ATRIA: *A*utonomous *Tra*ff*ic*-*A*ware Scheduling for Industrial Wireless Sensor-Actuator Networks

Xia Cheng, Mo Sha
Knight Foundation School of Computing and Information Sciences
Florida International University
{xcheng, msha}@fiu.edu

***Abstract*—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, such as WirelessHART, ISA100, WIA-FA, and 6TiSCH, 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 *ATRIA*, a novel *A*utonomous *Tra*ff*ic*-*A*ware 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 that 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.**

***Index Terms*—Industrial Wireless Sensor-Actuator Networks, IEEE 802.15.4, Transmission Scheduling, TSCH, RPL**

I. Introduction

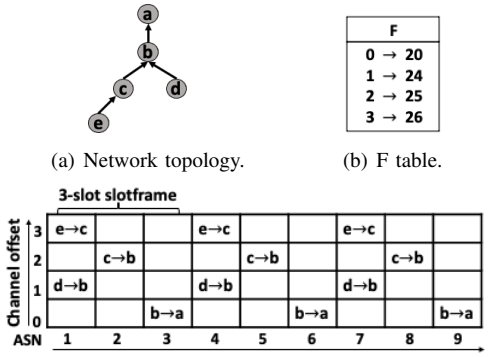
Industrial Internet of Things (IoT), which underlies the Fourth Industrial Revolution (or Industry 4.0) [1], promises one of the largest potential economic effects of IoT – up to $47 trillion in added value globally by 2025, according to the McKinsey report on future disruptive technologies [2]. Industrial 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 wired solutions, such as HART [3]. 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 [4], ISA100 [5], WIA-FA [6], and 6TiSCH [7], 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 [8]. 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 [9]. 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) [10] have been deployed for various applications [11], [12]. 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 [13], which allows each network device to generate its transmission schedule based on its local routing information, and Kim et al. developed ALICE [14], 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 [15]. 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

978-1-6654-4131-5/21/$31.00 ©2021 IEEE

[1]A cell denotes the combination of a time slot and a physical channel.



(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.

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*ffic-*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, and then schedule one or more cells based on its specific traffic load. We have implemented ATRIA under Contiki [16] 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 II introduces the background of TSCH, RPL, and ALICE. Section III presents our experimental study. Section IV introduces our design of ATRIA. Section V presents our experimental evaluation. Section VI reviews the related work. Section VII concludes the paper.

II. Background

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

A. TSCH

TSCH was amended into the IEEE 802.15.4e standard [17] 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:

$$channel = F[(ASN + ChannelOffset)\%L_C] \quad (1)$$

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.

B. 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) [18] is one of the commonly used OFs, which adopts the Expected Transmission Count (ETX) metric [19] 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 Acknowl-

[2]A slotframe consists of a group of successive time slots, which repeats over time.

edgment (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.

C. 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 [14]. 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

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

where $Hash(x)$ is a HASH function used to randomize the input value to reduce the conflict between different directional links [20], 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

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

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 $(2N - 2)/(M - 1)$.

III. 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 [21] from the FIT IoT-LAB testbed [15] to form a mesh network and run the ALICE implementation provided by Kim et al. [22]. Figure 2 plots the device deployment for our studies. All

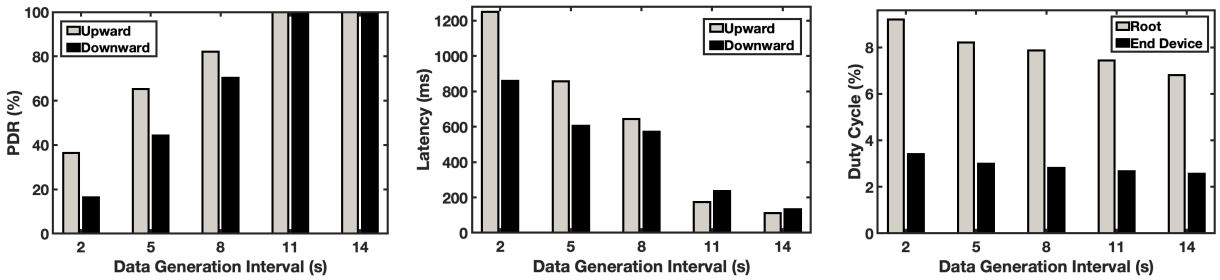
Fig. 2. Testbed used for our studies. All 50 devices are deployed on the same floor in an office building. One device serves as the border router (Root) and the rest are end devices.

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.

A. ALICE's Performance

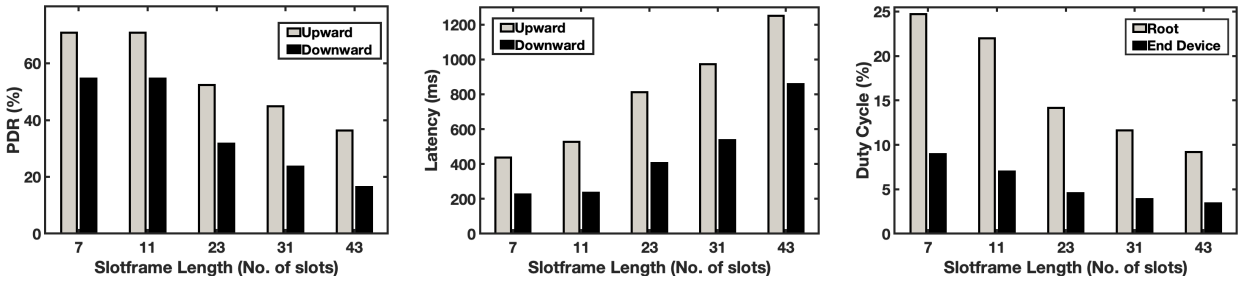
We first examine the network performance when we increase the data generation interval of each device from 2s to 14s for both upward traffic and downward traffic, typical data rates for industrial applications [23], [24], [25]. We set the length of the unicast slotframe to 43$slots$ (by default in ALICE). 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 5 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 5 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 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



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

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



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

Fig. 4. Performance when the network uses different slotframe lengths.

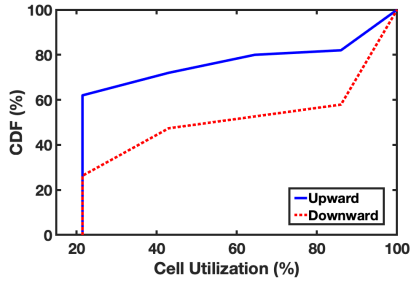


Fig. 5. Cell utilization for upward and downward traffic.

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.*

B. 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 $2s$ for both upward traffic and downward traffic in all experiments. Figure 4 plots the end-to-end PDR, the end-to-end latency, and the radio duty cycle when the slotframe lengths are $7slots$, $11slots$, $23slots$, $31slots$, and $43slots$ (the five default values in ALICE), respectively. As Figure 4(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 $43slots$ to $11slots$. 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 $7slots$. This is because the contention for available time slots becomes so severe that many cell allocation failures occur when the slotframe is too small. Figure 4(b) and 4(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.*

IV. Our Design of ATRIA

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

A. Overview

As Figure 6 shows, operating between RPL and TSCH, ATRIA takes the routing information from RPL and the parameters specified by the application 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

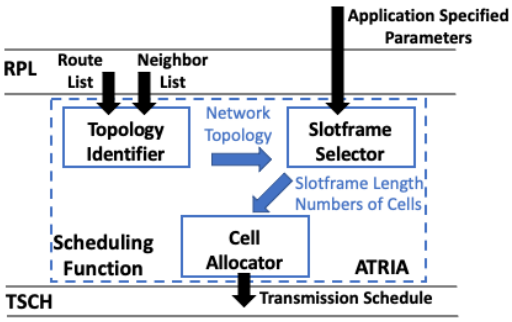


Fig. 6. Overview of ATRIA.

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 III-A 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, and then schedules one or more cells to each directional link with three modules:

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

We will present our designs of those three modules next.

B. 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 IV-C) and allocate cells (See Section IV-D).

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.

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

C. Slotframe Selector

Our experimental study in Section III-B shows that the selection of the slotframe length significantly affects network performance. As Figure 4 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.

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 all 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

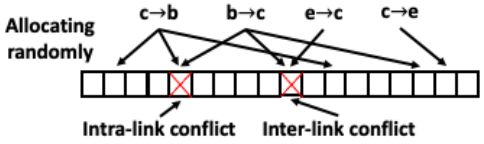


Fig. 7. Example slot conflicts. A slotframe consists of 18 slots. The numbers of cells scheduled for link $c \rightarrow b$, $b \rightarrow c$, $e \rightarrow c$, and $c \rightarrow e$ are three, three, one, and one, respectively.

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:

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

where R denotes the success rate of the chosen cell allocation algorithm. We will introduce our cell allocation algorithm in Section IV-D. 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.

D. 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. 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 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

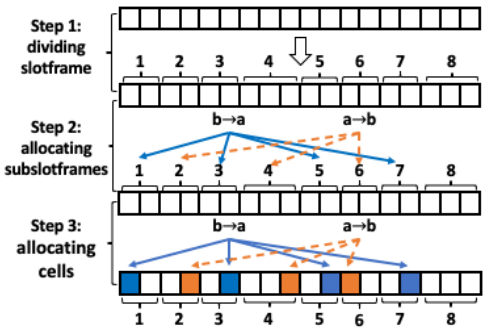


Fig. 8. Example slot allocations. A slotframe that consists of 18 slots is divided into eight subslotframes. Seven cells distribute among eight subslotframes uniformly. Each cell is allocated to a slot in its subslotframe.

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

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

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

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

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

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

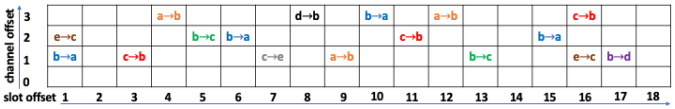


Fig. 9. Example cell allocations to four pairs of bidirectional links in Figure 1(a). A slotframe that consists of 18 slots is divided into eight subslotframes.

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 $b \to a$, $a \to b$, $c \to b$, $b \to c$, $d \to b$, $b \to d$, $e \to c$, and $c \to e$ 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 $E(x,y,z) > R(x,y,z)$.

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.

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

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

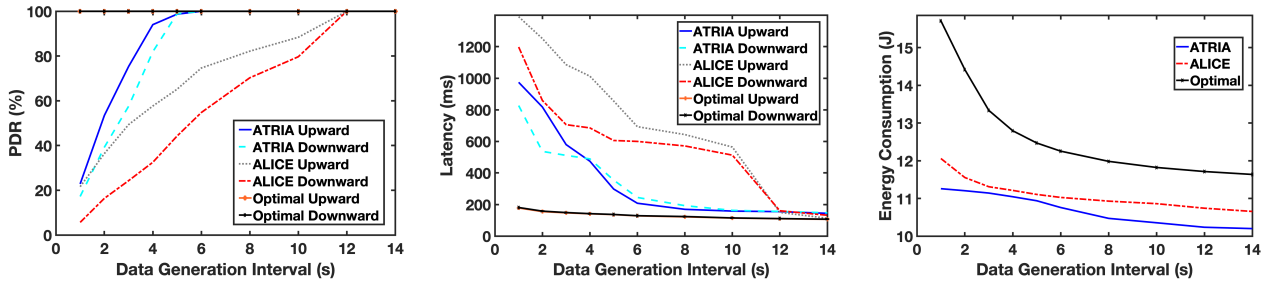
After expanding and merging similar items by following the similar process, we have $E(x,y,q,z) > R(x,y,q,z)$. Similarly, under the case where a device has a parent and N child nodes, we construct the expressions for $E(x_1, x_2 \cdots x_{N+1}, z)$ and $R(x_1, x_2 \cdots x_{N+1}, z)$ and compare them 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.

V. 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 [15]. 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. Finally, we vary the ratio of upward traffic to downward traffic and study its impact on network performance. 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. To compare the performance of ATRIA and ALICE, all devices operate on four channels and send packets using the transmission power of -17dBm (the default values 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 slot lasts 10ms.

A. 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

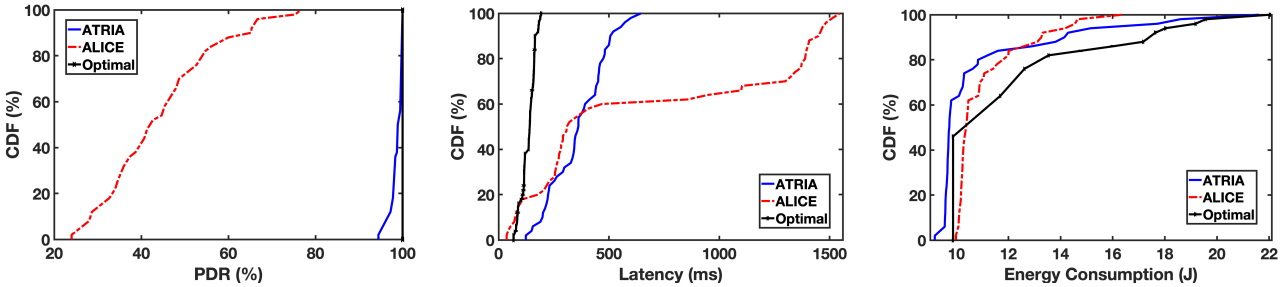


(a) End-to-end PDR averaged among 49 data flows.

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

(c) Energy consumption averaged among 50 devices.

Fig. 10. Network performance when we vary the data generation interval from $2s$ to $14s$. In each experiment, all devices generate data at the same rate.



(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. 11. Network performance when the data generation interval is $5s$.

devices that run ATRIA always select $200slots$ as the length. We use the default length ($43slots$) for ALICE and $100slots$ for Optimal.

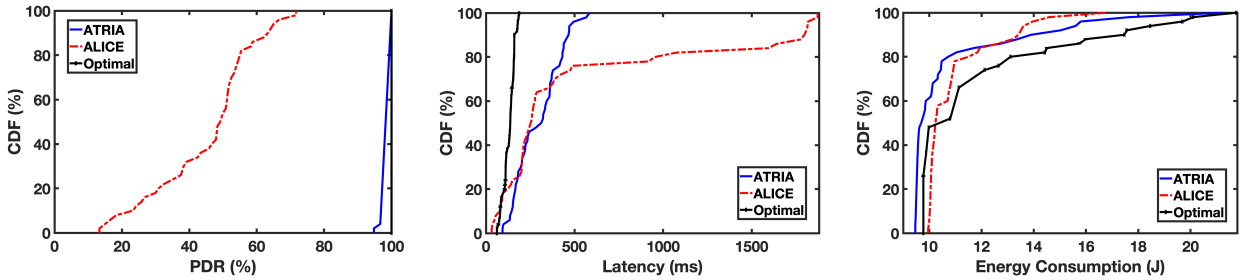
Figure 10 plots the network performance when we vary the data generation interval. As Figure 10(a) shows, the PDRs of the upward and downward data flows are 100% when the data generation interval is no less than $12s$. As expected, all methods perform well when the data is generated at low rates. The PDRs of the upward and downward data flows under ALICE decrease 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 10(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 10(c) plots energy consumption[5] averaged among all 50 devices. As Figure 10(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.

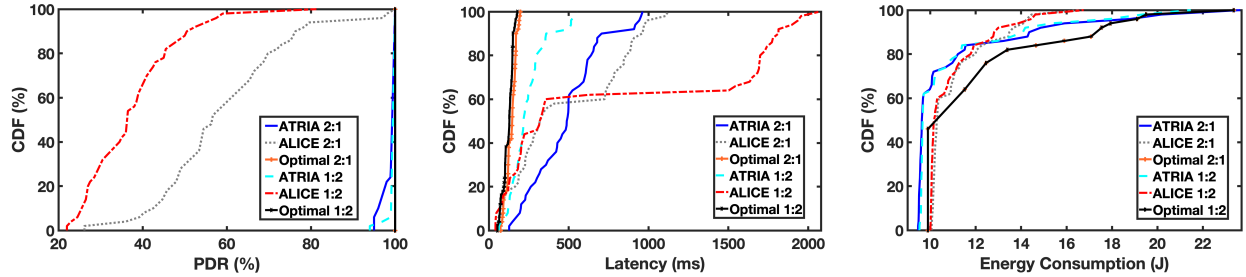
Figure 11 provides a detailed look at the network performance when the data generation interval is $5s$. Figure 11(a) plots the CDF of the end-to-end PDR of each data flow. When the devices run ATRIA, 45.0% of the data flows achieve 100% PDR. The median and minimum PDRs are 98.9% and 94.4%, respectively. As a comparison, the maximum, median, and minimum PDRs under ALICE are 76.7%, 42.0%, and 24.0%, respectively. Figure 11(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

[5]Energy consumption is calculated based on the measured duty cycles and the M3 device datasheet [21].



(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 devices generate data at multiple rates.



(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 upward traffic and downward traffic are under different data rates.

and reduce the latency by scheduling cells based on the traffic demands. As Figure 11(c) shows, ATRIA and ALICE provide comparable performance 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 10 and Figure 11 show, compared to ALICE, ATRIA provides significantly higher reliability and lower latency without introducing additional energy overhead at high data rates, while it consumes less energy to achieve comparable performance on the network reliability and latency at low data rates.

B. 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 12(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 12(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 12(c) plots the CDF of energy consumption. As Figure 12(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.

C. 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 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 13(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 13(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 13(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 without additional energy overhead, when upward traffic and downward traffic are under different data rates.

VI. Related Work

Traffic-aware scheduling for Time Division Multiple Access (TDMA) based networks has been studied in the literature [26]. For example, Wang et al. proposed to use a general weighted link coloring model to estimate the traffic and schedule transmissions accordingly [27] and Gobriel et al. proposed to enable slot stealing and sleeping to adjust the slot assignments when facing different traffic loads [28]. 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 [29] and Palattella et al. proposed to allocate the minimum number of needed cells to each device based on its traffic [30]. 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 [31]. Palattella et al. presented an algorithm that dynamically matches the scheduled bandwidth between pairs of devices to their actual traffic loads [32]. 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 [33], [34], [35], [36], [37]. For example, Duquennoy et al. developed Orchestra [13], which allows each device to generate its transmission schedule based on the local routing information. Kim et al. developed ALICE [14], 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 [38], 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 [39]. However, the existing traffic-aware methods depend on exchanging information between neighbors and introduce communication overhead. Although e-TSCH-Orch [40] 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.

The security aspects of industrial networks have been studied recently and many enhancements have been developed [41], [42], [43], [44], [45], [46], [47], [48]. For example, Rajakaruna et.al proposed to use a mobile edge server to enable end-to-end secure connectivity [45] and Cheng et al. identified the vulnerabilities of the TSCH channel hopping [49].

VII. 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 and then schedule its transmissions accordingly. 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. The authors thank the anonymous reviewers and the shepherd Andrei Gurtov at Linkoping University for their insightful comments.

REFERENCES

[1] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for Implementing the Strategic Initiative Industrie 4.0," 2013. [Online]. Available: http://alvarestech.com/temp/tcn/CyberPhysicalSystems-Industrial4-0.pdf

[2] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs, "Disruptive Technologies: Advances that will Transform Life, Business, and the Global Economy," 2013. [Online]. Available: http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/disruptive-technologies

[3] HART, "HART Communication Protocol and Foundation (Now the FieldComm Group)," 2020. [Online]. Available: https://fieldcommgroup.org/

[4] WirelessHART, "WirelessHART," 2019. [Online]. Available: https://fieldcommgroup.org/technologies/wirelesshart

[5] ISA100, "ISA100," 2018. [Online]. Available: http://www.isa100wci.org/

[6] IEC, "WIA-FA," 2017. [Online]. Available: https://webstore.iec.ch/publication/32718

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

[8] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems*, vol. 104, no. 5, pp. 1013–1024, 2016.

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

[10] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," 2012. [Online]. Available: https://tools.ietf.org/html/rfc6552

[11] A. Elsts, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "TSCH Networks for Health IoT: Design, Evaluation, and Trials in the Wild," *ACM Trans. Internet Things*, vol. 1, no. 2, 2020.

[12] R. Piyare, G. Oikonomou, and A. Elsts, "TSCH for Long Range Low Data Rate Applications," *IEEE Access*, vol. 8, pp. 228 754–228 766, 2020.

[13] S. Duquennoy, B. A. Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *ACM Conference on Embedded Networked Sensor Systems*, 2015.

[14] S. Kim, H.-S. Kim, and C. Kim, "ALICE: Autonomous Link-Based Cell Scheduling for TSCH," in *International Conference on Information Processing in Sensor Networks*, 2019.

[15] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed," in *IEEE World Forum on Internet of Things (IEEE WF-IoT)*, 2015.

[16] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *IEEE International Conference on Local Computer Networks*, 2004.

[17] 802.15.4e, "IEEE802.15.4e WPAN Task Group," 2013. [Online]. Available: http://www.ieee802.org/15/pub/TG4e.html

[18] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," 2012. [Online]. Available: https://tools.ietf.org/html/rfc6719

[19] E. JP. Vasseur, E. M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks," 2012. [Online]. Available: https://tools.ietf.org/html/rfc6551

[20] T. Wang, "32-bit Integer Hash Function," 2007. [Online]. Available: https://gist.github.com/badboy/6267743

[21] M3, "IoT-LAB M3," 2020. [Online]. Available: https://iot-lab.github.io/docs/boards/iot-lab-m3/

[22] S. Kim, H.-S. Kim, and C. Kim, "ALICE," 2019. [Online]. Available: https://github.com/skimskimskim/ALICE

[23] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu, "Wireless Routing and Control: A Cyber-Physical Case Study," in *International Conference on Cyber-Physical Systems*, 2016.

[24] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu, "Incorporating Emergency Alarms in Reliable Wireless Process Control," in *International Conference on Cyber-Physical Systems*, 2015.

[25] B. Li, Z. Sun, K. Mechitov, G. Hackmann, C. Lu, S. J. Dyke, G. Agha, and B. F. Spencer, "Realistic Case Studies of Wireless Structural Control," in *International Conference on Cyber-Physical Systems*, 2013.

[26] A. Sgora, D. J. Vergados, and D. D. Vergados, "A Survey of TDMA Scheduling Schemes in Wireless Multihop Networks," *ACM Computing Surveys*, vol. 47, no. 3, 2015.

[27] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder, "Efficient Interference-Aware TDMA Link Scheduling for Static Wireless Networks," in *International Conference on Mobile Computing and Networking*, 2006.

[28] S. Gobriel, D. Mosse, and R. Cleric, "TDMA-ASAP: Sensor Network TDMA Scheduling with Adaptive Slot-Stealing and Parallelism," in *IEEE International Conference on Distributed Computing Systems*, 2009.

[29] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A Centralized Scheduling Algorithm for IEEE 802.15.4e TSCH based Industrial Low Power Wireless Networks," in *IEEE Wireless Communications and Networking Conference*, 2016.

[30] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3655–3666, 2013.

[31] A. Tinka, T. Watteyne, and K. Pister, "A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping," in *Ad Hoc Networks*, 2010.

[32] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, 2016.

[33] A. Elsts, S. Kim, H. Kim, and C. Kim, "An Empirical Survey of Autonomous Scheduling Methods for TSCH," *IEEE Access*, vol. 8, pp. 67 147–67 165, 2020.

[34] A. Elsts, X. Fafoutis, J. Pope, G. Oikonomou, R. Piechocki, and I. Craddock, "Scheduling High-Rate Unpredictable Traffic in IEEE 802.15.4 TSCH Networks," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2017.

[35] S. Oh, D. Hwang, K.-H. Kim, and K. Kim, "Escalator: An Autonomous Scheduling Scheme for Convergecast in TSCH," *Sensors*, vol. 18, no. 4, 2018.

[36] J. Shi, M. Sha, and Z. Yang, "DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," in *International Conference on Distributed Computing Systems*, 2018.

[37] ——, "Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1669–1682, 2019.

[38] S. Jeong, J. Paek, H. Kim, and S. Bahk, "TESLA: Traffic-Aware Elastic Slotframe Adjustment in TSCH Networks," *IEEE Access*, vol. 7, pp. 130 468–130 483, 2019.

[39] J. Jung, D. Kim, J. Hong, J. Kang, and Y. Yi, "Parameterized slot scheduling for adaptive and autonomous TSCH networks," in *IEEE Conference on Computer Communications Workshops*, 2018.

[40] S. Rekik, N. Baccour, M. Jmaiel, K. Drira, and L. A. Grieco, "Autonomous and traffic-aware scheduling for TSCH networks," *Computer Networks*, vol. 135, pp. 201–212, 2018.

[41] "IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams," *IEEE Std 802.15.9-2016*, pp. 1–74, 2016.

[42] M. K. Hanawal, D. N. Nguyen, and M. Krunz, "Jamming attack on in-band full-duplex communications: Detection and countermeasures," in *IEEE Conference on Computer Communications*, 2016.

[43] Y. Zou, D. Yu, L. Wu, J. Yu, Y. Wu, Q. Hua, and F. C. M. Lau, "Fast Distributed Backbone Construction Despite Strong Adversarial Jamming," in *IEEE Conference on Computer Communications*, 2019.

[44] A. H. Sodhro, S. Pirbhulal, G. H. Sodhro, A. Gurtov, M. Muzammal, and Z. Luo, "A Joint Transmission Power Control and Duty-Cycle Approach for Smart Healthcare System," *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8479–8486, 2019.

[45] A. Rajakaruna, A. Manzoor, P. Porambage, M. Liyanage, M. Ylianttila, and A. Gurtov, "Enabling End-to-End Secure Connectivity for Low-Power IoT Devices with UAVs," in *IEEE Wireless Communications and Networking Conference Workshop*, 2019.

[46] L. Zhang, Z. Guan, and T. Melodia, "Cooperative anti-jamming for infrastructure-less wireless networks with stochastic relaying," in *IEEE Conference on Computer Communications*, 2014.

[47] X. Cheng, J. Shi, and M. Sha, "Cracking Channel Hopping Sequences and Graph Routes in Industrial TSCH Networks," *ACM Trans. Internet Technol.*, vol. 20, no. 3, 2020.
[48] X. Cheng, J. Shi, M. Sha, and L. Guo, "Launching Smart Selective Jamming Attacks in WirelessHART Networks," in *IEEE Conference on Computer Communications*, 2021.
[49] X. Cheng, J. Shi, and M. Sha, "Cracking the Channel Hopping Sequences in IEEE 802.15.4e-Based Industrial TSCH Networks," in *International Conference on Internet of Things Design and Implementation*, 2019.