Enabling Direct Message Dissemination in Industrial Wireless Networks via Cross-Technology Communication

Di Mu, Yitian Chen, Xingjian Chen, Junyang Shi
Department of Computer Science
State University of New York at Binghamton
Email: {dmu1, cyitian1, xchen218, jshi28}@binghamton.edu

Mo Sha*
Knight Foundation School of Computing and Information Sciences
Florida International University
Email: msha@fiu.edu

***Abstract*—IEEE 802.15.4 based industrial wireless networks have been widely deployed to connect sensors, actuators, and gateway in industrial facilities. Although wireless mesh networks work satisfactorily most of the time thanks to years of research, they are often complex and difficult to manage once the networks are deployed. Moreover, the deliveries of time-critical messages suffer long delay, because all messages have to go through hop-by-hop transport. Recent studies show that adding a low-power wide-area network (LPWAN) radio to each device in the network can effectively overcome such limitations, because network management and time-critical messages can be transmitted from gateway to field devices directly through long-distance LPWAN links. However, industry practitioners have shown a marked reluctance to embrace the new solution because of the high cost of hardware modification. This paper presents a novel system, namely *DIrect MEssage dissemination (DIME)* system, that leverages the cross-technology communication technique to enable the direct message dissemination from gateway to field devices in industrial wireless networks without the need to add a second radio to each field device. Experimental results show that our system effectively reduces the latency of delivering time-critical messages and improves network reliability compared to a state-of-the-art baseline.**

***Index Terms*—Industrial Wireless Network, IEEE 802.15.4, Message Dissemination, Cross-Technology Communication, LoRa, Transmission Scheduling**

I. Introduction

The Internet of Things (IoT) refers to a broad vision whereby everyday objects, places, and environments are interconnected via the Internet [1]. Until recently, most of the IoT infrastructure and application development by businesses have focused on smart homes and wearables. However, it is the production and manufacturing segment of the IoT, which underlies the Fourth Industrial Revolution (or Industry 4.0 [2]), that promises one of the largest potential economic effects of IoT [3] – up to $47 trillion in added value globally by 2025, according to the McKinsey report on future disruptive technologies [4]. Industrial networks that serve industrial IoT typically connect field devices (sensors and actuators) and gateway in industrial facilities, such as manufacturing plants, steel mills, oil refineries. Industrial applications pose unique challenges to networking due to 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. The stringent demand has been traditionally met by specifically chosen wired solutions, such as the highway addressable remote transducer (HART) communication protocol [5], where cables connect sensors and forward sensor readings to the gateway where a controller sends commands to actuators. However, wired networks are often costly to deploy and maintain in industrial environments and difficult to reconfigure to accommodate new production requirements.

Over the past decade, IEEE 802.15.4 based wireless networks have been widely adopted to replace wired networks in industrial facilities. Battery-powered wireless modules are used to easily and inexpensively retrofit existing sensors and actuators without the need to run cables for communication and power. To meet the stringent reliability and real-time requirements, the industrial wireless standards, such as WirelessHART [6], ISA100 [7], WIA-FA [8], and 6TiSCH [9], make a set of specific design choices, such as employing the time slotted channel hopping (TSCH) technology, which distinguish themselves from traditional wireless sensor networks (WSNs) designed for best effort services [10]. A decade of real-world deployments of those standards has demonstrated the feasibility to achieve reliable wireless communication in industrial facilities. Although wireless mesh networks work satisfactorily in industrial facilities most of the time thanks to years of research, they are often complex and difficult to manage once the networks are deployed. Moreover, the deliveries of time-critical messages, especially those carrying urgent information such as emergency alarms, suffer long delay, because all messages have to go through hop-by-hop transport. Recent studies show that adding a low-power wide-area network (LPWAN) radio (e.g., a LoRa [11] radio) to each device in the network effectively overcomes such limitations, because messages and time synchronization beacons can be transmitted from gateway to field devices directly through long-distance LPWAN links [12], [13]. However, industry practitioners have shown a marked reluctance to embrace the new solution because of the high cost of hardware modifica-

* Corresponding author

tion.

To address the issue, we develop a novel system, namely *DIrect MEssage dissemination (DIME)* system, that leverages the cross-technology communication (CTC) from LoRa to IEEE 802.15.4 devices [14], [15] to enable the direct message dissemination from gateway to field devices in industrial wireless networks without the need to add a second radio to each field device. To our knowledge, this paper represents the first networking solution that takes advantage of CTC to reduce the network management complexity and the latency of delivering time-critical messages in industrial wireless networks. Specifically, we make the following contributions:

- We develop a new TSCH slotframe structure, which enables direct message dissemination using CTC;
- We develop a novel autonomous transmission scheduling method that runs on top of the routing protocol for low-power and lossy networks (RPL) on each device and schedule transmissions without the need to handshake with neighboring devices;
- We implement DIME and evaluate it on a physical testbed with 40 devices. Experimental results show that DIME significantly improves end-to-end reliability and reduces end-to-end latency compared to a state-of-the-art baseline.

The remainder of this paper is organized as follows. Section II introduces the background of TSCH, RPL, and CTC. Section III presents our DIME system. Section IV discusses our testbed and evaluation of DIME. Section V reviews the related work. Section VI concludes the paper.

II. Background

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

A. TSCH

TSCH [16] was introduced in 2012 as an amendment (IEEE 802.15.4e) to the medium access control (MAC) portion of the IEEE 802.15.4 standard. It is designed to provide time-deterministic packet deliveries for industrial process control and automation. TSCH divides time into timeslots (or slots) for time-deterministic media access and enables channel hopping in every timeslot for resilience against interference. Under TSCH, the gateway and all field devices in the network must keep globally time synchronized to support time slotted access. Starting from the gateway, each synchronized device shares its time information to its neighboring devices by periodically broadcasting enhanced beacons (EBs). Contained in each EB, the absolute slot number (ASN) indicates the number of timeslots passed since the network starts. ASN is set to zero when the network starts and increased by one at the end of each timeslot. A fixed number of timeslots are grouped into a slotframe, which repeats over time. This fixed number is defined as the slotframe length L_{SF}. A timeslot in a slotframe is specified by the time offset t_o, the channel offset c_o, and the type of operation (e.g., transmission, reception, or sleep). The time offset t_o of a timeslot is represented by the modulo operation between ASN and L_{SF}:

$$t_o = mod(ASN, L_{SF}) \tag{1}$$

A sequence of channels used for channel hopping, called frequency hopping sequence (FHS), is known by all devices. The frequency channel used in a timeslot is determined by FHS, ASN, the channel offset (c_o), and the length of FHS (L_{FHS}):

$$channel = FHS(mod(ASN + c_o, L_{FHS})) \tag{2}$$

A TSCH schedule generated by a transmission scheduling algorithm determines each device's operation type and channel offset in every timeslot.

B. RPL

RPL is a destination oriented distance-vector routing protocol designed to support multi-hop routing on the resource-constrained field devices with IPv6 compatibility. RPL uses a destination-oriented directed acyclic graph (DODAG) rooted at the gateway (border router device). Each field device in the network computes its rank based on a cost function, such as the accumulated expected transmission counts (ETXs) from itself to the gateway. DODAG information object (DIO) messages are broadcasted in the network to exchange the routing information between neighboring devices, which allows each field device to update its rank and preferred parent device. A destination advertisement object (DAO) message is sent to the selected parent device from the child device. By exchanging DIO and DAO messages, each device sets up its downward and upward routes in the DODAG.

C. CTC

The CTC techniques are designed to enable direct communication between two heterogeneous wireless devices that run different physical layers but in the same frequency band. As a key technology that enhances wireless coexistence in the 2.4 GHz frequency band, CTC has been explored in recent research to enable the direct communication between WiFi and IEEE 802.15.4 devices [17]–[25], between WiFi and Bluetooth devices [26], [27], and between Bluetooth and IEEE 802.15.4 devices [28], [29]. The emerging LPWAN technologies, such as LoRa, have emerged to provide low-power long distance wireless connections. To leverage the long communication range of LPWANs, several CTC techniques have been proposed to enable direct messaging from LoRa to IEEE 802.15.4 device [14], [15], [30], [31]. For example, under the CTC technique developed by Shi et al. [14], [15], an IEEE 802.15.4 radio detects the radio-frequency (RF) energy patterns transmitted from a LoRa radio in the 2.4 GHz frequency band and decodes the information carried by those patterns. Shi et al. identified 454 distinguishable RF energy patterns, which can be used to encode information in a $15ms$ timeslot.

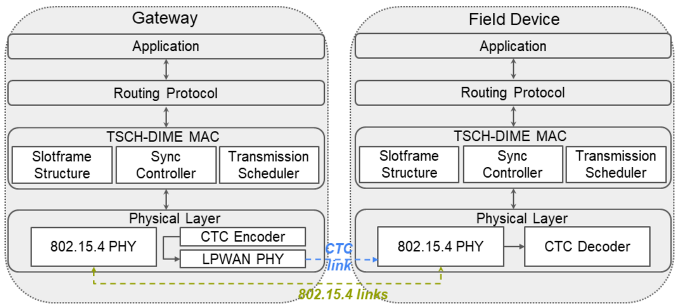


Fig. 1. System architecture of DIME.

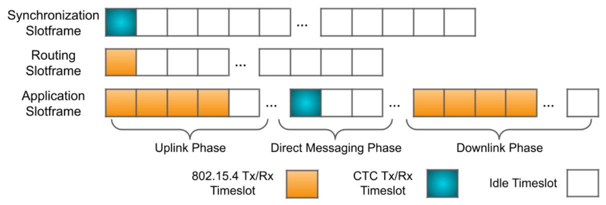


Fig. 2. Slotframe structure.

III. Our Design of DIME

In this section, we first present an overview of DIME and then introduce the designs of our slotframe structure, CTC technique, synchronization controller, and transmission scheduler in detail.

A. System Overview

The primary design goals of DIME are to ensure timely deliveries of time-critical messages from gateway, reduce network management complexity, and improve network reliability. To achieve such goals, the network that runs DIME allows its gateway to directly disseminate time-critical messages to field devices through long distance CTC links, leverage CTC packets to synchronize the clocks of all devices in the network, and deliver important messages through both CTC and hop-by-hop transport. Figure 1 shows the system architecture of DIME. The gateway is equipped with two radios: an IEEE 802.15.4 radio and a LPWAN radio and each field device is equipped with an IEEE 802.15.4 radio. DIME adopts the IEEE 802.15.4 and LPWAN physical layers to handle the interactions between DIME and its underlying radio hardware. *CTC Encoder* that runs on the gateway and *CTC Decoder* that runs on the field device create the CTC link between them. DIME extends TSCH, namely *TSCH-DIME MAC*, by adding *Slotframe Structure*, *Synchronization Controller*, and *Transmission Scheduler*. *Slotframe Structure* specifies three types of slotframes for time synchronization, routing, and application traffic. *Synchronization Controller* manages the CTC-based time synchronization process on each device. *Transmission Scheduler* that runs on each device is responsible for determining the device's action and communication channel in each timeslot. The application and the routing protocol (e.g., RPL) run on top of TSCH-DIME MAC. The detailed design of each component will be presented next.

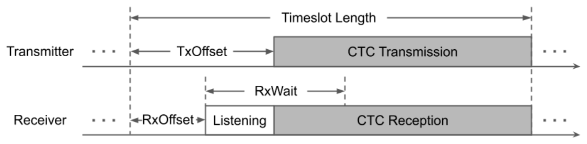


Fig. 3. Timeslot in direct messaging phase.

B. Slotframe Structure

We follow the suggestions made by Duquennoy et al. [32] and define three types of slotframes: one for time synchronization traffic, one for routing traffic, and one for application traffic. Figure 2 shows the slotframe structure. The first timeslot of each synchronization slotframe is reserved for the gateway to broadcast a CTC beacon. All field devices are active in that timeslot and use the beacon to synchronize their clocks (see Section III-D). The first timeslot of each routing slotframe is shared among the gateway and all field devices to exchange the packets generated by the routing protocol. Each application slotframe consists of three phases: uplink phase, direct messaging phase, and downlink phase. The timeslots in the uplink phase are reserved for the field devices to forward the packets generated by the source devices to the gateway and the timeslots in the downlink phase are reserved for the field devices to forward the packets generated by the gateway to the destination devices. One or more timeslots in the direct messaging phase are reserved for the gateway to send direct messages to the destination devices through the long distance CTC links.

Figure 3 shows the timeslot in the direct messaging phase. The CTC transmission starts at $TxOffset$, which is the time offset computed according to the on-air time duration of the CTC transmission to make sure that the CTC transmission finishes at the exact end of that timeslot. $TxOffset$ must be no less than the recommended value in TSCH (e.g., $3ms$ for a $15ms$ timeslot) to make CTC transmissions tolerable to the clock drifts on field devices. The receiver starts to listen to the channel at $RxOffset$ and looks for the start of a CTC message during $RxWait$. If the receiver cannot detect any CTC signals during $RxWait$, it turns off the radio to save energy; otherwise, it keeps the radio on to receive and decode the CTC message. We employ a coding method with a set of distinguishable CTC symbols, which can be encoded in a single timeslot (see Section III-C).

C. CTC

DIME is not tied up with any CTC techniques. Our implementation adopts a state-of-the-art technique developed by Shi et al. [14], which supports CTC from LoRa to IEEE 802.15.4 device in the 2.4 GHz frequency band and employs a coding method with 454 distinguishable CTC symbols. DIME reserves eight symbols for CTC beacons and the rest for application messages. All CTC symbols can fit into a single $15ms$ timeslot. Under such a CTC technique, the LoRa transmitter generates radio energy patterns by tuning the packet payload length and the time to transmit packets. The IEEE 802.15.4 device receiver samples the received signal strength (RSS) values to recognize the radio energy patterns. The LoRa packets with 15 different payload lengths can generate the on-air transmission time ranging from $3.705ms$ to $14.405ms$ within a $15ms$ timeslot. Depending on the variable payload length, one or more (up to four) LoRa packets can be transmitted within a timeslot. Non-transmitting time can be selectively inserted between two consecutive packets to create more energy patterns. A unique CTC symbol is created by the combined energy pattern of the transmitted LoRa packet(s) in a timeslot and the optional non-transmitting time between the packets. Our IEEE 802.15.4 devices can continuously sample the RSS with the interval of $0.088ms$, which is enough to distinguish different LoRa packet lengths and the presence of non-transmitting time.

D. Synchronization Controller

Synchronization Controller that runs on each field device is responsible for using the periodically broadcasted CTC beacons to synchronize its clock. As discussed in Section II, TSCH uses EBs for time synchronization. DIME replaces EBs with CTC beacons, which are used to time synchronize all devices and accommodate the new device when it joins the network. Similar to EBs, the CTC beacons contain time information and are broadcasted periodically. Unlike EBs that are flooded by all field devices in the network, the CTC beacons are only transmitted by the gateway, which significantly reduces time synchronization overhead. All field devices use the gateway as their time source and use the CTC beacons to keep them time synchronized. Each CTC beacon is transmitted in the first timeslot of each synchronization slotframe. When the transmission of a CTC beacon is finished, all field devices use the time when the CTC signal disappears to synchronize their local clocks. Each CTC beacon carries the synchronization period number (SPN), which can be used to calculate the absolute time since the network starts. SPN can be encoded in one CTC beacon or multiple consecutive synchronization slotframes, where the CTC beacon in the last synchronization slotframe contains a special CTC symbol that indicates the end of SPN encoding. SPN is initialized as 0 and incremented by 1 after each time that it has been encoded. Assuming there are N_{sym} CTC symbols reserved for the CTC beacons, each SPN is encoded in N_{SF} synchronization slotframes, the length of a synchronization slotframe is L_{SF}^{sync} timeslots, and the duration of a timeslot is T_s, the maximum time T_{max} that can be represented by SPN can be calculated as

$$T_{max} = T_s \times L_{SF}^{sync} \times N_{SF} \times (N_{sym} - 1)^{(N_{SF}-1)}. \quad (3)$$

For example, if $N_{sym} = 8$, $N_{SF} = 10$, $L_{SF}^{sync} = 397$, and $T_s = 15ms$, then T_{max} is more than 76 years. Based on the received SPN, the field devices can calculate the current ASN by

$$ASN = SPN \times L_{SF}^{sync} \times N_{SF}. \quad (4)$$

There are two ways for a new device to join the network. It either joins the network when it starts or after the network starts. When the network starts to operate, the gateway continuously broadcasts CTC beacons for a fixed period of time (e.g., $30s$), which allows all field devices to set their CTC physical-layer parameters (e.g., the RSS threshold for CTC signal detection) based on link measurements. If a field device joins the network when the network has already started, it uses the default CTC physical-layer parameters and looks for the CTC beacons, which allows it to derive the complete SPN and ASN. Please note that there is no need for the field devices that join the network when it starts to derive SPN, because the ASN starts from 0.

E. Transmission Scheduler

Transmission Scheduler that runs on each device is responsible for generating the operation schedule for that device. It first determines whether the device transmits, receives, or sleeps in each timeslot of every slotframe and generates a operation schedule for each slotframe. It then combines all schedules into a single one for runtime execution. The priority assigned to each slotframe determines whether the schedule of that slotframe should yield to another when a scheduling conflict appears during the schedule combination process. The lengths of all slotframes are three mutually prime numbers to minimize the schedule conflicts. We will discuss the priority assignments and transmission scheduling next.

Synchronization Slotframe has the highest priority because the network cannot operate without time synchronization. The only active timeslot scheduled in the synchronization slotframe is the first one, where the time offset t_o is 0 and the operation channel is fixed to the best one available.

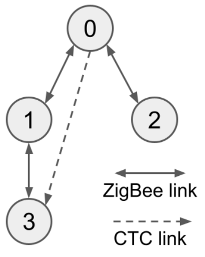
Routing Slotframe has the priority lower than Synchronization Slotframe's. The first timeslot of each routing slotframe is scheduled to exchange routing information, where the time offset t_o is 0 and the channel offset c_o is 1.

Application Slotframe has the lowest priority. As Figure 2 shows, the application slotframe has three phases: uplink phase, direct messaging phase, and downlink phase.

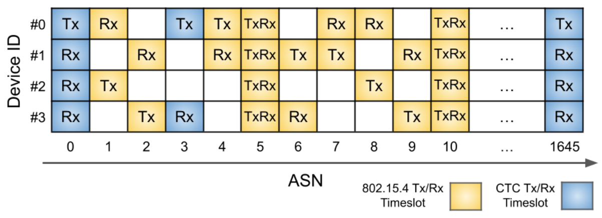
To provide contention-free transmissions, Transmission Scheduler assigns a sender-based dedicated (SBD) timeslot in the application slotframe's uplink phase to each field device to forward the application traffic generated by the source devices to the gateway. Similarly, Transmission Scheduler assigns a receiver-based dedicated (RBD) timeslot in the application slotframe's downlink phase to each field device to forward the application traffic generated by the gateway to the destination devices. In the uplink phase, each field device receives incoming packets during its child devices' SBD timeslots and forwards them during its own SBD timeslot. The time offset of each SBD timeslot t_o^{SBD} is calculated by performing the modulo operation between the sender device's ID $txID$ ($txID \geqslant 1$) and the length of the uplink phase L_{UP}:

$$t_o^{SBD}(txID) = mod(txID - 1, L_{UP}) \quad (5)$$

Following Eq. 5, each device can calculate each timeslot for packet transmission using its own device ID and the



(a) Network topology.



Device ID	0	1	2	3	4	5	6	7	8	9	10	...	1645
#0	Tx	Rx		Tx	Tx	TxRx		Rx	Rx		TxRx	...	Tx
#1	Rx		Rx		Rx	TxRx	Tx	Tx		Rx	TxRx	...	Rx
#2	Rx	Tx				TxRx			Tx		TxRx	...	Rx
#3	Rx		Tx	Rx		TxRx	Rx			Tx	TxRx	...	Rx

(b) Generated transmission schedule.

Fig. 4. An example of transmission schedule generated based on a given network topology.

ones for packet receptions using its child devices' IDs. The direct messaging phase begins after the uplink phase. The gateway groups the messages for all destination devices into L_{CTC} CTC transmission(s), where L_{CTC} is the length of the direct messaging phase. The time offset of the first CTC transmission t_o^{CTC} equals L_{UP} because it immediately follows the uplink phase. All destination devices listen to the CTC transmission(s) in the direct messaging phase and decode the CTC messages. In the downlink phase, each device on every downlink path (the path from the gateway to the destination device) listens during its RBD timeslot and forwards downlink traffic to its child device along the downlink path until reaching the destination device. The time offset of each RBD timeslot t_o^{RBD} is calculated by the receiver device's ID $rxID$ ($rxID \geqslant 1$), L_{UP}, L_{CTC}, and the length of the downlink phase L_{DP}:

$$t_o^{RBD}(rxID) = L_{UP} + L_{CTC} + mod(rxID - 1, L_{DP}) \quad (6)$$

Following Eq. 6, each device along the downlink path can calculate the timeslot for packet reception using its own device's ID and the one for packet transmission using its child device's ID. The channel offset c_o of the application slotframe is set to 2. To ensure contention-free transmissions in both uplink and downlink phases, both L_{UP} and L_{DP} must be no less than the number of field devices in the network. We choose $L_{UP} = L_{DP} = 50$, $L_{CTC} = 1$, and $L_{SF}^{app} = L_{UP} + L_{CTC} + L_{DP} = 101$ as the default lengths in our implementation for 40 field devices.

Figure 4(b) shows an example schedule generated based on the network topology shown in Figure 4(a). In this network, #0 is the gateway device, #2 is a source device, and #3 acts as both source and destination device. In this example, the lengths of synchronization, routing, and application slotframes are 47, 5, and 7, respectively. The length of application slotframe is the summation of $L_{UP} = 3$, $L_{CTC} = 1$, and $L_{DP} = 3$. The first application slotframe begins when ASN is 0 and the second one begins when ASN is 7. However, the timeslots are preempted by the synchronization and routing slotframes when ASN is 0, 5, and 10 in the first two application slotframes. When ASN is 0, all field devices (#1, #2, and #3) listens to the CTC synchronization beacon broadcasted by the gateway, because the synchronization slotframe has the highest priority and is active in that timeslot. The uplink phase of the application slotframe is active when ASN is 1 and 2, in which #2 and #3 transmit uplink application packets to their parent devices in the senders' SBD timeslots. The direct messaging phase of the application timeslot is active when ASN is 3, in which #3 listens to the CTC direct message from the gateway. The downlink phase of the application slotframe is active when ASN is 4 and 6, in which #0 and #1 transmit downlink application packets to their child devices on the downlink path in the receivers' RBD timeslots. When ASN is 5 and 10, the routing slotframe is active and preempts the application slotframe to allow all devices in the network exchange routing messages. The second application slotframe begins when ASN is 7 with uplink transmissions. The entire schedule repeats every $47 \times 5 \times 7 = 1645$ timeslots.

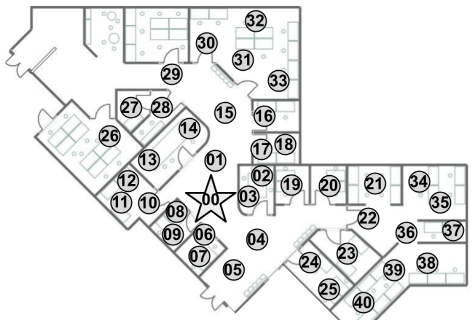
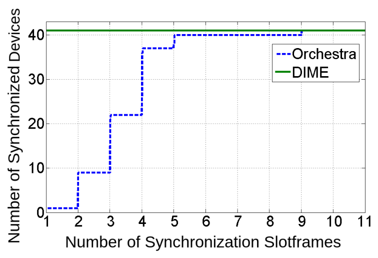


Fig. 5. Testbed deployment. Device 00 is the gateway marked as a star.

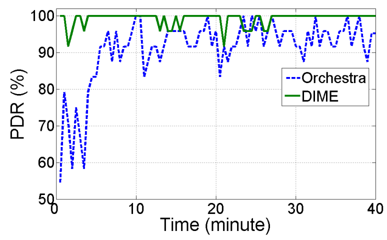
IV. Evaluation

To evaluate the performance of DIME, we have implemented DIME under Contiki OS [33], integrated it with RPL under storing mode, and performed a series of experiments on our testbed. We first perform microbenchmark experiments to examine the network initialization process when the network starts to operate with DIME and then test DIME's resilience to network changes. We then measure the performance of DIME when it runs on the networks with various device densities. Finally, we perform two 24-hour measurements to evaluate the performance of DIME when the devices generate data at a high rate and a low rate, respectively. We compare DIME against Orchestra [32], [34], one of the state-of-the-art networking solutions designed for TSCH based industrial wireless networks.

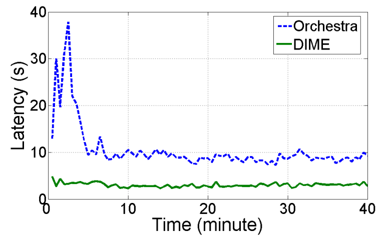
Figure 5 shows the device deployment of our testbed, which consists of one gateway and 40 field devices deployed throughout 22 office and lab areas in an office environment [35]. The gateway is a TelosB mote [36] integrated with a WiMOD iM282A LoRa module [37] and each field device is a TelosB



(a) Number of synchronized devices over time.



(b) End-to-end PDR.



(c) End-to-end latency.

Fig. 6. Network performance during network initialization.

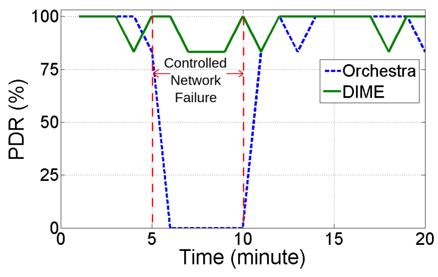
TABLE I
THE SOURCE AND DESTINATION DEVICE IDS OF EACH DATA FLOW.

Data Flow Number	Source Device ID	Destination Device ID
1	11	12
2	13	14
3	15	30
4	18	16
5	20	21
6	22	36
7	24	25
8	09	05

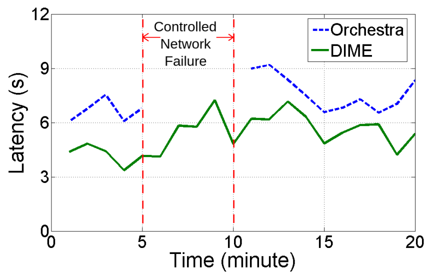
mote. We run a monitor-control application with eight data flows, each of which has a source device and a destination device. Table I lists the source and destination devices of each data flow. The source device of a data flow generates an uplink packet periodically based on the data rate specified by the application. Each uplink packet is then forwarded to the gateway where a controller generates and sends a downlink packet to its corresponding destination device. We precompute eight downlink paths and use them at runtime due to the hardware limitation of TelosB motes. We configure DIME and Orchestra with three slotframes: synchronization slotframe (397 timeslots), routing slotframe (31 timeslots), and application slotframe (101 timeslots). The length of each timeslot is $15ms$. To ensure a fair comparison, we use the same slotframe lengths and settings in DIME and Orchestra.

A. *Network Initialization*

To examine the network initialization process, we record the number of synchronized devices in each synchronization slotframe and measure the end-to-end packet delivery ratio (PDR) and the end-to-end latency after the network starts to operate with DIME. We configure each source device to generate an uplink packet every $10s$. Figure 6(a) shows the number of synchronized devices in each synchronization slotframe after the network starts to operate. As Figure 6(a) shows, DIME successfully synchronizes all 40 devices in the first synchronization slotframe ($5.96s$) after the network starts to operate, while Orchestra takes nine synchronization slotframes ($53.60s$) to synchronize all devices due to the need of flooding synchronization beacons across the entire network. Figure 6(b) and 6(c) plot the network performance during the network initialization process, where each data point shows the averaged value in a $30s$ time window. As Figure 6(b) and 6(c) show, DIME helps the network stabilize its performance quickly and consistently provides PDRs higher than 91.7% and latency less than $4.84s$. As a comparison, Orchestra takes more than five minutes to stabilize the network performance and provide PDRs higher than 90%. More importantly, DIME consistently provides higher network reliability and shorter latency over Orchestra by employing the CTC links, which will be evaluated more extensively in Section IV-C and IV-D.



(a) End-to-end PDR.

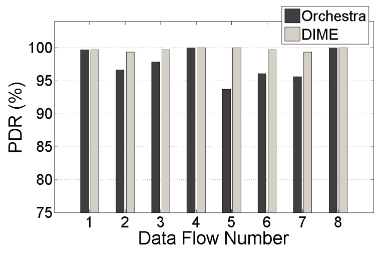


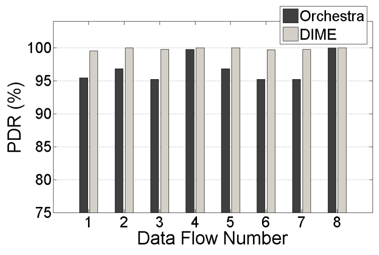
(b) End-to-end latency.

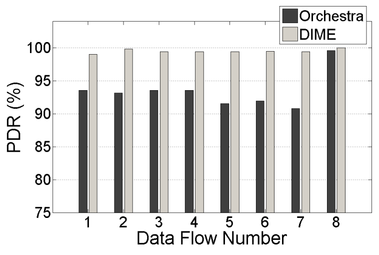
Fig. 7. Network performance of Data Flow 1 when we disable Device 8 for five minutes (time between two red dashed lines).

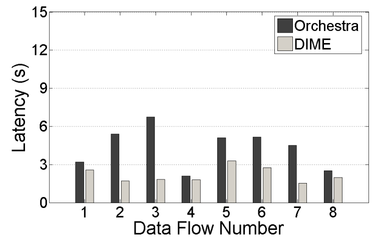
B. *Adaptation to Network Changes*

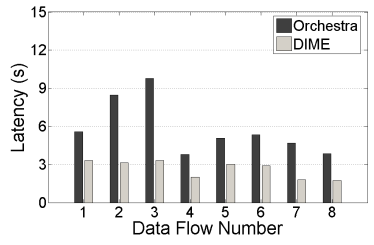
To test DIME's resilience to network changes, we mimic devices failures by disabling different devices in the network and measure network performance changes. For example, Figure 7 plots the network performance of Data Flow 1 when we disable Device 8 after the network operates with five minutes and enable it after another five minutes. As Figure 7(a) shows, the network that runs either DIME or Orchestra provides PDR larger than 83% before the failure occurs and after the network recovers. When the failure occurs, the network that runs DIME can still provide high PDRs


(a) End-to-end PDR at low device density.


(b) End-to-end PDR at medium device density.


(c) End-to-end PDR at high device density.


(d) End-to-end latency at low device density.


(e) End-to-end latency at medium device density.

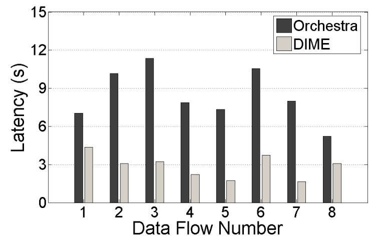
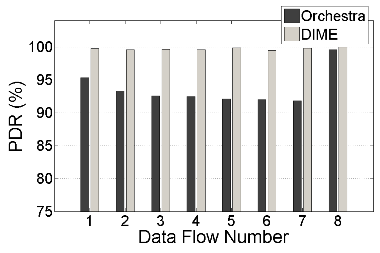
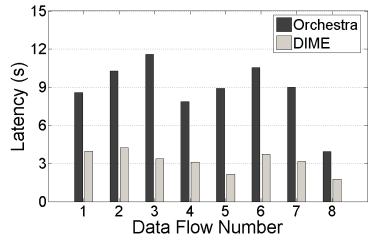

(f) End-to-end latency at high device density.

Fig. 8. Performance of each data flow when the network with different device densities runs DIME and Orchestra.


(a) End-to-end PDR.


(b) End-to-end latency.

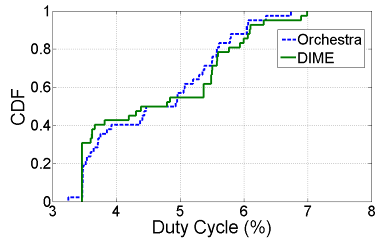

(c) Device duty cycle.

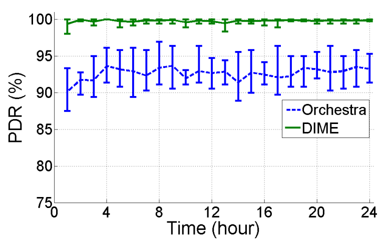
Fig. 9. Averaged network performance of each data flow when source devices generate data at high rate.

thanks to its CTC links, while the PDRs of the network that runs Orchestra drop to 0% due to the disconnection of the downlink path. As Figure 7(b) shows, DIME helps the network keep the end-to-end latency shorter than 7.3s even when the network faces device failure. We observe similar results when the failures occur at other devices. The results demonstrates DIME's resilience to network changes.

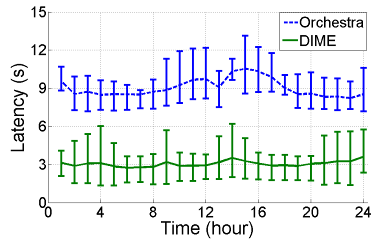
C. Network Performance under Various Device Densities

To evaluate the performance of DIME, we create three networks by randomly disabling different numbers of devices on our testbed and measure the performance of each network for an hour. Specifically, we enable all 40 field devices on our testbed to create the network with high device density and use 34 field devices to create the network with medium device density. We only enable 22 field devices to create the network with low device density. Figure 8 plots the end-to-end PDR and the end-to-end latency of each data flow when the network runs DIME and Orchestra, respectively. DIME consistently outperforms Orchestra. For example, as Figure 8(a) and 8(d) plot, the network with low device density achieves PDRs higher than 99.6% and latency shorter than 3.3s on all data flows when it runs DIME. As a comparison, the network reaches the lowest PDR of 93.7% on Data Flow 5 and the longest latency of 6.7s on Data Flow 3 when it runs Orchestra. In the network with medium device density, DIME consistently helps the network achieve PDRs higher than 99.4% and latency shorter than 4.0s on all data flows, while Orchestra has the lowest PDR of 93.6% and the longest latency of 9.6s, as Figure 8(b) and Figure 8(e) show. Similarly, in the network with high device density, DIME consistently helps the network achieve PDRs higher than 99.0% and latency shorter than 4.4s on all data flows, while Orchestra has the lowest PDR of 90.8% and the longest latency of 11.4s, as Figure 8(c) and Figure 8(f) show.

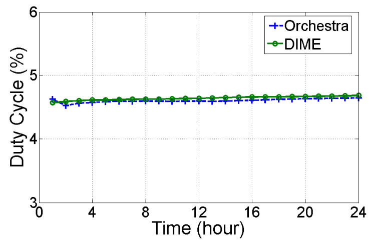
We also observe that the improvements provided by DIME increase with network density. For example, Orchestra experiences decreased PDRs from 100% to 93.5% and increased latency from 2.1s to 7.9s on Data Flow 4 when the device density increases from low to high, while DIME only has minor changes on PDRs from 100% to 99.4% and latency from 1.8s to 2.2s. This is because the higher device density increases network management complexity, resulting in more packet losses and larger latency under Orchestra. On the other hand, DIME can consistently provide good network performance with the help of CTC links.



(a) End-to-end PDR.

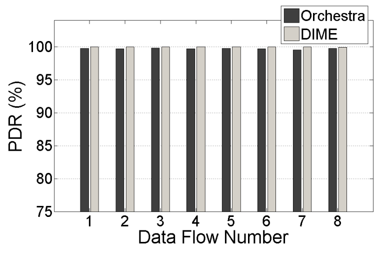


(b) End-to-end latency.

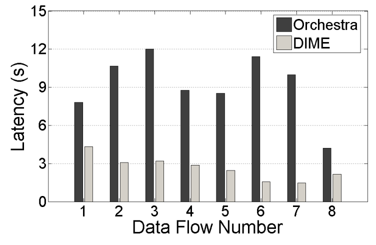


(c) Device duty Cycle.

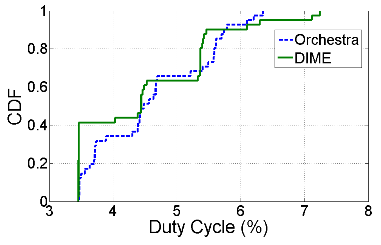
Fig. 10. Network performance over 24 hours when source devices generate data at high rate. Each data point is calculated within a one-hour time window.



(a) End-to-end PDR.

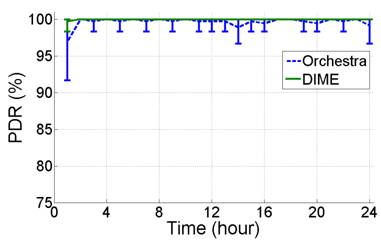


(b) End-to-end latency.

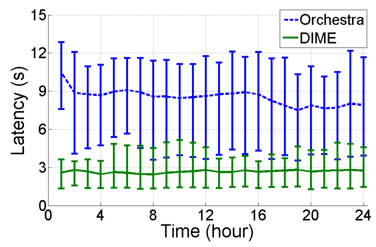


(c) Device duty cycle.

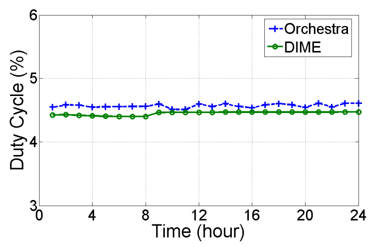
Fig. 11. Averaged network performance of each data flow when source devices generate data at low rate.



(a) End-to-end PDR.



(b) End-to-end latency.



(c) Device duty cycle.

Fig. 12. Network performance over 24 hours when source devices generate data at low rate. Each data point is calculated within a one-hour time window.

D. 24-Hour Performance under Different Data Rates

Finally, we perform two 24-hour measurements to evaluate DIME's performance when the devices generate data at a high rate and a low rate, respectively. To evaluate DIME's performance at high data rates, we configure each source device to periodically generate an uplink packet every $10s$. Figure 9 plots the averaged network performance of each data flow. As Figure 9(a) shows, the network that runs DIME achieves PDRs higher than 99.5% on all data flows, consistently outperforming the one that runs Orchestra with the lowest PDR of 91.8%. Figure 9(b) plots the averaged end-to-end latency of each data flow. As Figure 9(b) shows, the network that runs DIME achieves latency always shorter than $4.23s$, which is significantly shorter the latency when the network runs Orchestra. For instance, the averaged end-to-end latency of Data Flow 3 is $3.36s$ under DIME and $11.58s$ under Orchestra. We also measure the device duty cycle[1] of each device when it runs DIME and Orchestra. Figure 9(c) plots the CDF of the device duty cycles of all field devices under DIME and Orchestra, where the duty cycles are similar between two solution. The device duty cycles range from 3.46% to 6.78% under DIME and vary from 3.26% to 6.75% under Orchestra. The median device duty cycle is 4.61% under DIME and 4.69% under Orchestra. These results show that DIME improves network reliability and reduces latency without consuming more energy.

[1]The device duty cycle is defined as the percentage of timeslots in which the radio is active. This metric is an indicator of energy consumption.

Figure 10 shows the network performance over the 24-hour period. DIME consistently outperforms Orchestra at the high data rate. As Figure 10(a) shows, the end-to-end PDRs are always higher than 99.3% under DIME and vary between 90.2% and 93.7% under Orchestra. As Figure 10(b) shows, the end-to-end latency are always shorter than $3.61s$ under DIME and varies between $8.23s$ and $10.54s$ under Orchestra. As Figure 10(c) shows, the field devices experience similar duty cycles under both solutions over 24 hours. On average, DIME improves the end-to-end PDR from 92.6% to 99.8% and reduces the end-to-end latency from $9.03s$ to $3.06s$ compared to Orchestra. More importantly, the variations of PDRs under

DIME are much smaller than the ones under Orchestra, which represents a significant advantage in industrial applications that demand consistent high reliability in harsh industrial facilities.

To evaluate DIME's performance when the network operates at the low data rate, we increase the data generation period from $10s$ to $60s$ and repeat our experiments. Figure 11 plots our measurements. As Figure 11(a) shows, the network that runs DIME has the averaged PDRs of all data flows higher than 99.93% and the network that runs Orchestra achieves comparable performance (higher than 99.52%). Figure 11(b) shows the averaged end-to-end latency of each data flow. As Figure 11(a) shows, DIME reduces the latency by $6.53s$ on average compared to Orchestra. For instance, the averaged end-to-end latency of Data Flow 6 is $1.57s$ under DIME and $11.42s$ under Orchestra. Figure 11(c) plots the CDF of the device duty cycles of all field devices under DIME and Orchestra. Both solutions provide similar device duty cycles. The median duty cycle is 4.44% under DIME and 4.48% under Orchestra. The results confirms that DIME improves network reliability and reduces latency without consuming more energy.

Figure 12 shows the network performance when the network runs DIME and Orchestra, respectively, over the 24-hour period. As Figure 12(a) shows, DIME achieves the PDR of 99.7% during the first hour and always delivers 100% PDR after that, while Orchestra experiences the lowest PDR of 96.9% in the first hour and achieves the PDRs higher than 98.8% after that. As Figure 12(b) shows, DIME always keeps the latency less than $2.82s$, while Orchestra experiences the latency up to $10.45s$. As Figure 12(c) shows, the field devices experience similar duty cycles under both solutions over 24 hours.These results show that DIME consistently outperforms Orchestra at the low data rate.

V. Related Work

In recent years, many centralized and decentralized transmission scheduling methods have been developed for TSCH-based wireless networks. The centralized method relies on a global scheduler to determine the transmission schedule of each device in the network. For instance, Jin et al. propose a centralized multi-hop scheduling method that allocates more resources to more vulnerable links [38] and Palattella et al. develop a traffic-aware scheduling algorithm, which reduces energy consumption of each device by allocating the minimum number of active slots to it [39]. The centralized scheduling methods suffer poor scalability and low flexibility once the networks are deployed. The decentralized scheduling methods have been proposed to overcome those drawbacks. Under a decentralized scheduling method, each device generates its own transmission schedules after exchanging information with its neighboring devices at runtime. For example, Municio et al. develop a decentralized broadcast-based scheduling algorithm for dense multi-hop TSCH networks, which uses selective broadcasting to exchange transmission schedules of neighboring devices [40] and Aijaz et al. designs a decentralized adaptive multi-hop scheduling protocol, which is traffic-aware and adaptive to topology changes [41]. The decentralized scheduling methods improve network scalability and flexibility at the cost of introducing significant signaling overhead due to the need of information exchange between neighboring devices. To eliminate such overhead, research efforts in the recent years have produced autonomous scheduling methods including Orchestra, DiGS, ALICE, and ATRIA. Orchestra allows network devices to generate their transmission schedules autonomously based on their local RPL routing information [32]. DiGS is a distributed graph routing and autonomous scheduling solution that allows network devices to compute their own transmission schedules based on their graph routes [42]. ALICE is an autonomous link-based cell scheduling scheme, which allocates a unique cell for each directional link by using the local information in the routing layer [43]. ATRIA is an autonomous traffic-aware transmission scheduling method, in which each device detects its traffic load based on its local routing information and then schedules its transmissions accordingly [44]. Although the autonomous scheduling methods effectively eliminate the signaling overhead, they still suffer long latency when delivering network control messages (e.g., time synchronization) and time-critical application messages (e.g., emergency alarms) because all messages have to go through hop-by-hop transport.

Emerging LPWAN technologies, such as LoRa, offer new opportunities to overcome the limitations of multi-hop wireless networks by adding long-distance links. Recent studies have combined LPWAN radios with IEEE 802.15.4 radios to form hybrid wireless networks. For example, Gu et al. propose one-hop out-of-band control planes that use LoRa links to deliver control messages directly from gateway to field devices [12], [13] and Singh et al. develop a hybrid network architecture that integrates LoRa and IEEE 802.15.4 radios for oil pipeline monitoring applications [45]. However, adding new radios to all devices in a network requires significant effort and cost because of the need of hardware modification. Industry practitioners therefore have shown a marked reluctance to embrace such solutions. To address this issue, our solution leverages the CTC technique from LoRa to IEEE 802.15.4 devices to provide long-distance message deliveries and does not require any hardware modification on field devices.

Recently, many techniques have been developed to enable the CTC between heterogeneous wireless devices that operate in the same frequency band. Significant efforts have been made to enable the CTC between WiFi and IEEE 802.15.4 devices [17]–[25], between WiFi and Bluetooth devices [26], [27], and between Bluetooth and IEEE 802.15.4 devices [28], [29], and between LoRa and IEEE 802.15.4 devices [14], [15], [30], [31]. Our solution adopts the CTC technique developed by Shi et al. [14], [15] to enable long-distance communication in the 2.4 GHz band.

VI. Conclusion

A decade of real-world deployments of industrial standards has demonstrated the feasibility to achieve reliable wireless communication in industrial environments. Although wireless mesh networks work satisfactorily in industrial facilities most of the time thanks to years of research, they are often complex and difficult to manage once the networks are deployed. Moreover, the deliveries of time-critical messages, especially those carrying urgent information such as emergency alarms, suffer long delay, because all messages have to go through the hop-by-hop transport. To address the issue, we develop DIME, which consists of a new TSCH slotframe structure for direct message dissemination and time synchronization using CTC and a new autonomous transmission scheduling method that schedules transmissions without the need to handshake with neighboring devices. We implement DIME and evaluate it on a physical testbed with 40 devices. Experimental results show that DIME significantly improves end-to-end reliability and reduces end-to-end latency compared to a state-of-the-art baseline.

Acknowledgment

This work was supported in part by the National Science Foundation under grant CNS-1657275 and CNS-2150010.

References

[1] M. E. Porter and J. E. Heppelmann, "How Smart, Connected Products are Transforming Competition," *Harvard Business Review*, vol. 92, no. 11, 2014.

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

[3] A. Thierer and A. Castillo. Projecting the Growth and Economic Impact of the Internet of Things. [Online]. Available: https://www.mercatus.org/publications/technology-and-innovation/projecting-growth-and-economic-impact-internet-things

[4] 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. [Online]. Available: http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/disruptive-technologies

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

[6] WirelessHART. [Online]. Available: https://fieldcommgroup.org/technologies/hart/hart-technology

[7] ISA 100. [Online]. Available: http://www.isa100wci.org/

[8] WIA-FA. [Online]. Available: https://webstore.iec.ch/publication/32718

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

[10] 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*, vol. 104, no. 5, 2016.

[11] LoRa. [Online]. Available: https://lora-alliance.org

[12] C. Gu, R. Tan, X. Lou, and D. Niyato, "One-Hop Out-of-Band Control Planes for Low-Power Multi-Hop Wireless Networks," in *INFOCOM*, 2018.

[13] C. Gu, R. Tan, and X. Lou, "One-Hop Out-of-Band Control Planes for Multi-Hop Wireless Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 15, no. 4, 2019.

[14] J. Shi, X. Chen, and M. Sha, "Enabling Direct Messaging from LoRa to ZigBee in the 2.4 GHz Band for Industrial Wireless Networks," in *ICII*, 2019.

[15] ——, "Enabling Cross-Technology Communication from LoRa to ZigBee in the 2.4 GHz Band," *ACM Transactions on Sensor Networks*, vol. 18, no. 2, 2021.

[16] IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7554

[17] W. Wang, X. Liu, Y. Yao, and T. Zhu, "Exploiting WiFi AP for Simultaneous Data Dissemination among WiFi and ZigBee Devices," in *ICNP*, 2021.

[18] X. Guo, Y. He, X. Zheng, L. Yu, and O. Gnawali, "ZigFi: Harnessing Channel State Information for Cross-Technology Communication," *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, 2020.

[19] X. Guo, Y. He, and X. Zheng, "WiZig: Cross-Technology Energy Communication Over a Noisy Channel," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, 2020.

[20] S. Wang, S. M. Kim, and T. He, "Symbol-Level Cross-Technology Communication via Payload Encoding," in *ICDCS*, 2018.

[21] S. Wang, Z. Yin, Z. Li, and T. He, "Networking Support for Physical-Layer Cross-Technology Communication," in *ICNP*, 2018.

[22] X. Zheng, Y. He, and X. Guo, "StripComm: Interference-Resilient Cross-Technology Communication in Coexisting Environments," in *INFOCOM*, 2018.

[23] Z. Yin, W. Jiang, S. M. Kim, and T. He, "C-Morse: Cross-Technology Communication with Transparent Morse Coding," in *INFOCOM*, 2017.

[24] Z. Li and T. He, "WEBee: Physical-Layer Cross-Technology Communication via Emulation," in *MobiCom*, 2017.

[25] Z. Chi, Y. Li, H. Sun, Y. Yao, Z. Lu, and T. Zhu, "B2W2: N-Way Concurrent Communication for IoT Devices," in *ACM Conference on Embedded Network Sensor Systems*, 2016.

[26] X. Guo, Y. He, X. Zheng, Z. Yu, and Y. Liu, "LEGO-Fi: Transmitter-Transparent CTC with Cross-Demapping," *IEEE Internet of Things Journal*, vol. 8, no. 8, 2021.

[27] Z. Li and Y. Chen, "BlueFi: Physical-Layer Cross-Technology Communication from Bluetooth to WiFi," in *ICDCS*, 2020.

[28] W. Jiang, Z. Yin, R. Liu, Z. Li, S. M. Kim, and T. He, "Boosting the Bitrate of Cross-Technology Communication on Commodity IoT Devices," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, 2019.

[29] W. Jiang, S. M. Kim, Z. Li, and T. He, "Achieving Receiver-Side Cross-Technology Communication with Cross-Decoding," in *International Conference on Mobile Computing and Networking*, 2018.

[30] J. Shi, D. Mu, and M. Sha, "Enabling Cross-technology Communication from LoRa to ZigBee via Payload Encoding in Sub-1 GHz Bands," *ACM Transactions on Sensor Networks*, vol. 18, no. 1, 2021.

[31] ——, "LoRaBee: Cross-Technology Communication from LoRa to ZigBee via Payload Encoding," in *ICNP*, 2019.

[32] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks through Autonomously Scheduled TSCH," in *SenSys*, 2015.

[33] Contiki OS. [Online]. Available: https://github.com/contiki-os/contiki

[34] Orchestra Source Code. [Online]. Available: https://github.com/simonduq/orchestra

[35] "Wireless Embedded System Testbed at the State University of New York at Binghamton." [Online]. Available: https://users.cs.fiu.edu/~msha/testbed.htm

[36] TelosB Mote Platform. [Online]. Available: https://www.cs.albany.edu/~jhh/Resources/TelosB_Datasheet.pdf

[37] WiMOD iM282A Datasheet. [Online]. Available: https://wireless-solutions.de/wp-content/uploads/2019/02/iM282A_Datasheet_V1_0.pdf

[38] 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 *WCNC*, 2016.

[39] 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 IEEE 802.15.4e TSCH," *IEEE Sensors Journal*, vol. 13, no. 10, 2013.

[40] E. Municio and S. Latré, "Decentralized Broadcast-Based Scheduling for Dense Multi-Hop TSCH Networks," in *MobiArch*, 2016.

[41] A. Aijaz and U. Raza, "DeAMON: A Decentralized Adaptive Multi-Hop Scheduling Protocol for 6TiSCH Wireless Networks," *IEEE Sensors Journal*, vol. 17, no. 20, 2017.

[42] J. Shi, M. Sha, and Z. Yang, "DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," in *ICDCS*, 2018.

[43] S. Kim, H.-S. Kim, and C. Kim, "ALICE: Autonomous Link-based Cell Scheduling for TSCH," in *IPSN*, 2019.

[44] X. Cheng and M. Sha, "ATRIA: Autonomous Traffic-Aware Scheduling for Industrial Wireless Sensor-Actuator Networks," in *ICNP*, 2021.

[45] R. Singh, M. Baz, C. Narayana, M. Rashid, A. Gehlot, S. V. Akram, S. S. Alshamrani, D. Prashar, and A. S. AlGhamdi, "Zigbee and Long-Range Architecture Based Monitoring System for Oil Pipeline Monitoring with the Internet of Things," *Sustainability*, vol. 13, no. 18, 2021.