same pair of links one after another. In addition, Cell Allocator reduces the inter-link conflicts by employing a novel dual-slotframe design and performing the following three steps:

(1) **Dividing slotframe**: The slotframe is divided equally into a number of subslotframes. The number ($2P_j^m$) is enough to provide each cell with a unique subslotframe;
(2) **Allocating subslotframes**: The cells for the directional link of upward data flows are distributed among the subslotframes, with similar distances between each other. Similarly, the cells for downward data flows are distributed among the remaining unoccupied subslotframes;
(3) **Allocating cells:** A pair of random functions is used to generate the slot offset and the channel offset of each cell in its subslotframe.

We illustrate this process in Figure 8. We assume that a network is composed of device a and b. According to Slotframe Selector, L_S is 18 and P_j^m is four. Four cells are scheduled for link $b \rightarrow a$ and three cells are scheduled for link $a \rightarrow b$, resulting from the unbalanced traffic loads. As Figure 8 shows, the slotframe is first divided into eight subslotframes. The length of six subslotframes is two slots, while the fourth and eighth subslotframes include three slots because of the remainder (Step 1). Then, seven cells scheduled for link $b \rightarrow a$ and $a \rightarrow b$ are sequentially allocated to the subslotframes except the last one (Step 2). In Step 3, Cell Allocator assigns the slot and channel offsets by revising Eq. 2 and Eq. 3. Specifically, the slot offset $T_{a,b}^{k,i}$ of the i-th cell for the link from device a to device b, in the j-th subslotframe of the k-th slotframe since the network started is calculated as

$$T_{a,b}^{k,i} = mod(Hash(\alpha ID(a) + ID(b) + k \times i), L_{SS}^{j}) \tag{8}$$

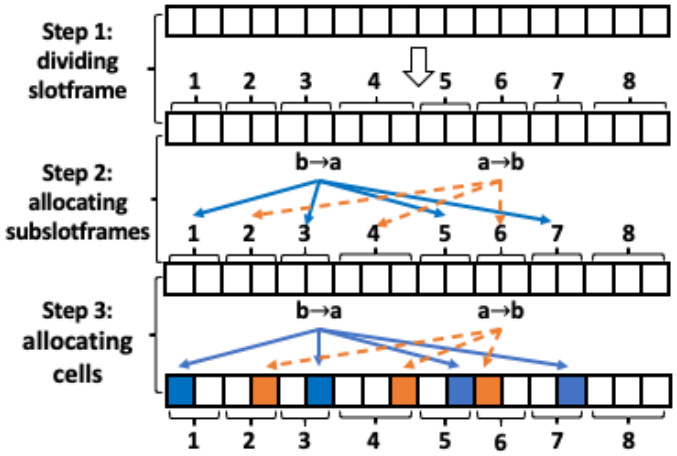


Fig. 8. Example slot allocations. A slotframe that consists of 18 slots is divided into eight subslotframes. Seven cells distribute among eight subslotframes uniformly. Each cell is allocated to a slot in its subslotframe.

where the coefficient α[4] is used to differentiate traffic directions, e.g., link $a \rightarrow b$ vs. link $b \rightarrow a$. The product of k and i is used to differentiate the inputs of the HASH function in different slotframes to prevent the contention for the same slot from happening repeatedly in successive slotframes. L_{SS}^{j} denotes the length of the j-th subslotframe. Similarly, the channel offset $C_{a,b}^{k,i}$ is calculated as

$$C_{a,b}^{k,i} = mod(Hash(\alpha ID(a) + ID(b) + k \times i), L_C - 1) + 1 \tag{9}$$

where L_C is the length of the sequence of physical channels. In the end, Cell Allocator maps the slot offset $T_{a,b}^{k,i}$ in the subslotframe to the slot offset in the unicast slotframe for TSCH operations. Figure 9 plots example cell allocations for the network plotted in Figure 1(a). We assume that L_S is 18 and P_j^m is four. The numbers of cells scheduled for link $b \rightarrow a$, $a \rightarrow b$, $c \rightarrow b$, $b \rightarrow c$, $d \rightarrow b$, $b \rightarrow d$, $e \rightarrow c$, and $c \rightarrow e$ are four, three, three, two, one, one, two, and one, respectively. As Figure 9 shows, the cells scheduled for the same pair of links are allocated uniformly, without any intra-link conflict. Cell Allocator also significantly reduces the inter-link conflicts, only one conflict in slot 16.

We now prove that using our dual-slotframe design can provide a device with more cells without slot conflict compared to the random allocation methods. We start from a case where a given device has a parent and a child node. We assume that x cells are scheduled to the pair of links between the device and its parent in a slotframe that consists of z slots. Meanwhile, y cells are scheduled to the pair of links between the device and its child. Let $E(x, y, z)$ and $R(x, y, z)$ denote the numbers of allocated cells while executing Cell Allocator and the random allocation method.

Proposition 1. *For any $x \in \mathbb{N}^+$ and $y \in \mathbb{N}^+$ such that $x > y$ and $x + y < z$, we have $E(x, y, z) > R(x, y, z)$.*

[4]We set α to 256, the maximum value of the last byte of MAC address.

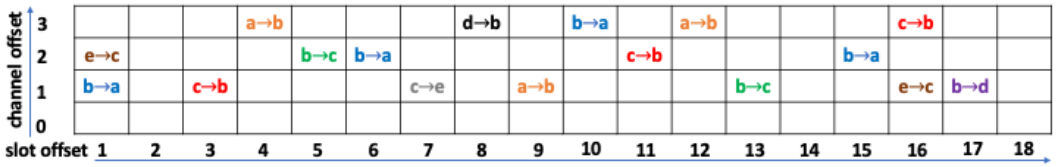


Fig. 9. Example cell allocations to four pairs of bidirectional links in Figure 1(a). A slotframe that consists of 18 slots is divided into eight subslotframes.

Proof. We prove this inequality by constructing the expressions for $E(x, y, z)$ and $R(x, y, z)$ according to each allocation algorithm and comparing them.

$$E(x,y,z) = \overbrace{1+\ldots+1}^{\text{x items}} + \overbrace{\frac{z-x}{z}+\ldots+\frac{z-x}{z}}^{\text{y items}} = x + \frac{(z-x)y}{z}$$

$$R(x,y,z) = 1+\frac{z-1}{z}+\ldots+\left(\frac{z-1}{z}\right)^{x+y-1} = z\left(1-\left(\frac{z-1}{z}\right)^{x+y}\right)$$

To prove $E(x,y,z) - R(x,y,z) > 0$, it suffices to prove the equality: $(E(x,y,z) - R(x,y,z))z^{x+y-1} > 0$. After expanding $(E(x,y,z) - R(x,y,z))z^{x+y-1}$ and merging similar items, we have:

$$\frac{3(x^2+y^2-x-y)z-(x^2+y^2+2xy-3x-3y+2)(x+y)}{6} z^{x+y-3} + \binom{x+y}{4} z^{x+y-4} - \binom{x+y}{5} z^{x+y-5} \ldots + \binom{x+y}{x+y} (-1)^{x+y} z^0$$

Because $x, y \in \mathbb{N}^+$ and $x + y < z$, it is easy to derive that the coefficient of z^{x+y-3} is positive. Therefore, to prove the above inequality, it suffices to prove the following inequality:

$$\binom{x+y}{4} z^{x+y-4} - \binom{x+y}{5} z^{x+y-5} \ldots + \binom{x+y}{x+y} (-1)^{x+y} z^0 > 0$$

We separate the following proof into two cases: (1) $x + y$ is odd and (2) $x + y$ is even.
Case 1: In this polynomial, each positive term is followed by a negative term. After merging similar items in each pair of terms, we observe that the absolute value of the former term is always larger than that of the latter term. The sum of these pair of terms is positive, we have $E(x, y, z) > R(x, y, z)$.
Case 2: In this polynomial, each positive term is followed by a negative term, except the last term, which is positive. It is easy to derive that the sum of these terms is positive according to Case 1. So we have $E(x, y, z) > R(x, y, z)$. □

When a given device has one more child node and q cells are scheduled for the pair of links between the device and its second child, we construct the expressions for $E(x, y, q, z)$ and $R(x, y, q, z)$ and compare them.

$$E(x,y,q,z) = x + \frac{(z-x)y}{z} + \frac{(z^2-zx-zy+xy)q}{z^2}$$

$$R(x,y,q,z) = z\left(1-\left(\frac{z-1}{z}\right)^{x+y+q}\right)$$

After expanding and merging similar items by following the similar process, we have $E(x, y, q, z) > R(x, y, q, z)$. Similarly, under the case where a device has a parent and N child nodes, we construct the expressions for $E(x_1, x_2 \cdots x_{N+1}, z)$ and $R(x_1, x_2 \cdots x_{N+1}, z)$ and compare them to prove that our dual-slotframe design provides more cells without slot conflict. We derive the success rate of allocating cells without introducing slot conflict by comparing $E(x_1, x_2 \cdots x_{N+1}, z)$ to the number

of scheduled cells and use the success rate as R in Eq. 7. After repeating the dual-slotframe design for each device, Cell Allocator provides the network with more cells without slot conflict.

4.5 Schedule Adaptor

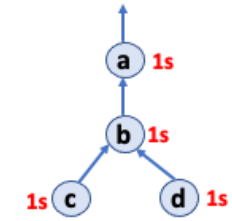
The traffic demand of a device may change at runtime, therefore it is important to adapt the cell assignments at runtime. Schedule Adaptor is designed to run on each device and update the numbers of cells that are scheduled for each link periodically. It takes the number of packets transmitted through each link in a specified time duration provided by the MAC layer and the slotframe length L_S provided by Slotframe Selector as inputs, and updates the number of cells scheduled for each link accordingly. Specifically, Schedule Adaptor performs the following three steps: (1) monitoring traffic loads; (2) detecting traffic changes; (3) adjusting cell assignments.

Algorithm 1: Confirm Traffic Change

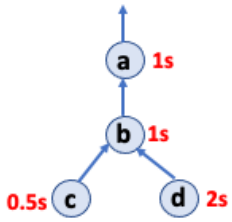
Input : T_k^O, T^T, C^T
Output: T_k^N

```
for i = 1;; i++ do
    for k = 1; k ≤ M; k++ do
        Collect the number of packets transmitted successfully through link k;
    end
    Generate T_k^N of each link;
    if There is no routing topology change then
        for k = 1; k ≤ M; k++ do
            if T_k^N ∉ (T^O − T^T, T^O + T^T) then
                C_k^A++;
                if C_k^A > C^T then
                    Replace T_k^O with T_k^N and output T_k^N;
                end
            end
            else
                Reset C_k^A when it does not change in a time duration;
            end
        end
    end
end
```

In Step 1, Schedule Adaptor monitors the current traffic load of each link in a checking period. We set the length of the checking period equal to D (the least common multiple of the data generation intervals) to detect the traffic change happened in each data interval in time. To make the sender and the receiver of a communicating link share the same number of packets transmitted successfully through the link, the packet acknowledgements are enabled in the MAC layer. Each packet that is transmitted successfully is identified and counted according to its corresponding link by the MAC layer. By reading the parameter that stores the number of transmitted packets, Schedule Adaptor is able to derive the current traffic load of each link. In Step 2, Schedule Adaptor detects traffic load changes. Algorithm 1 presents the details of Step 1 and Step 2. The input of Algorithm 1 consists of three parameters: the original traffic load of each link during D (T_k^O), the traffic variation threshold (T^T), and the counter threshold of traffic changes detected in a series of continuous checking



Link	c → b	d → b	b → a
Traffic Load (packet/second)	1	1	3
Number of Cells	2	2	6

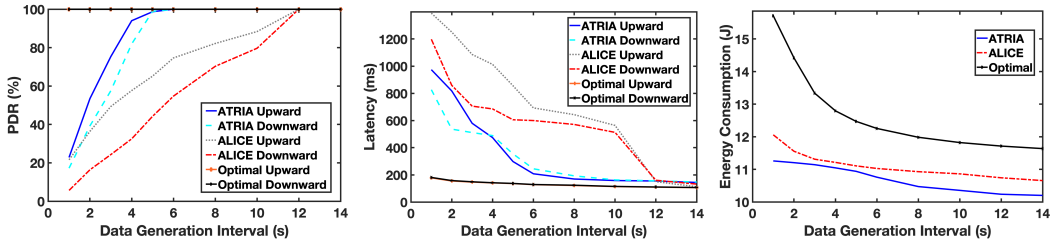


Link	c → b	d → b	b → a
Traffic Load (packet/second)	2	0.5	3.5
Number of Cells	4	1	7

Fig. 10. Example of updating the numbers of scheduled cells.

periods (C^T). Schedule Adaptor is executed in every checking period (line 1). It first collects the number of packets transmitted through each link during this checking period by traversing all M links (line 2 – 4) and generates the latest traffic load T_k^N of each link (line 5). When the network topology changes, the traffic loads usually change sequentially and Slotframe Selector updates the number of scheduled cells for each link synchronously. Therefore, Algorithm 1 checks whether the routing topology changes recently (line 6). If there is no routing change, it then traverses all communicating links (line 7) and compares T_k^N and T_k^O (line 8). When the first attempt of a packet transmission fails in a checking period, the second or third attempt may happen and succeed in the next checking period, leading to the difference between T_k^N and T_k^O. To exclude such cases, only when the difference between T_k^N and T_k^O is larger than the variation threshold T^T (line 8), the counter of continuous traffic changes C_k^A is added (line 9). If the counter C_k^A is larger than the threshold C^T, Algorithm 1 confirms that the traffic load of link k changes continuously and replaces T_k^O with T_k^N (line 10 – 12). If the difference between T_k^N and T_k^O is small and then C_k^A does not change in a time duration, the counter C_k^A is reset (line 14 – 16).

After Algorithm 1 updates the traffic load of each link, Schedule Adaptor continues to perform Step 3. In Step 3, Schedule Adaptor keeps using the slotframe length L_S provided by Slotframe Selector and calculates the number of cells that should be scheduled in such a slotframe according to the latest traffic load. This is because L_S is shared by all network devices and the latest traffic load is only shared by each pair of devices without introducing additional communication among devices. We use an example to illustrate Step 3 (Figure 10). We assume that device a, b, c, and d form the network, where the Root collects data from all other devices. Device a is responsible for forwarding the data packets generated by device b, c, and d to its parent. We also assume that Slotframe Selector sets L_S to 100, while N_R is two and S is $10ms$. At first, all four devices generate one data packet every one second. Therefore, the traffic loads of link $c \rightarrow b$, link $d \rightarrow b$, and link $b \rightarrow a$ are one packet per second, one packet per second, and three packets per second, respectively. Accordingly, the numbers of cells scheduled for such three links are two, two, and six, respectively. After a long time of operation, device c starts to generate two packets every second, while device d changes its data generation interval to two seconds. We assume that Schedule Adaptor detects and confirms the traffic change by performing Step 1 and Step 2. In Step 3, as the traffic loads of link $c \rightarrow b$, link $d \rightarrow b$, and link $b \rightarrow a$ change to two packets per second, 0.5 packets per second, and



(a) End-to-end PDR averaged among 49 data flows.

(b) End-to-end latency averaged among 49 data flows.

(c) Energy consumption averaged among 50 devices.

Fig. 11. Network performance when we vary the data generation interval from 2s to 14s. In each experiment, all devices generate data at the same rate.

3.5 packets per second, respectively, Schedule Adaptor updates the numbers of cells scheduled for these three links from two to four, from two to one, and from six to seven, respectively. At the start of the next slotframe, Cell Allocator allocates such numbers of cells to meet the updated traffic demands of these three links.

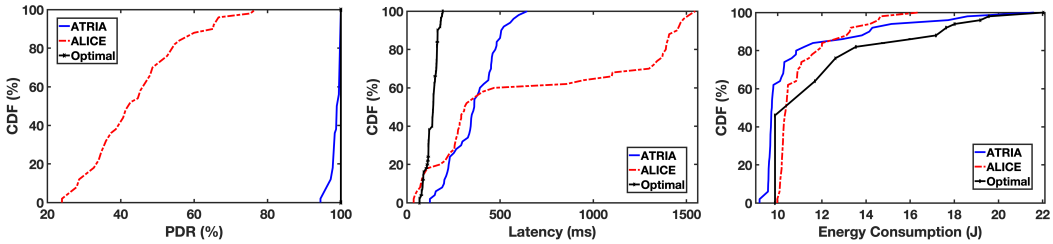
5 EVALUATION

To validate the effectiveness of ATRIA in improving network performance at high data rates, we perform a series of experiments on the FIT IoT-LAB testbed [2]. We first examine the performance of ATRIA when all devices generate data at the same rate. We then evaluate the capability of ATRIA to provide high network reliability when the devices have different data rates. Next, we vary the ratio of upward traffic to downward traffic and study its impact on network performance. Finally, we evaluate the effectiveness of schedule adaptations. We let one device serve as the Root and configure 49 end devices to generate periodic upward traffic. To study the data dissemination performance, we also configure the Root to generate packets periodically (downward traffic). Figure 2 plots the testbed deployment. The network consists of multiple data flows and the routing path of each data flow includes no more than six hops. All routes are generated by the RPL protocol with MRHOF. To compare the performance of ATRIA and ALICE, all devices operate on four channels (physical channel 15, 25, 26, and 20) and send packets using the transmission power of -17dBm (the default value in the ALICE implementation). We also compare their performance against that of the optimal scheduling method (Optimal) with the objective of maximizing the end-to-end reliability. Please note that Optimal is based on backward data analysis and cannot be implemented at runtime. We set the slotframe lengths for routing and time synchronization to 19$slots$ and 397$slots$, respectively. Each time slot lasts 10ms.

5.1 Performance with A Single Data Generation Interval

In this set of experiments, we configure all devices to generate data at the same rate and vary the data interval from 2s to 14s. We vary N_R from 1 to 7. Leveraging Eq. 7, the devices that run ATRIA always select 200$slots$ as the length. We use the default length (43$slots$) for ALICE and 100$slots$ for Optimal.

Figure 11 plots the network performance when we vary the data generation interval. As Figure 11(a) shows, the PDRs of the upward and downward data flows are 100% when the data generation interval is no less than 12s. As expected, all methods perform well when the data is generated at low rates. The PDRs of the upward and downward data flows under ALICE decrease



(a) CDF of end-to-end PDR of data flows. (b) CDF of end-to-end latency of data flows. (c) CDF of energy consumption of devices.

Fig. 12. Network performance when the data generation interval is 5s.

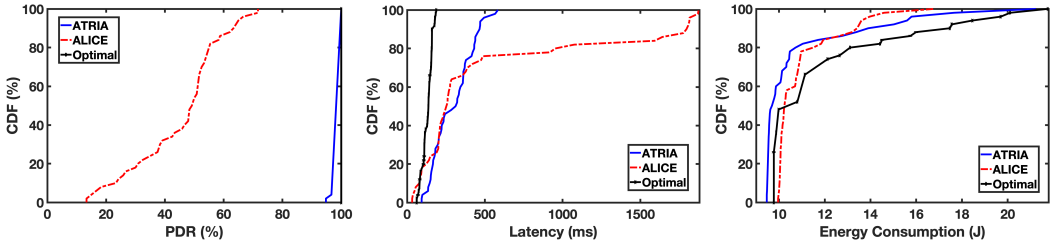
rapidly when the data rate increases. For instance, the PDR of the upward traffic drops from 97.2% to 74.6% and the PDR of the downward traffic drops from 94.9% to 54.9% when the data generation interval decreases from 11s to 6s. As a comparison, both ATRIA and Optimal provide 100% PDR at those rates. When we further reduce the data generation interval to 4s, the PDRs provided by ATRIA are 94.0% (upward) and 82.1% (downward), while the PDRs provided by ALICE are 57.7% (upward) and 32.5% (downward). The PDRs under Optimal are 100%, because it can always allocate enough cells for transmission resulting from the backward data analysis. The results clearly show the effectiveness of ATRIA in enhancing network reliability at high data rates by allocating cells based on the traffic demands.

Figure 11(b) plots the averaged end-to-end latency. All methods provide low latency (around 130ms) when the data generation interval is no less than 12s. As the data rate increases, ATRIA and Optimal outperform ALICE. For example, when the data interval decreases from 11s to 6s, the latency of the upward data flows under ATRIA and Optimal increases slightly from 157ms to 207ms and from 114ms to 130ms, respectively, while the one under ALICE increases sharply from 292ms to 693ms. When the data interval is less than 5s, both ATRIA and ALICE suffer high latency. The results demonstrate the effectiveness of ATRIA in keeping the latency low at high data rates.

Figure 11(c) plots energy consumption[5] averaged among all 50 devices. As Figure 11(c) shows, the averaged energy consumption under ATRIA is consistently the lowest. For instance, when the data interval decreases from 14s to 2s, the energy consumption under ATRIA increases from 10.2J to 11.2J, while the one under ALICE increases from 10.7J to 11.6J. This is because all devices that run ATRIA allocate cells based on their specific traffic loads and wake up accordingly, resulting in lower energy consumption. The devices that run Optimal consume more energy, because Optimal schedules more cells to ensure high reliability.

Figure 12 provides a detailed look at the network performance when the data generation interval is 5s. Figure 12(a) plots the CDF of the end-to-end PDR of each data flow. When the devices run ATRIA, 45.0% of the data flows achieve 100% PDR. The median and minimum PDRs are 98.9% and 94.4%, respectively. As a comparison, the maximum, median, and minimum PDRs under ALICE are 76.7%, 42.0%, and 24.0%, respectively. Figure 12(b) plots the CDF of the end-to-end latency of each data flow. Under ATRIA, the minimum, upper quartile, and maximum latency values are 124ms, 455ms, and 645ms, respectively. As a comparison, when the devices run ALICE, the minimum, upper quartile, and maximum latency values are 37ms, 1372ms, and 1548ms, respectively. The results clearly show that ATRIA can enhance the reliability and reduce the latency by scheduling cells based on the traffic demands. As Figure 12(c) shows, ATRIA and ALICE provide comparable performance

[5]Energy consumption is calculated based on the measured duty cycles and the M3 device datasheet [36].



(a) CDF of end-to-end PDR of data flows. (b) CDF of end-to-end latency of data flows. (c) CDF of energy consumption of devices.

Fig. 13. Network performance when the devices generate data at multiple rates.

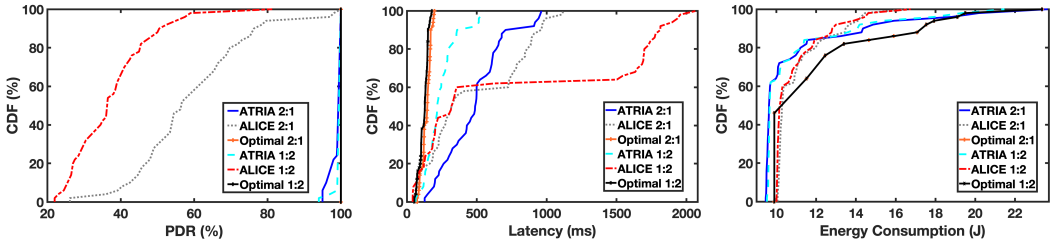
on energy consumption. The energy consumption varies from 9.2J to 21.6J under ATRIA, while it ranges from 10.0J to 16.4J under ALICE. The devices with different traffic demands are scheduled to use different numbers of cells in each slotframe, leading to larger variations on energy consumption among devices under ATRIA. The slight increases in energy consumption variations are in exchange for a proportionally much-larger enhancement in reliability and reduction in latency. As Figure 11 and Figure 12 show, compared to ALICE, ATRIA provides significantly higher reliability and lower latency with comparable energy consumption at high data rates, while it consumes less energy to achieve comparable performance on the network reliability and latency at low data rates.

5.2 Performance with Multiple Data Generation Intervals

In this set of experiments, we evaluate the capability of ATRIA to provide high network reliability when the devices have different data rates, leading to more unbalanced traffic. We configure 15 devices to generate a packet every 3s and let 15 devices generate a packet every 6s. The rest of the devices generate data with an interval of 12s. Figure 13(a) plots the CDF of the end-to-end PDR of each data flow. The performance provided by ATRIA is close to Optimal's. When the devices run ATRIA, the minimum PDR is 94.8%, while the maximum, median, and minimum PDRs under ALICE are 71.7%, 49.2%, and 15.0%, respectively. The results show the benefit of providing more cells to those devices with heavier traffic loads. Figure 13(b) plots the CDF of the end-to-end latency of each data flow. When the devices run ATRIA, the minimum, median, and maximum latency values are 93ms, 311ms, and 584ms, respectively. Under ALICE, the minimum, median, and maximum latency values are 32ms, 253ms, and 1,875ms, respectively. Many devices that generate data with small intervals do not have enough cells to deliver their data, resulting in the long tail under ALICE. As a comparison, those values under Optimal are 62ms, 137ms, and 189ms, respectively. Figure 13(c) plots the CDF of energy consumption. As Figure 13(c) shows, 88.0% of the devices have the lowest energy consumption when they run ATRIA. The minimum, median, and maximum values under ATRIA are 9.5J, 9.7J, and 21.2J, respectively. As a comparison, those values are 10.0J, 10.2J, and 16.8J, respectively, when they run ALICE. The devices consume more energy to allocate more cells to provide higher reliability under Optimal. By observing the results, we can conclude that ATRIA can better handle packet deliveries across the network when the devices generate data at different rates.

5.3 Impact of Traffic Patterns

In this set of experiments, we create unbalanced traffic loads by varying the ratio of the upward traffic to the downward traffic and study its impact on network performance. We first create a network that mainly collects data by setting the data generation intervals for the upward and



(a) CDF of end-to-end PDR of data flows. (b) CDF of end-to-end latency of data flows. (c) CDF of energy consumption of devices.

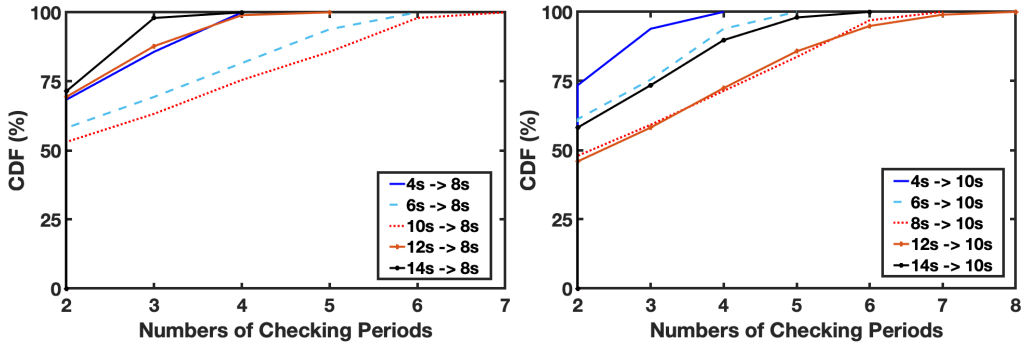
Fig. 14. Network performance when upward traffic and downward traffic are under different data rates.

downward traffic to 4s and 8s. We then create a network that mainly disseminates data by setting data intervals for the upward and downward traffic to 8s and 4s. Leveraging Eq. 7, the devices that run ATRIA set the slotframe length to 300$slots$. Figure 14(a) plots the CDF of the end-to-end PDR of each data flow under different traffic ratios. When the network mainly collects data, ATRIA provides the performance close to Optimal's. The minimum PDR is 95.0% and 72% of the data flows achieve 100% PDR. As a comparison, while the devices run ALICE, the maximum, median, and minimum PDRs are 98.0%, 58.7%, and 26.2%, respectively. When the network mainly disseminates data, ATRIA also performs better than ALICE. These results demonstrate the benefit of scheduling cells based on the traffic loads. Figure 14(b) plots the CDF of the end-to-end latency of each data flow. When the network runs mainly to disseminate data, it takes more time to deliver packets under ALICE. For example, the minimum, median, and maximum values under ALICE are 42ms, 319ms, and 2,070ms, respectively. As a comparison, the minimum, median, and maximum values are 63ms, 216ms, and 552ms, respectively, when the devices run ATRIA. When the network mainly collects data, we observe similar results. Figure 14(c) presents the CDF of energy consumption. When the network mainly disseminates data, ATRIA and ALICE provide comparable performance on energy efficiency. For example, the minimum, median, and maximum energy consumption values under ATRIA are 9.5J, 9.7J, and 21.4J, while those values under ALICE are 10.0J, 10.2J, and 16.8J, respectively. As a comparison, the devices consume more energy when allocating more cells under Optimal to achieve 100% PDR. We observe similar results when the network runs mainly to collect data. In summary, ATRIA provides higher reliability and lower latency with comparable energy consumption, when upward traffic and downward traffic are under different data rates.

5.4 Effectiveness of Schedule Adaptations

In this set of experiments, we validate the effectiveness of schedule adaptations and evaluate the network performance when the devices change the data generation rate or the network topology changes at runtime. We first configure all devices to change the data generation interval from 4s, 6s, 10s, 12s, and 14s to 8s at runtime. Then we configure all devices to change the data generation interval from 4s, 6s, 8s, 12s, and 14s to 10s. Next, we evaluate the performance when the network faces topology changes. We also repeat the experiments when configuring different numbers of devices (17, 33, and 50) to change the data interval from 10s to 6s. Finally, we evaluate the performance changes when Schedule Adaptor performs schedule adaptations in response to the data interval changes at runtime. We set T^T and C^T to two and one, respectively.

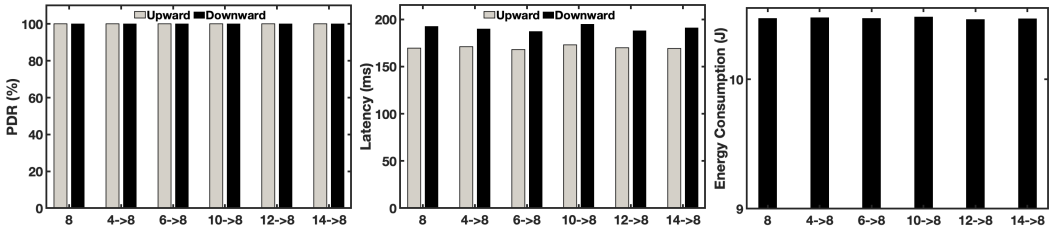
Figure 15(a) and Figure 15(b) plot the CDF of the time consumed by each communicating link to complete schedule adaptations when all devices change the data generation interval to 8s and 10s,



(a) Data generation interval changes from 4*s*, 6*s*, 10*s*, 12*s*, and 14*s* to 8*s*.

(b) Data generation interval changes from 4*s*, 6*s*, 8*s*, 12*s*, and 14*s* to 10*s*.

Fig. 15. Time consumed by schedule adaptations.



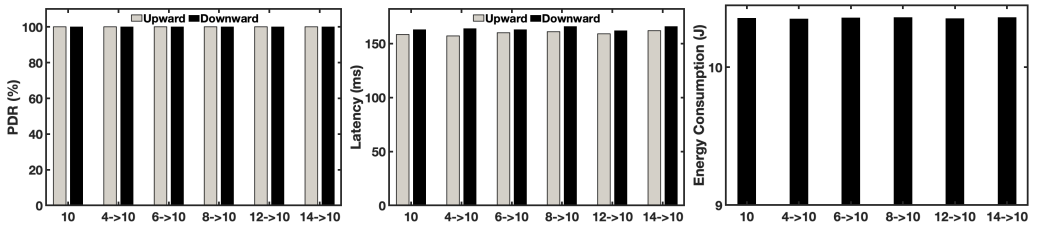
(a) PDR.

(b) Latency.

(c) Energy Consumption.

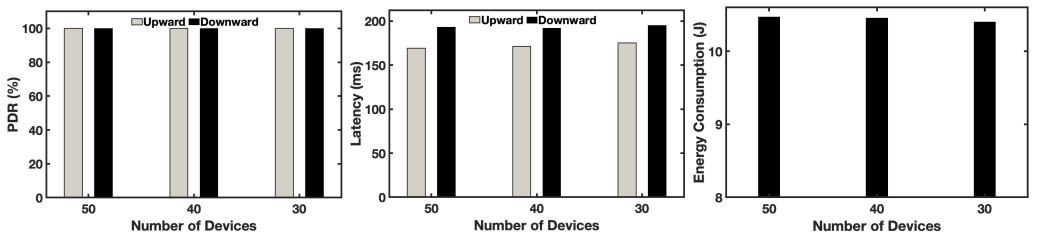
Fig. 16. Network performance when the data generation interval changes from 4*s*, 6*s*, 10*s*, 12*s*, and 14*s* to 8*s*.

respectively. As Figure 15(a) shows, the time consumed by schedule adaptations increases when the change of the data interval decreases. For example, when the data generation interval changes from 14*s* to 8*s*, 98.0% of the links complete their schedule adaptations within three checking periods and the other links take one more checking period. When the data interval changes from 10*s* to 8*s*, the upper quartile, and maximum time consumption values are five checking periods and seven checking periods, respectively. This is because it takes Schedule Adaptor more time to sense and confirm the slight traffic change by counting the data packets, especially for the links that are not responsible for forwarding traffic. Figure 15(b) shows similar results when the data generation interval changes to 10*s*. Figure 16 plots the network performance when the data generation interval changes to 8*s*. As Figure 16(a) shows, the PDRs of the upward and downward data flows are 100% after completing each schedule adaptation. As Figure 16(b) shows, the end-to-end latency after traffic change is approximately equal to the latency without traffic change. For example, the latency values of the upward data flows are 168*ms* and 173*ms*, respectively, when the data interval changes from 6*s* and 10*s* to 8*s*. As a comparison, the latency is 169*ms* when the data interval is consistently 8*s*. As Figure 16(c) shows, the energy consumption averaged among all devices is always around 10.5*J* after performing schedule adaptations, which is equal to the energy consumption when all devices consistently generate a packet every 8*s*. Figure 17 plots the network performance when the data generation interval changes to 10*s*. The PDRs of the upward and downward data flows are always 100%, while the energy consumption is around 10.3*J* after performing each schedule



(a) PDR. (b) Latency. (c) Energy Consumption.

Fig. 17. Network performance when the data generation interval changes from 4*s*, 6*s*, 8*s*, 12*s*, and 14*s* to 10*s*.



(a) PDR. (b) Latency. (c) Energy Consumption.

Fig. 18. Network performance when 10 or 20 devices leave the network.

adaptation. The end-to-end latency under each traffic change is close to the value under the fixed data interval. For example, the latency of the upward data flows is 159*ms* after we change the data interval from 12*s* to 10*s*, while the one under the fixed 10*s* interval is 158*ms*. The results shown in Figure 16 and Figure 17 demonstrate the effectiveness of Schedule Adaptor in meeting the traffic demand when all devices change the data rate at runtime.

Figure 18 plots the network performance when the network faces topology changes after 10 or 20 devices leave the network. All devices generate a data packet every 8*s*. As Figure 18(a) shows, the PDRs of the upward and downward data flows are always 100% after completing each schedule adaptation. As Figure 18(b) shows, the end-to-end latency increases slightly after 10 or 20 devices leave the network. For example, when the network consists of 50 devices, the latency of the upward data flows is 169*ms*. After 20 devices leave the network, the latency increases to 175 *ms*. This is because the numbers of scheduled cells are adjusted according to the new topology, leading to less cells allocated in each slotframe. As Figure 18(c) shows, the energy consumption averaged among all 30 devices is 10.4*J*, slightly lower that averaged among 50 devices (10.5*J*). This is because the averaged traffic load decreases when some devices leave the network. Figure 19 plots the numbers of cells scheduled for each device. As Figure 19 shows, the numbers of scheduled cells decrease after sets of devices leave the network. For instance, at first, Device 40 is responsible for forwarding the traffic of four devices and 18 cells are scheduled for its traffic. After 20 devices leave the network, the number of child nodes of Device 40 decreases to two and the number of scheduled cells drops to 10 accordingly. Similarly, 42 cells are scheduled for Device 47 at first. This number decreases to 34 or 22 after 10 or 20 devices leave the network. The results plotted in Figure 18 and Figure 19 demonstrate the effectiveness of ATRIA in meeting the traffic demand when the network topology changes.

Figure 20 plots the network performance when we change the number of devices that change the data generation interval. As Figure 20(a) shows, the PDRs of the upward and downward data flows

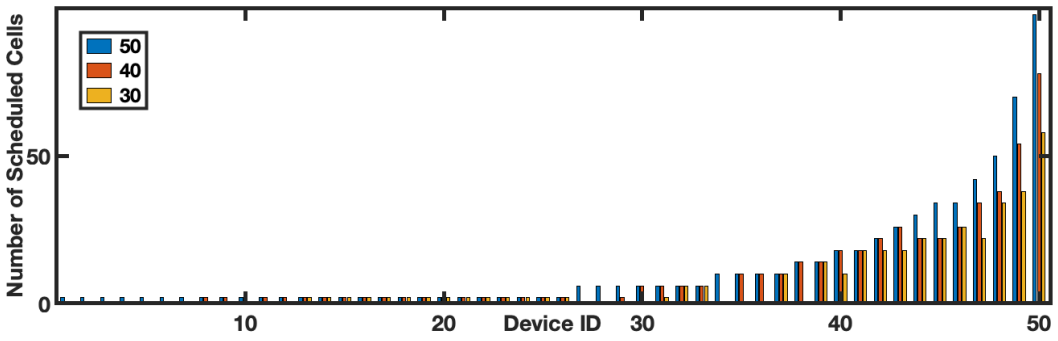
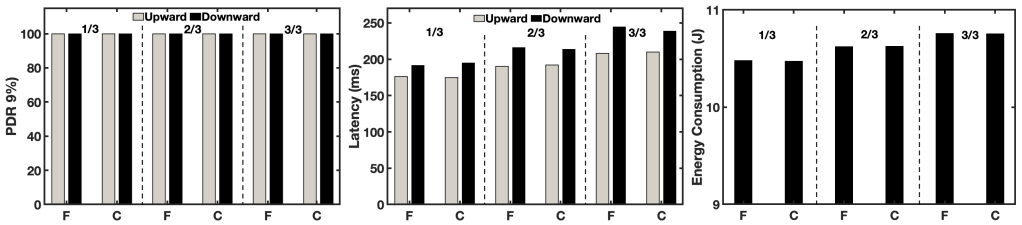


Fig. 19. Number of scheduled cells for each device when 10 or 20 devices leave the network.

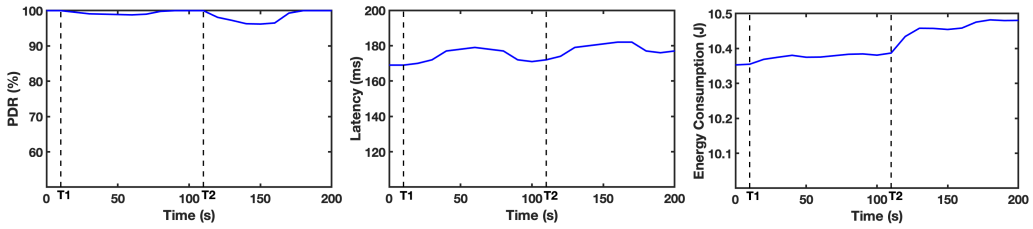


(a) PDR. (b) Latency. (c) Energy Consumption.

Fig. 20. Network performance when different numbers of devices change their data generation intervals. F: devices generate data at the fixed rates; C: 17, 33, and 50 devices change the data generation interval from 10s to 6s.

are always 100% after 17, 33, and 50 devices are configured to change the data rate and Schedule Adaptor completes schedule adaptation. As Figure 20(b) shows, when 17 devices are configured to change the data generation interval from 10s to 6s, the latency of the upward data flows is 175ms, which is close to the one (176ms) provided by ATRIA when devices are set to generate data at the fixed rates. We observe similar results when 33 and 50 devices change the data interval. As Figure 20(c) shows, when 33 devices change the data interval, the averaged energy consumption is 10.6J, which is close to the energy consumed by devices when generating data at the fixed rates. We also observe similar results in other groups. The results clearly show that ATRIA provides the desirable performance with the help of Schedule Adaptor when some devices in the network change their data generation intervals.

Figure 21 plots the performance changes when Schedule Adaptor performs schedule adaptations in response to the changes of the data generation interval at runtime. As Figure 21(a) shows, the PDR averaged among all data flows decreases from 100% at T1 (10s) to 98.9% at 50s, and rises to 100% at 90s. It takes 80s for Schedule Adaptor to adjust the numbers of scheduled cells to accommodate the change. Similarly, the PDR starts to decrease from 100% when the data generation intervals change again at T2 (110s) and then increase to 100% after Schedule Adaptor finishes the schedule adaptation. As Figure 21(b) shows, the end-to-end latency also decreases and then increases when the data generation interval changes at runtime. For instance, the latency rises from 169ms at T1 to 178ms at 60s, and then decreases to 172ms at 90s. Similarly, the latency increases from 172ms at T2 to 182ms at 160s, and then decreases to 176ms after Schedule Adaptor finishes the schedule adaptation. As Figure 21(c) shows, the averaged energy consumption increases from 10.35J at T1 to 10.38J at T2, and reaches 10.48J at 180s. This is because the overall traffic of the network



(a) PDR. (b) Latency. (c) Energy Consumption.

Fig. 21. Performance changes when the data generation interval of the network changes at runtime. At T1, half of devices change the data interval from 10*s* to 8*s*, while the other half change the data interval from 10*s* to 12*s*. At T2, the data intervals further change from 8*s* to 6*s* and 12*s* to 14*s*, respectively.

increases and the total number of cells scheduled in each slotframe rises accordingly. The results show ATRIA can help the network achieve the desirable performance when some devices change the data generation intervals at runtime.

6 RELATED WORK

Traffic-aware scheduling for Time Division Multiple Access (TDMA) based networks has been studied in the literature [44]. For example, Wang et al. proposed to use a general weighted link coloring model to estimate the traffic and schedule transmissions accordingly [52] and Gobriel et al. proposed to enable slot stealing and sleeping to adjust the slot assignments when facing different traffic loads [17]. Unfortunately, the existing solutions developed for TDMA based networks are not directly applicable to the TSCH networks, because they neither consider multi-channel communication nor support channel hopping.

In recent years, there has been increasing interest in developing new transmission scheduling solutions for TSCH networks. Many centralized scheduling algorithms have been developed in the literature. For instance, Jin et al. proposed a method that enables sequential multi-hop scheduling and allocates more resources to the vulnerable links [25] and Palattella et al. proposed to allocate the minimum number of needed cells to each device based on its traffic [39]. The centralized scheduling algorithm simplifies network management at the cost of poor network scalability. Significant efforts have been made to develop decentralized scheduling algorithms to enhance network scalability. For example, Tinka et al. introduced a distributed scheduling algorithm that lets all devices continuously advertise their presences and allows neighbors to discover and contact one another [49]. Palattella et al. presented an algorithm that dynamically matches the scheduled bandwidth between pairs of devices to their actual traffic loads [40]. Those decentralized scheduling methods require neighboring devices to exchange information at runtime. More recently, autonomous scheduling methods have been proposed for TSCH networks to eliminate the negotiation overhead between neighbors [13, 14, 38, 45, 46]. For example, Duquennoy et al. developed Orchestra [11], which allows each device to generate its transmission schedule based on the local routing information. Kim et al. developed ALICE [29], which allocates a unique cell to each link and uses all available channels for communication. Unfortunately, the existing autonomous methods fail to consider the traffic loads of different devices, resulting in compromised network performance at high data rates. There exist some traffic-aware scheduling solutions recently developed for TSCH networks. For example, Jeong et al. developed TESLA [24], which enables devices to dynamically self-adjust the slotframe length based on the traffic by exchanging information. Jung et al. proposed a parameterized algorithm that works adaptively to traffic intensity, slotframe length, and reliability requirements [26]. However, the existing traffic-aware methods depend on exchanging information between neighbors and

introduce communication overhead. Although e-TSCH-Orch [43] allows devices to dynamically add a number of consecutive slots based on the number of queued packets, without the need to exchange information, it may ruin the Orchestra priority setting and result in severe slot conflicts. In contrast to the existing traffic-aware solutions, ATRIA does not require neighboring device to exchange information and can significantly reduce slot conflicts.

Opportunistic routing provides a new paradigm for improving network performance by providing better routes [3, 4, 9, 33, 34]. For instance, Coutinho et al. proposed an opportunistic routing protocol that routes data packets from sensors to multiple access points [9] and Liu et al. introduced a robust routing protocol that combines opportunistic routing and asynchronous sleep to improve the reliability and efficiency [34]. Improving transmission scheduling and enhancing routing are complementary to each other. Our paper focuses on improving network performance by providing better transmission schedules for the networks that adopt RPL as the routing protocol and leaves the integration of our transmission scheduling method and opportunistic routing protocols as future work.

The security aspects of industrial networks have been studied recently and many enhancements have been developed [7, 8, 18, 19, 42, 47, 54, 55]. For example, Rajakaruna et.al proposed to use a mobile edge server to enable end-to-end secure connectivity [42] and Cheng et al. identified the vulnerabilities of the TSCH channel hopping [6].

7 CONCLUSIONS

In this paper, we present ATRIA, a novel autonomous traffic-aware transmission scheduling method for industrial WSANs. The device that runs ATRIA can detect its traffic load based on its local routing information, then schedule its transmission accordingly, and adapt the schedule at runtime. We have implemented ATRIA under Contiki and evaluated its performance using a network that consists of 50 devices on the FIT IoT-LAB testbed. Experimental results show that ATRIA provides significantly higher end-to-end network reliability and lower end-to-end latency without introducing additional overhead compared to the existing method.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under grant CNS-2046538 and CNS-2150010.

REFERENCES

[1] 802.15.4e. 2013. IEEE802.15.4e WPAN Task Group. http://www.ieee802.org/15/pub/TG4e.html
[2] Cédric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frédéric Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. 2015. FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed. In *IEEE World Forum on Internet of Things (IEEE WF-IoT)*. IEEE, USA, 459–464.
[3] Sanjit Biswas and Robert Morris. 2004. Opportunistic Routing in Multi-Hop Wireless Networks. *SIGCOMM Comput. Commun. Rev.* 34, 1 (2004), 69–74.
[4] Azzedine Boukerche and Amir Darehshoorzadeh. 2014. Opportunistic Routing in Wireless Networks: Models, Algorithms, and Classifications. *ACM Comput. Surv.* 47, 2, Article 22 (2014), 36 pages.
[5] Xia Cheng and Mo Sha. 2021. ATRIA: Autonomous Traffic-Aware Scheduling for Industrial Wireless Sensor-Actuator Networks. In *IEEE International Conference on Network Protocols (ICNP)*. IEEE, USA, 1–12.
[6] Xia Cheng, Junyang Shi, and Mo Sha. 2019. Cracking the Channel Hopping Sequences in IEEE 802.15.4e-Based Industrial TSCH Networks. In *International Conference on Internet of Things Design and Implementation*. Association for Computing Machinery, New York, NY, USA, 130–141.
[7] Xia Cheng, Junyang Shi, and Mo Sha. 2020. Cracking Channel Hopping Sequences and Graph Routes in Industrial TSCH Networks. *ACM Trans. Internet Technol.* 20, 3, Article 23 (2020), 28 pages.
[8] Xia Cheng, Junyang Shi, Mo Sha, and Linke Guo. 2021. Launching Smart Selective Jamming Attacks in WirelessHART Networks. In *IEEE Conference on Computer Communications*. IEEE, USA, 1–10.

[9] Rodolfo W. L. Coutinho, Azzedine Boukerche, Luiz F. M. Vieira, and Antonio A. F. Loureiro. 2016. Geographic and Opportunistic Routing for Underwater Sensor Networks. *IEEE Trans. Comput.* 65, 2 (2016), 548–561.

[10] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. 2004. Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *IEEE International Conference on Local Computer Networks*. IEEE, USA, 455–462.

[11] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. 2015. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *ACM Conference on Embedded Networked Sensor Systems*. Association for Computing Machinery, New York, NY, USA, 337–350.

[12] Atis Elsts, Xenofon Fafoutis, George Oikonomou, Robert Piechocki, and Ian Craddock. 2020. TSCH Networks for Health IoT: Design, Evaluation, and Trials in the Wild. *ACM Trans. Internet Things* 1, 2 (2020), 27 pages.

[13] Atis Elsts, Xenofon Fafoutis, James Pope, George Oikonomou, Robert Piechocki, and Ian Craddock. 2017. Scheduling High-Rate Unpredictable Traffic in IEEE 802.15.4 TSCH Networks. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, USA, 3–10.

[14] Atis Elsts, Seohyang Kim, Hyung-Sin Kim, and Chongkwon Kim. 2020. An Empirical Survey of Autonomous Scheduling Methods for TSCH. *IEEE Access* 8 (2020), 67147–67165. https://doi.org/10.1109/ACCESS.2020.2980119

[15] Emerson. 2020. Emerson. https://www.emerson.com/en-us/expertise/automation/industrial-internet-things/pervasive-sensing-solutions/wireless-technology

[16] Omprakash Gnawali and Philip Levis. 2012. The Minimum Rank with Hysteresis Objective Function. https://tools.ietf.org/html/rfc6719

[17] Sameh Gobriel, Daniel Mosse, and Robert Cleric. 2009. TDMA-ASAP: Sensor Network TDMA Scheduling with Adaptive Slot-Stealing and Parallelism. In *IEEE International Conference on Distributed Computing Systems*. IEEE, USA, 458–465.

[18] 802.15 WG Wireless Specialty Networks (WSN) Working Group. 2016. IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams. https://standards.ieee.org/standard/802_15_9-2016.html

[19] Manjesh K. Hanawal, Diep N. Nguyen, and Marwan Krunz. 2016. Jamming attack on in-band full-duplex communications: Detection and countermeasures. In *IEEE Conference on Computer Communications*. IEEE, USA, 1–9.

[20] HART. 2020. HART Communication Protocol and Foundation (Now the FieldComm Group). https://fieldcommgroup.org/

[21] IEC. 2017. WIA-FA. https://webstore.iec.ch/publication/32718

[22] IETF. 2020. 6TiSCH: IPv6 over the TSCH mode of IEEE 802.15.4e. https://datatracker.ietf.org/wg/6tisch/documents/

[23] ISA100. 2018. ISA100. http://www.isa100wci.org/

[24] Seungbeom Jeong, Jeongyeup Paek, Hyung-Sin Kim, and Saewoong Bahk. 2019. TESLA: Traffic-Aware Elastic Slotframe Adjustment in TSCH Networks. *IEEE Access* 7 (2019), 130468–130483. https://doi.org/10.1109/ACCESS.2019.2940457

[25] Yichao Jin, Parag Kulkarni, James Wilcox, and Mahesh Sooriyabandara. 2016. A Centralized Scheduling Algorithm for IEEE 802.15.4e TSCH based Industrial Low Power Wireless Networks. In *IEEE Wireless Communications and Networking Conference*. IEEE, USA, 1–6.

[26] Jinhwan Jung, Daewoo Kim, Joonki Hong, Joohyun Kang, and Yung Yi. 2018. Parameterized slot scheduling for adaptive and autonomous TSCH networks. In *IEEE Conference on Computer Communications Workshops*. IEEE, USA, 76–81.

[27] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig. 2013. Recommendations for Implementing the Strategic Initiative Industrie 4.0. http://alvarestech.com/temp/tcn/CyberPhysicalSystems-Industrial4-0.pdf

[28] Seohyang Kim, Hyung-Sin Kim, and Chongkwon Kim. 2019. ALICE. https://github.com/skimskimskim/ALICE

[29] Seohyang Kim, Hyung-Sin Kim, and Chongkwon Kim. 2019. ALICE: Autonomous Link-Based Cell Scheduling for TSCH. In *International Conference on Information Processing in Sensor Networks*. Association for Computing Machinery, New York, NY, USA, 121–132.

[30] Bo Li, Yehan Ma, Tyler Westenbroek, Chengjie Wu, Humberto Gonzalez, and Chenyang Lu. 2016. Wireless Routing and Control: A Cyber-Physical Case Study. In *International Conference on Cyber-Physical Systems*. IEEE, USA, 1–10.

[31] Bo Li, Lanshun Nie, Chengjie Wu, Humberto Gonzalez, and Chenyang Lu. 2015. Incorporating Emergency Alarms in Reliable Wireless Process Control. In *International Conference on Cyber-Physical Systems*. Association for Computing Machinery, New York, NY, USA, 218–227.

[32] Bo Li, Zhuoxiong Sun, Kirill Mechitov, Gregory Hackmann, Chenyang Lu, Shirley J. Dyke, Gul Agha, and Billie F. Spencer. 2013. Realistic Case Studies of Wireless Structural Control. In *International Conference on Cyber-Physical Systems*. Association for Computing Machinery, New York, NY, USA, 179–188.

[33] Haitao Liu, Baoxian Zhang, Hussein T. Mouftah, Xiaojun Shen, and Jian Ma. 2009. Opportunistic routing for wireless ad hoc and sensor networks: Present and future directions. *IEEE Communications Magazine* 47, 12 (2009), 103–109.

[34] Shucheng Liu, Mo Sha, and Liusheng Huang. 2010. ORAS: Opportunistic routing with asynchronous sleep in Wireless Sensor Networks. In *2010 2nd International Conference on Future Computer and Communication*, Vol. 3. IEEE, USA, 234–238.

[35] Chenyang Lu, Abusayeed Saifullah, Bo Li, Mo Sha, Humberto Gonzalez, Dolvara Gunatilaka, Chengjie Wu, Lanshun Nie, and Yixin Chen. 2016. Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems. *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems* 104, 5 (2016), 1013–1024.

[36] M3. 2020. IoT-LAB M3. https://iot-lab.github.io/docs/boards/iot-lab-m3/

[37] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, and Alex Marrs. 2013. Disruptive Technologies: Advances that will Transform Life, Business, and the Global Economy. http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/disruptive-technologies

[38] Sukho Oh, DongYeop Hwang, Ki-Hyung Kim, and Kangseok Kim. 2018. Escalator: An Autonomous Scheduling Scheme for Convergecast in TSCH. *Sensors* 18, 4 (2018). https://doi.org/10.3390/s18041209

[39] Maria Rita Palattella, Nicola Accettura, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, and Thomas Engel. 2013. On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH. *IEEE Sensors Journal* 13, 10 (2013), 3655–3666. https://doi.org/10.1109/JSEN.2013.2266417

[40] Maria Rita Palattella, Thomas Watteyne, Qin Wang, Kazushi Muraoka, Nicola Accettura, Diego Dujovne, Luigi Alfredo Grieco, and Thomas Engel. 2016. On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks. *IEEE Sensors Journal* 16, 2 (2016), 550–560. https://doi.org/10.1109/JSEN.2015.2480886

[41] Rajeev Piyare, George Oikonomou, and Atis Elsts. 2020. TSCH for Long Range Low Data Rate Applications. *IEEE Access* 8 (2020), 228754–228766. https://doi.org/10.1109/ACCESS.2020.3046769

[42] Archana Rajakaruna, Ahsan Manzoor, Pawani Porambage, Madhusanka Liyanage, Mika Ylianttila, and Andrei Gurtov. 2019. Enabling End-to-End Secure Connectivity for Low-Power IoT Devices with UAVs. In *IEEE Wireless Communications and Networking Conference Workshop*. IEEE, USA, 1–6.

[43] Sana Rekik, Nouha Baccour, Mohamed Jmaiel, Khalil Drira, and Luigi Alfredo Grieco. 2018. Autonomous and traffic-aware scheduling for TSCH networks. *Computer Networks* 135 (2018), 201–212. https://doi.org/10.1016/j.comnet.2018.02.023

[44] Aggeliki Sgora, Dimitrios J. Vergados, and Dimitrios D. Vergados. 2015. A Survey of TDMA Scheduling Schemes in Wireless Multihop Networks. *Comput. Surveys* 47, 3 (2015), 39 pages. https://doi.org/10.1145/2677955

[45] Junyang Shi, Mo Sha, and Zhicheng Yang. 2018. DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks. In *International Conference on Distributed Computing Systems*. IEEE, USA, 354–364. https://doi.org/10.1109/ICDCS.2018.00043

[46] Junyang Shi, Mo Sha, and Zhicheng Yang. 2019. Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks. *IEEE/ACM Transactions on Networking* 27, 4 (2019), 1669–1682. https://doi.org/10.1109/TNET.2019.2925816

[47] Ali Hassan Sodhro, Sandeep Pirbhulal, Gul Hassan Sodhro, Andrei Gurtov, Muhammad Muzammal, and Zongwei Luo. 2019. A Joint Transmission Power Control and Duty-Cycle Approach for Smart Healthcare System. *IEEE Sensors Journal* 19, 19 (2019), 8479–8486. https://doi.org/10.1109/JSEN.2018.2881611

[48] Pascal Thubert. 2012. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). https://tools.ietf.org/html/rfc6552

[49] Andrew Tinka, Thomas Watteyne, and Kris Pister. 2010. A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping. In *Ad Hoc Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 201–216.

[50] Jean-Philippe Vasseur, Mijeom Kim, Kris Pister, Nicolas Dejean, and Dominique Barthel. 2012. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. https://tools.ietf.org/html/rfc6551

[51] Thomas Wang. 2007. 32-bit Integer Hash Function. https://gist.github.com/badboy/6267743

[52] Weizhao Wang, Yu Wang, Xiang-Yang Li, Wen-Zhan Song, and Ophir Frieder. 2006. Efficient Interference-Aware TDMA Link Scheduling for Static Wireless Networks. In *International Conference on Mobile Computing and Networking*. Association for Computing Machinery, New York, NY, USA, 262–273.

[53] WirelessHART. 2019. WirelessHART. https://fieldcommgroup.org/technologies/wirelesshart

[54] Liyang Zhang, Zhangyu Guan, and Tommaso Melodia. 2014. Cooperative anti-jamming for infrastructure-less wireless networks with stochastic relaying. In *IEEE Conference on Computer Communications*. IEEE, USA, 549–557.

[55] Yifei Zou, Dongxiao Yu, Libing Wu, Jiguo Yu, Yu Wu, Qiang-sheng Hua, and Francis C.M. Lau. 2019. Fast Distributed Backbone Construction Despite Strong Adversarial Jamming. In *IEEE Conference on Computer Communications*. IEEE, USA, 1027–1035.