A Distributed Real-time Scheduling System for Industrial Wireless Networks

VENKATA P. MODEKURTHY, University of Nevada Las Vegas
ABUSAYEED SAIFULLAH, Wayne State University
SANJAY MADRIA, Missouri University of Science and Technology

The concept of Industry 4.0 introduces the unification of industrial Internet-of-Things (IoT), cyber physical systems, and data-driven business modeling to improve production efficiency of the factories. To ensure high production efficiency, Industry 4.0 requires industrial IoT to be adaptable, scalable, real-time, and reliable. Recent successful industrial wireless standards such as WirelessHART appeared as a feasible approach for such industrial IoT. For reliable and real-time communication in highly unreliable environments, they adopt a high degree of redundancy. While a high degree of redundancy is crucial to real-time control, it causes a huge waste of energy, bandwidth, and time under a centralized approach and are therefore less suitable for scalability and handling network dynamics. To address these challenges, we propose DistributedHART—a distributed real-time scheduling system for WirelessHART networks. The essence of our approach is to adopt local (node-level) scheduling through a time window allocation among the nodes that allows each node to schedule its transmissions using a real-time scheduling policy locally and online. DistributedHART obviates the need of creating and disseminating a central global schedule in our approach, thereby significantly reducing resource usage and enhancing the scalability. To our knowledge, it is the first distributed real-time multi-channel scheduler for WirelessHART. We have implemented DistributedHART and experimented on a 130-node testbed. Our testbed experiments as well as simulations show at least 85% less energy consumption in DistributedHART compared to existing centralized approach while ensuring similar schedulability.

CCS Concepts: • **Computer systems organization → Sensors and actuators**; **Sensor networks**; **Real-time system architecture**; • **Networks → Sensor networks;**

Additional Key Words and Phrases: WirelessHART, real-time networking, distributed scheduling

ACM Reference format:
Venkata P. Modekurthy, Abusayeed Saifullah, and Sanjay Madria. 2021. A Distributed Real-time Scheduling System for Industrial Wireless Networks. *ACM Trans. Embedd. Comput. Syst.* 20, 5, Article 46 (July 2021), 28 pages.
https://doi.org/10.1145/3464429

Modekurthy and Saifullah are co-first authors. This note is missing.

This work was supported by NSF through grants CAREER-1846126, CNS-2006467, and CNS-1461914.

Authors' addresses: V. P. Modekurthy (corresponding author), University of Nevada Las Vegas, 4247 SEB, 4505 S Maryland Pkwy, Las Vegas, NV, 89154; email: prashant.modekurthy@unlv.edu; A. Saifullah (corresponding author), Wayne State University, 5057 Woodward Ave, Suite# 14110.2, Detroit, MI 48202; email: saifullah@wayne.edu; S. Madria, Missouri University of Science and Technology, 325A Computer Science Building, 500 W. 15th Street Rolla, Mo. 65409; email: madrias@mst.edu.

1539-9087/2021/07-ART46 $15.00
https://doi.org/10.1145/3464429

1 INTRODUCTION

The concept of Industry 4.0 introduces the unification of industrial Internet-of-Things, cyber physical systems, and data-driven business modeling to improve production efficiency of the factories [37]. To ensure high production efficiency, Industry 4.0 requires industrial Internet-of-Things to be adaptable, scalable, real-time, and reliable. Recent successful industrial wireless standards such as WirelessHART have shown their feasibility as a cost-efficient, real-time, and robust approach for industrial Internet-of-Things [23].

To make reliable and real-time communication in highly unreliable wireless environments, WirelessHART adopts a high degree of redundancy using a **Time Division Multiple Access (TDMA)**-based **Media Access Control (MAC)** protocol. A time slot can be either *dedicated* (where at most one transmission is scheduled to a receiver) or *shared* (where multiple nodes may contend to send to a common receiver). To handle transmission failures, each node on a path from a sensor to an actuator is assigned two dedicated time slots and a third shared slot on a separate path for retransmission [2]. A network manager creates the transmission schedule **centrally** and in advance for all nodes and then disseminates them. A centralized WirelessHART scheduler with high redundancy raises several practical challenges in achieving scalability as described below.

High level of redundancy in centralized algorithms [27, 29] causes a huge waste of time and bandwidth, and hence is not scalable. For example, if the transmission of a packet along a particular link succeeds, then all time slots (on the current link and redundant links) that were assigned to handle its failure remain unused. Similarly, if it fails along that particular link, then all time slots that were assigned for its subsequent links to handle a successful transmission remain unused. Our experiments observed up to 70% unused time slots in WirelessHART networks (see Section 4). In industrial IoT, safety-critical events or emergencies can occur unpredictably or aperiodically. For example, a WirelessHART network in an oil refinery may suddenly detect a safety valve displacement requiring immediate attention to avoid accidents. Existing solution handles emergencies by allocating time slots in the centrally created schedule and by stealing slots in the absence of emergencies [17]. However, this approach leaves most of the slots of the periodic server unstolen, and hence unused. Thus, the network remains largely underutilized, which affects the scalability of the system.

Schedule dissemination in centralized algorithm consumes bandwidth, energy, and time, even for a smaller network or workload. Typically, hyper-period and length of the schedule increase exponentially with the increase in the number of flows or their periods, hindering network scalability. Note that in general, periods can be non-harmonic to ensure stability or control performance [28]. Furthermore, the mobility of nodes introduces discernible issues for a central scheduler due to the frequent changes to the network topology. In an industrial environment, moving objects such as robotic arms or carts can affect link quality of nodes and change the topology of the network. Such frequent changes to the topology require frequent computation and re-dissemination of schedules. Nonetheless, the data-driven business model in Industry 4.0 introduces frequent changes to sampling rates, which also requires re-configuration and re-dissemination of schedules. Frequent re-dissemination of the schedule consumes high energy, time, and bandwidth. Thus, fully centralized scheduling is less suitable for industrial Internet-of-Things. Besides, it is typically suitable for deterministic traffic patterns (like periodic traffic) arising from stationary nodes.

To address the above limitations, in this article, we propose a distributed real-time scheduling system for WirelessHART networks. Designing a distributed TDMA protocol with scheduling performance close to a centralized one is highly challenging, as the former has to achieve this without global knowledge. For a WirelessHART network, a distributed TDMA protocol also has to

incorporate dedicated and shared slots in local scheduling. We address these challenges by proposing DistributedHART. We make the following contributions in the article:

- We propose DistributedHART, the first ***distributed*** real-time multi-channel scheduling for WirelessHART networks. DistributedHART adopts local (node-level) scheduling through a time window allocation among the nodes that allows each node to schedule its transmissions locally and online. Thus, DistributedHART can handle any communication pattern (periodic or aperiodic) and any length of schedule. It obviates the need for creating and disseminating a global schedule.
- We provide a schedulability test for DistributedHART that can be used to determine if all the packets in the network meet their real-time deadlines (or not) with a high probability. We evaluated the performance of the schedulability test against that observed in simulation. We observed that the performance of the schedulability test was close to that of the simulation.
- We have implemented DistributedHART in TinyOS [1] for TelosB [3] platform and performed experiments on a 130-node physical indoor testbed [4] to show the effectiveness of DistributedHART under scalability. We also evaluated DistributedHART through simulations under scalability, different workloads, and workload dynamics on TOSSIM [16] using the topology of another testbed [34]. In both experiments and simulations, we observe at least 85% less energy consumption in DistributedHART compared to existing centralized approach.

The rest of the article is organized as follows: Section 2 reviews related work. Section 3 describes the model. Sections 4 and 5 present the design and delay analysis of DistributedHART, respectively, for a uniform time window length. Section 6 presents non-uniform time window allocation for DistributedHART and its delay analysis. Section 7 presents latency performance of DistributedHART. Sections 8, 9, and 10 present experiments, simulations, and conclusion, respectively.

2 RELATED WORK

CSMA/CA-based real-time scheduling has been studied in References [14, 39]. In contrast, WirelessHART adopts a TDMA-based MAC to achieve predictable latency bounds. TDMA-based real-time scheduling without multi-channel communication or multi-path graph routing was studied in References [12, 21, 24, 38, 47]. Real-time routing was studied in References [5, 22, 41, 42]. For WirelessHART networks, priority assignment [31], channel assignment [13, 32], and security vulnerabilities [7] were studied recently. Schedulability analysis for industrial wireless networks was studied in References [20, 27, 30, 33]. These works do not focus on the real-time scheduling. Existing work in Reference [29] showed that the real-time scheduling for flows in WirelessHART networks is NP-hard and proposed real-time scheduling policies for WirelessHART. A flexible retransmission policy for WirelessHART networks was proposed in Reference [6]. Scheduling under multiple co-existing wirelessHART networks was studied in Reference [15]. Mobility-aware real-time scheduling of packets was studied in Reference [8]. These papers adopt a fully centralized scheduler that creates a schedule in advance, and they employ high degree of redundancy as specified in the WirelessHART standard. Such an approach causes a huge waste of time, bandwidth, energy, and memory, making it less suitable for dynamics and scalability. In this article, we aim to address these limitations and propose an online and distributed real-time scheduling system for WirelessHART.

Orchestra [10], D^2-PaS [43–45], and DiGS [35, 36] are the recent distributed scheduling approaches for a multi-hop wireless network. However, they have the following limitations: First, they only consider a single channel protocol while WirelessHART uses multiple channels. Second, they do not consider shared slots while WirelessHART adopts graph routing with both

dedicated and shared slot transmissions. In Orchestra and DiGS, the end-to-end communication latency of a flow is in the order of the number of nodes in the network. Such a high latency is less suitable for real-time communications. Due to these limitations, Orchestra, D^2-PaS, and DiGS are less suitable for WirelessHART. In contrast, DistributedHART is a practical scheduling system for WirelessHART that considers multichannel and graph routing, which are highly critical for wireless control applications in unreliable environments and is not limited to sparse traffic.

3 BACKGROUND AND SYSTEM MODEL

WirelessHART operates in the 2.4 GHz band and is built based on the physical layer of IEEE 802.15.4. WirelessHART network is a multi-hop mesh topology of nodes—field devices, multiple access points, and a Gateway. The *field devices* are wirelessly networked sensors and actuators. Each node contains a *half-duplex* omnidirectional radio transceiver that cannot transmit and receive a packet simultaneously and receive from at most one sender at a time. *Access points* provide redundant paths between the wireless network and the Gateway. The *network manager* and the controller remain at the *Gateway*. The network employs feedback control loops between sensors and actuators. Sensors measure process variables and deliver them to a controller via the multi-hop mesh network. The controller disseminates control commands to the actuators. Each actuator applies its control commands to adjust the physical processes accordingly. We use the term *flow* to denote an end-to-end wireless control loop between a sensor and an actuator through the controller.

WirelessHART adopts a multi-channel TDMA protocol, where multiple transmissions are scheduled on different channels within the same time slot. In wide-area deployment, two distant nodes that do not interfere with each other can use the same channel and time slot for a transmission, i.e., we allow spatial re-use of channels. Time in the network is globally synchronized. A receiver acknowledges each transmission from a sender. Both a transmission and its acknowledgment happen in a 10 ms time slot. A time slot can be dedicated for a receiver and a sender or shared between multiple senders and a receiver. In a *dedicated slot*, only one sender is allowed to transmit to a receiver along a link. In a *shared slot*, multiple senders can attempt to send to a common receiver. To mitigate collisions in a shared slot, a WirelessHART network adopts the random back-off policy.

For enhanced reliability, the network adopts *graph routing* [2]. A *routing graph* is a directed list of loop-free paths between a source and a destination. Each node in a routing graph has at least two neighbors that enable redundant paths to a destination. Graph routing allows to schedule a packet using multiple channels on multiple time slots to deliver a packet through multiple paths, thereby ensuring high reliability in highly unreliable environments. A routing graph consists of an uplink graph and multiple downlink graphs. An uplink graph connects all sensors to controllers, while a downlink graph connects a controller to an actuator. We generate routes using distributed graph routing [22]. However, DistributedHART works with any graph routing algorithm.

We consider the system has n real-time flows denoted by $F = \{F_1, F_2, \ldots, F_n\}$. The period and deadline of a flow F_i are denoted by T_i and D_i ($D_i \leq T_i$), respectively. Our system is applicable to fixed or dynamic priority assignment, and priority can be based on deadlines, periods, or criticalities.

Here, we present an outline of the current centralized flow-based scheduling approach adopted in WirelessHART networks. In the centralized flow-based scheduling, the network manager pre-allocates dedicated (or shared) time slots for each link on the graph route of each flow. Specifically, for scheduling a packet of one flow in the uplink graph, the network manager allocates two dedicated slots along each link on the graph route's primary path starting from the source. The second dedicated slot is used for retransmissions when transmission fails on the first dedicated slot. To handle transmission failures on both time slots, the network manager allocates a third shared slot

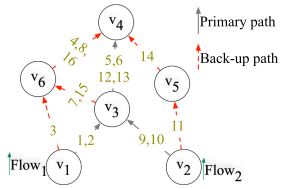


Fig. 1. Example of scheduling in WirelessHART.

on a separate path. The links in the downlink graph are scheduled similarly. The network manager creates a global schedule in advance, which is split into superframes. A *superframe* is a repeating schedule representing the communication pattern of a set of nodes. Typically, the length of a superframe is equal to the hyperperiod or least common multiple of the periods of all flows. Upon generating a schedule, the network manager disseminates the superframe to all nodes.

An example of centralized scheduling for two flows $Flow_1$ and $Flow_2$ originating from node v_1 and v_2, respectively, and terminating at access point v_4 is shown in Figure 1. The period and deadline for the two flows is 16 time slots. In Figure 1, solid gray line and dashed red line represent the primary path and a back-up path for flow, respectively. Labels on the lines denote the time slot allocation. According to the WirelessHART standard, each link on the primary path is allocated two dedicated time slots. For $Flow_1$, primary path links $v_1 \rightarrow v_3$ and $v_3 \rightarrow v_4$ are allocated dedicated time slots 1,2 and 5,6, respectively, and shared path links $v_1 \rightarrow v_6$, $v_3 \rightarrow v_6$, and $v_6 \rightarrow v_4$ are allocated shared time slots 3, 7, and 16, respectively. Similarly, for $Flow_2$, primary path links $v_2 \rightarrow v_3$ and $v_3 \rightarrow v_4$ are allocated time slots 9,10 and 12,13, respectively, and shared path links $v_2 \rightarrow v_5$, $v_5 \rightarrow v_4$, $v_3 \rightarrow v_6$, and $v_6 \rightarrow v_4$ are allocated time slots 11, 14, 15, and 16, respectively. Here, the superframe length is 16 time slots, i.e., the schedule repeats after every 16 time slots.

In this work, our objective is to develop a real-time distributed scheduling system where each node can locally schedule its transmissions. Generating routes is not the focus of this article.

4 THE DESIGN OF DISTRIBUTEDHART

In the existing centralized scheduling approaches, the network is largely underutilized. For example, in Figure 1, if the packets from both flows $Flow_1$ and $Flow_2$ are successful in the first attempt, then 4 time slots (namely, 1, 5, 9, and 10) are used out of 16 pre-allocated slots, leaving 75% of the slots unusable under good network conditions. In an experiment conducted on 30 flows on a testbed of 69 nodes, work in Reference [27] observed that 70% of the slots were unused for a flow in a run. Although redundancy is crucial for handling worst-case scenarios in real-time control, such scheduling with high redundancy causes a huge waste of energy, bandwidth, time, and memory (to store schedule) and is less suitable for network/workload dynamics and scalability. Furthermore, disseminating large schedules (with lengths equal to the hyperperiod of all flows) can cause long delays and consume high energy. To address these issues, we propose DistributedHART, which offers a distributed scheduling system for WirelessHART networks. We describe the design of DistributedHART below.

4.1 Overview of Distributed Scheduling in DistributedHART

The essence of our approach is to enable local and online scheduling at the nodes. To do so, in DistributedHART, we propose to assign time windows (which consists of multiple time slots) to nodes instead of every link on each flow. In this section, for the sake of simplicity in explanation, we use a uniform time window selection where each node selects w time slots. Extension of the proposed approach for non-uniform time windows is discussed in Section 6. During a node's

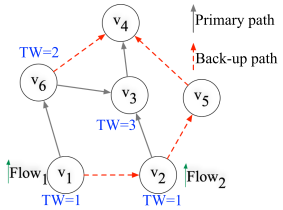


Fig. 2. DistributedHART's time window allocation example.

transmission time window, it locally selects and transmits an available packet, from its queue, based on a real-time scheduling policy. The local scheduling of packets (within a window) obviates the need for creating and distributing a schedule in advance. Thus, DistributedHART consumes less energy, memory, and bandwidth (even under frequent dynamics). Furthermore, within its transmission time window, a node locally and dynamically chooses the number of retransmission attempts required by a packet. If a packet transmission is successful on the first attempt, then it does not need any retransmission attempts. The node can use the rest of the time slots in its window to transmit other packets in its queue. If a packet transmission is not successful in the first attempt, then a node can attempt to successfully transmit a packet in at least two other time slots. If a node's queue is empty, then it can use more than two time slots to retransmit the packet. Thus, DistributedHART can provide better utilization of time slots and reliability than the WirelessHART standard.

In DistributedHART, transmission time windows repeat after a fixed interval. We define epoch as the time duration after which the transmission window schedule repeats. Note that superframe represents the repeating schedule in a centralized flow-based scheduler and is not related to epoch. Considering nodes pick γ unique transmission time windows with w slots each, the length of one epoch in a network is given by γw. An example of time window allocation along the primary path in DistributedHART is shown in Figure 2, where time windows 1, 1, 2, and 3 are assigned to nodes v_1, v_2, v_6, and v_3, respectively. For this time window selection, $w = 2$ and $\gamma = 3$, and hence, the epoch length is 6. During time slot 1, v_1 and v_2 can transmit a packet to node v_3 and v_6. During time slot 3, node v_6 can transmit a packet to node v_3. Time slots 2 and 4 can be used for packet retransmission in case of failures. During time slot 5, node v_3 has 2 packets in its queue, and v_3 can use any real-time scheduling policy to determine the next packet to transmit.

To minimize collisions in the network, DistributedHART generates a conflict-free channel and transmission time window allocation. We consider that a set of transmissions on the same channel is ***conflict-free*** if the **Signal-to-Noise plus Interference Ratio (SNIR)** of all receivers exceeds a threshold. In such a model, we say that two nodes a and b are conflict-free if both receivers can successfully receive a packet. Similarly, we say that two nodes a and b are ***in conflict*** if and only if simultaneous transmissions from a and b cause radio interference at a receiver. To minimize such conflicts, each node first collects an interference model of the network using SNIR such as the RID protocol [46]. Using the interference model, each node performs a receiver-based channel allocation based on vertex coloring proposed in Reference [26]. After channel allocation, each node performs time window allocation using distributed vertex coloring. Note that the time window allocation removes all conflict within the network.

In DistributedHART, nodes execute distributed channel and time window allocation during network initialization and under some network dynamics (e.g., when routes are affected) and some workload dynamics (e.g., when a change in time window length is required to ensure schedulability). Other network or workload dynamics (e.g., that does not affect routes) will not trigger these algorithms, keeping the overhead of DistributedHART low.

After the time window allocation, each node is aware of its transmission time windows and all its neighbors' transmission time windows. Nodes listen to their neighbors' time windows to receive a packet. Since a node does not precisely know when a packet will arrive, the energy overhead for listening to a packet is higher than that of a centralized scheduler. However, compared to the energy required to broadcast an entire schedule by a centralized schedule, this energy overhead is small. In our approach, nodes can still often sleep during the neighboring nodes' transmit windows. There are many cases where a node u certainly knows that there will be no more packets from a particular neighbor v and does not need to keep listening for v's transmission. For example, node u knows the periods of the packets that it receives from v (as period values are embedded inside a packet) and thus after receiving a packet from v it knows the earliest time when the next packet of the same flow may arrive. Therefore, exploiting such earliest times of packet generation, a node may often determine a time window within which it is guaranteed that v will not transmit to it, and can sleep if appropriate.

Note that DistributedHART is a novel scheduling system for WirelessHART. DistributedHART proposes local and online scheduling of packets, where a node locally decides which packet to transmit within its transmission time window rather than a central manager. To generate nodes' transmission time window, DistributedHART uses existing algorithms such as RID (to generate a conflict graph) and DRAND (to perform vertex coloring on the conflict graph). Although these algorithms are important to the working for DistributedHART, they individually do not provide a real-time and reliable scheduling policy for industrial wireless networks provided by DistributedHART. Furthermore, DistributedHART is compatible with any distributed conflict graph generation methods and distributed vertex coloring algorithms for the generation of transmission time windows. We describe channel and time window allocation, local scheduling policy, and online scheduling as a dedicated and shared slot in the following sections.

4.2 Channel and Time Window Allocation in DistributedHART

To generate a conflict-free transmission time window and channel allocation, nodes first develop an interference model with their neighbors. Nodes use the local interference model to generate two types of conflict graphs: a receiver conflict graph and a transmission conflict graph. We define a *receiver conflict graph* as a graph (over all nodes) in which two nodes are connected by an edge if and only if a packet transmission to one node interferes with the other. In a *transmitter conflict graph*, two transmitters have an edge if simultaneous transmissions by both transmitters can lead to a collision at one of the intended recipients. On the receiver conflict graph, nodes perform a receiver-based channel allocation. Since a channel allocation does not necessarily solve all conflicts, to remove all conflicts, nodes then perform time window allocation on a transmitter conflict graph. This section describes these steps in detail.

Conflict Graph Generation. To generate a local interference model, nodes use RID protocol [46]. RID protocol generates a SNIR model between a node and all other nodes in its interfering region. Note that, unlike the communication region, an interfering region can be up to 2-hop from the node. Each node uses the SNIR information to generate a list of interfering nodes. It then broadcasts this list of interfering nodes within the local neighborhood of 5-hop such that all nodes can identify its edges in receiver and transmitter conflict graphs. To broadcast the information within a local neighborhood, nodes can use CSMA/CA-based broadcasts with a **time-to-live (TTL)** field set as 5, similar to controlled flooding. Typically, two nodes with an edge in a transmission conflict graph can be 3-hop away from each other, since the maximum distance between the transmitting node and its receiver is 1-hop, and the interfering node and the receiver is 2-hop (coming from the RID protocol). Similarly, two nodes with an edge in a channel conflict graph can be 2-hop away from each other, since the maximum distance between the common interferer and the pair of nodes is 2.

Thus, nodes can use broadcasts in a 3-hop neighborhood to enable the generation of transmission and receiver conflict graphs. To ensure the reliability of the broadcasts, nodes can use a time-to-live of 5-hop. Broadcasting the information to a 5-hop neighborhood increases the chances of nodes receiving the same information at least once from one of their neighbors.

Overhead of Conflict Graph Generation on DistributedHART. Both WirelessHART and DistributedHART network assume each node maintains a physical interference of the network to identify neighbors and generate routes. The overhead of DistributedHART is the local broadcast of the physical interference model such that nodes can generate a local transmitter and receiver conflict graphs. In contrast, in WirelessHART, the physical interference model is collected by the central manager. Typically, for large networks, the local broadcast of interference model can be less energy-consuming than the collection of interference model by a central manager.

Receiver-based Channel Allocation. In a receiver-based channel allocation, all nodes that receive a packet are assigned a channel. Neighboring nodes transmit a packet to a node on its allocated channel. An optimal channel allocation that maximizes the number of conflict-free simultaneous transmissions is known to be NP-Hard [11]. To generate a channel assignment, we use DRAND [26], a distributed vertex coloring approach, on a receiver conflict graph. Since the number of channels is fixed, DRAND might not resolve all conflicts, i.e., two conflicting nodes can select the same channel for reception. These conflicts are removed through time window allocation.

Time Window Allocation. To remove all transmission conflicts, each node on the primary path of a graph route (a transmitting node) performs a time window allocation. During the time window allocation, nodes select non-conflicting transmission time windows. During its time window, a node transmits packets on the receiver's channel without interfering with (or being interfered by) other packet transmissions. To allocate time windows, we first represent all remaining conflicts using a transmitter conflict graph. We use distributed vertex coloring using DRAND [26] on the transmitter conflict graph to compute non-conflicting transmission windows at each transmitter node. During DRAND execution, each node selects the smallest time window number (color) from a list of available time windows. Note that there is no upper bound on the number of available transmission time windows, unlike channel assignment. During DRAND, nodes try to minimize the number of transmission time windows to decrease latency and facilitate scalability.

Handling Dynamics: An industrial wireless network can be highly unpredictable, with frequent changes to the networks. This section presents a high-level idea for handling network dynamics. During sustained network dynamics, a node observing a change in link qualities uses management and maintenance cycles to initiate, update, and exchange the physical interference model. Nodes use RID protocol to initiate and update the physical interference model. This model is exchanged within a 5-hop neighborhood using CSMA/CA MAC protocol during the management and maintenance cycles. In case of collisions, nodes randomly back off and re-transmit in the subsequent management cycles. Note that the update and exchange of the physical interference model can take several management cycles, depending on the number of link quality changes.

Upon exchanging the physical interference model, nodes verify if routes in their local neighborhood are valid, i.e., all links in the routes have a high PRR. If the existing routes are still valid, then nodes generate a new transmission conflict graph and verify if the current transmission time window assignment causes a conflict. Nodes identifying conflict locally initiate and execute DRAND during the management cycles to select a new transmission time window in the network. The new transmission time window is disseminated in the local 5-hop neighbors. Note that, in some scenarios, a node may select a transmission time window that changes the length of the epoch. In such cases, the node initiates a global dissemination of the new epoch length to all

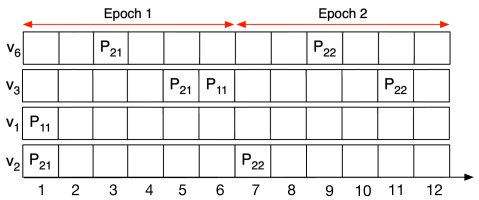


Fig. 3. Example of local scheduling in DistributedHART.

nodes. If the existing routes are not valid (i.e., links are broken), then nodes request the sensors, actuators, controller, and network manager to pause the communication between sensors-controller and controller-actuators. Once the network operation is halted, the nodes generate new routes. The new routes are used to generate channel conflict graphs and new channel allocations. Upon generating new channel allocations, nodes generate new transmission conflict graphs and time window allocation. Once the time window allocations are complete, the network operation can resume. Note that the controller can request a switch to a non-uniform time window assignment to improve the network's schedulability, which would require the same steps as mentioned above.

For a large network, the local exchange of channel, time window, and physical interference model is typically less energy-consuming than the centralized approach, where the central manager collects the physical interference model and disseminates the transmission schedule to all nodes in the network. Note that even for a small change in the link qualities, the central manager typically has to re-disseminate the schedule, which would require the controller to pause the communications between sensor-controller and controller-actuator. Thus, the overhead of generating a schedule in DistributedHART under network dynamics is smaller than that of existing centralized Wire- lessHART algorithms. Thus, handling network dynamics is a key advantage of DistributedHART.

4.3 Scheduling Policy

DistributedHART can work with any type of priorities—fixed or dynamic. To explain local scheduling policy for dynamic priority, we consider ***EDF (Earliest Deadline First)*** as an example here. EDF assigns priorities dynamically to packets according to their absolute deadlines. Since, in our method, we adopt node-level scheduling, each node has to adopt EDF policy locally. Namely, among the packets that it has to transmit or forward, the one with the shortest absolute deadline will have the highest priority. An example of local scheduling at each node is shown in Figure 3, where P_{21} represents the first packet of F_2. At time slot 1, v_1 and v_2 transmit packet P_{11} and P_{21} to v_3 and v_6, respectively. At time slot 3, v_6 transmits P_{21} to node v_3. At time slot 5, node v_3 has packets P_{11} and P_{21}. Based on the EDF policy, node v_3 selects packet P_{21} for transmission on time slot 5 and packet P_{11} remains in the queue for possible transmission at time slot 6.

To explain local scheduling policy for fixed priority scheduling, we consider ***Deadline Monotonic (DM)*** policy as an example here. DM assigns priorities to flows according to their deadlines. The flow with the shortest deadline acquires the highest priority. If the deadline is equal to the period, then the schedule generated by rate monotonic policy and DM are the same.

Since source nodes can update their sampling period/deadline and aperiodic events can occur (having their own deadlines), we do not rely on the network manager to assign priorities. Instead, the source node will append the period (or deadline) information in the packet. Thus, every intermediate node knows the priorities of the packets in its buffer. Thus, the network can handle changes due to plant/workload dynamics locally, and the manager need not update the entire schedule. Management and diagnostic superframes can run in parallel with the highest priority.

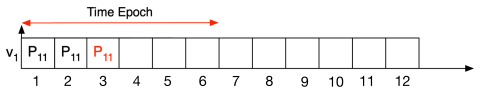


Fig. 4. Example of scheduling as dedicated and shared slot.

4.4 Online Scheduling as Dedicated and Shared Slots

A key **challenge** for DistributedHART is to incorporate both dedicated and shared slots. WirelessHART standard defines shared slot as a time slot, where many nodes transmit simultaneously to the same node. We adopt the following technique to handle shared and dedicated slots:

A node can use any slot within its time window as a dedicated or a shared slot, and any time slots outside its time window as a shared slot. When using a slot as a dedicated time slot, a node transmits a packet at the beginning of a time slot. When intending to use a slot as a shared slot, a node first senses the channel for a short duration θ. If the channel is busy, then it concludes that some node in its neighborhood is using the time slot as a dedicated time slot, and hence, does not make any transmission. If the channel is free, then it uses the slot as a shared slot. If the node intends to use a slot within its transmission time window as a shared slot, then waiting for θ time allows other nodes to use it as a shared slot. If the current slot is outside its transmission window, then waiting for θ informs nodes of transmission happening on a dedicated slot and minimizes collisions.

An example of dedicated and shared slot scheduling at node v_1 (for network shown in Figure 2 with periods $T_1 = 12$) is shown in Figure 4. If the transmission of packet P_{11} fails on time slots 1 and 2, then v_1 can use time slot 3 as shared slots even though it lies outside the transmission window of v_1.

Note that packet transmissions on a dedicated slot should not be interfered by any other transmissions. However, sensing for a short duration does not resolve all collisions caused by transmissions on a shared slot. Specifically, a node can interfere with a hidden node's transmission on a dedicated time slot. To mitigate the impact of collision and facilitate the successful decoding of a packet transmitted on a dedicated time slot, we enable **capture effect** [18] of the radio at the receiver.

Enabling Capture Effect. WirelessHART networks use IEEE 802.15.4-compliant radios [2]. In such radios, during the header decoding (or synchronization), a node's radio searches for a preamble and a start frame delimiter with the strongest **Received Signal Strength (RSS)** [2, 9]. After this, the radio generates an interrupt and locks to payload reception mode and does not search for preambles. Therefore, ***capture effect*** [9] can recover the stronger packet if it comes before the radio locks to a weaker packet's payload reception mode, requiring no physical layer modification. Hence, our objective is to ensure that a receiver (node v_3) receives a packet transmission on a dedicated slot (packet P_2) before the packet transmission on a shared slot (packet P_1). Moreover, the strongest packet can be recovered if its RSS is higher (by 1–3 dB based on modulation) than that of the other colliding signal/s. Hence, for successful reception of P_2, we adopt the following technique: When a node uses a slot as a dedicated slot, it will transmit immediately after the slot starts and will use the highest transmission (Tx) power. However, when a node uses a slot as a shared slot, it will transmit at a moderate Tx power to make the required RSS difference at the receiver. Also, a node transmits packets after θ time in a shared slot (while in a dedicated slot, it transmits packets in the very beginning of a slot). The transmission power difference and θ time difference ensures that the receiver's radio locks to the payload reception mode of P_2 and successfully receives P_2 even under collision.

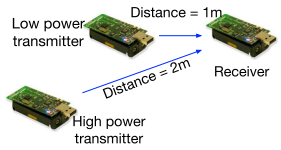


Fig. 5. Capture effect experiment setup.

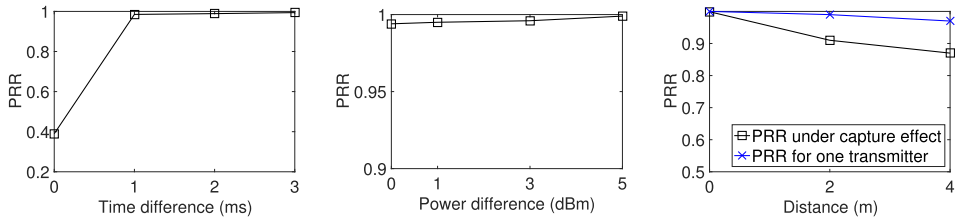


Fig. 6. Probability of successful reception through capture effect with varying distance, time, and power differences between two transmitters.

Existing work in Reference [40] demonstrates the effectiveness of capture effect for IEEE 802.15.4-based networks. Here, we experimentally determine values of θ (time difference) and Tx power difference to enable the capture effect. We use a setup consisting of three TelosB motes (that use radios based on 802.15.4), one receiver, and two time-synchronized transmitters, as shown in Figure 5. We performed one experiment to determine θ and another to determine the power difference for enabling capture effect. For the first experiment, we used a transmission power of 0 dBm for both the transmitters and varied θ and measured PRR (packet reception rate). We then found $3ms$ as a good value of θ, and using this value, in the second experiment, we decreased the power level of one transmitter while keeping the other transmitter's power at 0 dBm. We then found $3dBm$ as a good value of Tx power difference and using this value (and $\theta = 3ms$), we performed another experiment to observe the performance of capture effect under varying distance. We varied the differences between the distances (from the receiver node) of the two transmitters by increasing the distance (from the receiver node) of the transmitter that used $0dBm$. Each node transmitted 1,500 packets with a payload of 14 bytes with a period of 10 ms on channel 15. Due to the small variance in the obtained result, we present an aggregate result from 10 iterations.

Figure 6(a) shows the average PRR under varying time difference. We observed a very small-time synchronization error between the two nodes, which resulted in a PRR of 0.38 when both nodes transmit a packet at the same time. However, when θ is increased, we observed that PRR of dedicated transmission significantly improved. This phenomenon was due to two factors: (1) receiver's radio locked to the packet sent in a dedicated slot and (2) there was small/no overlap between the transmission times (due to short packet lengths). Figure 6(b) shows the average PRR under varying power difference. As expected, with the increase in power difference, we observed an increase in PRR for a packet transmitted at higher power. Based on the results shown in Figure 6, we set the value of θ to $3ms$ and Tx power difference to $3dBm$. Note that we do not change the time slot structure by delaying the transmission for 3 ms. Rather, we capitalize on the remaining 7 ms within the current time slot to successfully transmit a packet. Figure 6(c) shows the average PRR under varying distance. We observed that the PRR through capture effect decreases with an increase in distance. However, the decrease in PRR (by 0.1) is very minimal given the distance. Note that, for this experiment, we used a pessimistic scenario where the high power

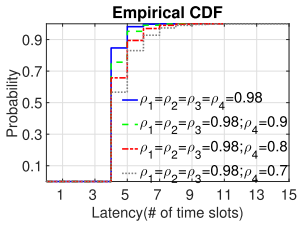


Fig. 7. Latency of one flow with 4 hops under different PRR.

transmitter is at a greater distance when compared to a low power transmitter and this may not be the case always. In DistributedHART, transmission power difference and θ can be adjusted based on the node placements, which is quite feasible as long as topology changes and/or mobility are not overwhelming.

5 END-TO-END DELAY ANALYSIS FOR DISTRIBUTEDHART

Industrial applications, such as process control, are usually time-critical and require guarantees that all packets reach their destinations within a certain deadline prior to network deployment. To meet this requirement, we propose a probabilistic end-to-end delay analysis (a sufficient test of schedulability for DistributedHART), where we compute the worst-case delay with a probability. In a probabilistic end-to-end delay analysis, the end-to-end delay experienced by a packet is a function of the required probability guarantees $\mathbb{P}_i$ and is given by $R_i(\mathbb{P}_i)$. Here, $\mathbb{P}_i$ represents the probability of a packet of flow F_i experiencing a maximum delay of $R_i(\mathbb{P}_i)$. A summary of notations used in this paper is shown in Table 1. A network designer can determine a good value of $\mathbb{P}_i$ based on the application requirements. For the sake of simplicity in estimating probabilistic end-to-end delay, we assume PRR of each link is independent of all other links in the network. The proposed probabilistic end-to-end delay analysis can be used as an acceptance test during network deployment and network/workload dynamics.

To motivate the use of a probabilistic end-to-end delay analysis technique on the primary path for DistributedHART, we perform a simulation in TOSSIM with a 4-hop network. Figure 7 shows the cumulative distribution function of latency observed from 10,000 packets on the primary path with a hop count of 4. In Figure 7, ρ_a represents the **Packet Reception Rate (PRR)** at node a. When the PRR of each link is 0.98, the worst-case delay observed was eight time slots, and the probability of a packet experiencing the worst-case delay was 0.001. However, the probability of a packet experiencing a delay less than or equal to 5 was 0.99, i.e., in most cases, the maximum delay is five time slots. A similar result can be observed for different PRR values. From these results, we can conclude that the worst-case delay experienced by a packet of a flow can be large, but the probability of a packet experiencing such a large delay is infinitesimal. Thus, we propose a probabilistic end-to-end delay analysis that gives the worst-case delay experienced by a packet with a probability, which is less pessimistic and tighter than the worst-case analysis.

In this section, we develop a delay bound analysis for DistributedHART considering DM scheduler. A similar approach can be used to develop a delay bound analysis for DistributedHART where the nodes use EDF scheduling policy. Note that, in this section, we compute the end-to-end delay experienced by a packet along the primary path of both uplink and downlink graphs. Nevertheless, the analysis can be extended to a graph route by considering a packet experiences delay on all paths of the graph route.

To calculate the end-to-end delay analysis of a flow, the central network manager must first estimate the value of γ prior to deployment. To estimate an upper bound on γ (or the chromatic

Table 1. Notations Used in the Article

Symbol	Description
D_i	Deadline of flow F_i
T_i	Period of Flow F_i
γ	Number of unique time windows in an epoch
w	Number of time slots in a time window
$\rho_{v,i}$	Probability of successful transmission of flow F_i from node v
$\mathbb{P}_i$	Probability requirement by the application
$R_i(\mathbb{P}_i)$	End-to-end delay of Flow F_i with probability $\mathbb{P}_i$
$\delta_v(F_i, \mathbb{P}_{v,i})$	Delay experienced by a packet of F_i at node v with probability $\mathbb{P}_{v,i}$
$\delta^{pre}_{v,i}(\mathbb{P}_{v,i})$	Delay experienced by a packet of F_i between its arrival at node v and the first time window
$c_{v,i}(\mathbb{P}_{v,i})$	Number of time slots required by node v to transmit a packet of F_i with probability $\mathbb{P}_{v,i}$
$\delta^{hp}_{v,i}(\mathbb{P}_{v,i})$	Delay caused by high priority flows on a packet of F_i at node v

number), the network manager can use Brook's theorem [25], where a safe upper bound on γ is the sum of 1 and maximum degree of a node in the transmission conflict graph.

The next step in estimating end-to-end delay analysis of a flow is to estimate the delay experienced by a packet at a node v on the primary path of flow F_i. Considering time windows consist of consecutive time slots, a packet experiences three sources of delay at v: (1) delay experienced between the arrival of a packet and the start of the first transmission time window ($\delta^{pre}_{v,i}(\mathbb{P}_{v,i})$), (2) number of time slots required to successfully transmit a packet ($c_{v,i}(\mathbb{P}_{v,i})$), and (3) delay caused by interrupting high priority flows ($\delta^{hp}_{v,i}(\mathbb{P}_{v,i})$). The total delay experienced by a packet of F_i at node v is given by Equation (1).

$$\delta_{v,i}(\mathbb{P}_{v,i}) = \delta^{pre}_{v,i}(\mathbb{P}_{v,i}) + c_{v,i}(\mathbb{P}_{v,i}) + \delta^{hp}_{v,i}(\mathbb{P}_{v,i}) \tag{1}$$

Considering time windows consist of non-consecutive time slots, a packet experiences an additional delay in the last epoch due to the unavailability of consecutive time slots for transmission. This additional delay can be upper bounded by the length of the transmission time window γw. Thus, the delay experienced by a packet of F_i at node v considering non-consecutive time windows can be computed as shown in Equation (2).

$$\delta_{v,i}(\mathbb{P}_{v,i}) = \delta^{pre}_{v,i}(\mathbb{P}_{v,i}) + c_{v,i}(\mathbb{P}_{v,i}) + \gamma w + \delta^{hp}_{v,i}(\mathbb{P}_{v,i}) \tag{2}$$

Computing $\delta^{pre}_{v,i}(\mathbb{P}_{v,i})$. A packet experiences the maximum delay between its arrival at a node and the first transmission time window when it arrives on the time slot immediately after the node's transmission time window. This delay can be computed as shown in Equation (3).

$$\delta^{pre}_{v,i}(\mathbb{P}_{v,i}) = (\gamma - 1)w \tag{3}$$

For example, consider $\gamma = 5$, $w = 2$, and transmission time slot of v is 9,10; a packet experiences a maximum delay if it arrives on time slot 1 and has to wait for eight time slots.

Computing $c_{v,i}(\mathbb{P}_{v,i})$. To compute the number of time slots required to successfully transmit a packet, let $\rho_{v,i}$ be the probability of successful transmission of a packet of flow F_i from node v, and $\delta_v(F_i, \mathbb{P}_{v,i})$ is the delay experienced by a packet of F_i at node v with probability $\mathbb{P}_{v,i}$. The probability $\mathbb{P}_{v,i}$ can also be interpreted as the probability requirement by the application on the link from node v for a packet of flow F_i. The probability of successful reception of a packet after c transmissions on dedicated slots by node v is expressed as $1 - (1 - \rho_{v,i})^c$. The number of slots needed to successfully transmit a packet with a probability $\mathbb{P}_{v,i}$, can be computed by equating the

probability of successful transmission after $c_{v,i}(\mathbb{P}_{v,i})$ transmissions to $\mathbb{P}_{v,i}$, i.e.,

$$\mathbb{P}_{v,i} = 1 - (1 - \rho_{v,i})^{c_{v,i}(\mathbb{P}_{v,i})}$$
$$\Rightarrow log(1 - \mathbb{P}_{v,i}) = c_{v,i}(\mathbb{P}_{v,i}) \times log(1 - \rho_{v,i}).$$

Thus, the number of time slots required by node v to transmit a packet of flow F_i with a probability $\mathbb{P}_{v,i}$ is given by Equation (4).

$$c_{v,i}(\mathbb{P}_{v,i}) = \left\lceil \frac{log(1 - \mathbb{P}_{v,i})}{log(1 - \rho_{v,i})} \right\rceil \tag{4}$$

For example, if the requirement on the probability of successful transmission on a link is 0.99 ($\mathbb{P}_{v,i} = 0.99$) and PRR ($\rho_{v,i}$) is 0.9, then $c_{v,i}(\mathbb{P}_{v,i})$ is computed as shown below.

$$c_{v,i}(\mathbb{P}_{v,i}) = \frac{log(1 - 0.99)}{log(1 - 0.9)} = \frac{log(0.01)}{log(0.1)} = \frac{-2}{-1} = 2$$

Computing $\delta^{hp}_{v,i}(\mathbb{P}_{v,i})$. Computing the delay experienced by a packet due to high priority packets at a node in DistributedHART is similar to the delay experienced by a task due to high-priority tasks on a uniprocessor platform. A detailed description of the similarity between a packet transmission and task scheduling can be found in Reference [20]. Thus, we use the delay computation of a task under a uniprocessor DM scheduling as the foundation for the $\delta^{hp}_{v,i}(\mathbb{P}_{v,i})$ computation in DistributedHART.

In a uniprocessor scheduling, a processor can execute tasks consecutively without halting. However, in DistributedHART, for an epoch of length γw, a node can only transmit a packet within the w time slots of its transmission time window. That is, if a high priority packet takes the entire transmission time window, then a packet has to wait until the next epoch to attempt a transmission. For example, consider v has a transmission time window of two time slots 2, 3, and the epoch length is 10 time slots. Assume at time slot 2, v has a higher priority packet P_1 and a lower priority packet P_2 in its queue. If P_1 requires slots 2,3 for transmission, then packet P_2 has to wait until time slot 12, since time slot 12 is in v's next transmission time window. Note that, during the epoch within which packet P_2 is successfully transmitted, packet P_2 is not delayed for γw time slots.

Accounting for the unavailability of time slots for packet transmissions, $\delta^{hp}_{v,i}(\mathbb{P}_{v,i})$ is given by Equation (5), where $HP_v(F_i)$ denotes the high priority flows of F_i passing through v.

$$\delta^{hp}_{v,i}(\mathbb{P}_{v,i}) = \sum_{F_j \in HP_v(F_i)} \left\{ \left\lfloor \left\lceil \frac{\delta_v(F_i) - D_j}{T_j} \right\rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w} \right\rfloor (\gamma w - 1) + \left\lceil \frac{\delta_v(F_i) - D_j}{T_j} \right\rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w} \right\} \tag{5}$$

Note that $\lfloor \lceil \frac{\delta_v(F_i)-D_j}{T_j} \rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w} \rfloor \gamma w$ denotes the delay of a packet in all of the epochs except the last one, and $\lceil \frac{\delta_v(F_i)-D_j}{T_j} \rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w} - \lfloor \lceil \frac{\delta_v(F_i)-D_j}{T_j} \rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w} \rfloor$ represents the delay of a packet in the last epoch. Here, $C_v(\mathbb{P}_{v,j})$ denotes the number of time slots required by node v to transmit a packet of F_j with probability $\mathbb{P}_{v,j}$. For a fixed priority local scheduler, the value of $C_v(\mathbb{P}_{v,j})$ can be computed during the response time calculation for flow F_j. For a dynamic priority local scheduler, an estimate of $C_v(\mathbb{P}_j)$ can be used. The total delay experienced by a packet at a node v is shown in

Equation (6).

$$\delta_v(F_i, \mathbb{P}_{v,i}) = (\gamma - 1)w + C_{v,i}(\mathbb{P}_{v,i}) + \sum_{F_j \in HP_v(F_i)} \left\{ \left\lfloor \left\lceil \frac{\delta_v(F_i) - D_j}{T_j} \right\rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w} \right\rfloor \times (\gamma w - 1) + \left\lceil \frac{\delta_v(F_i) - D_j}{T_j} \right\rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w} \right\} \tag{6}$$

Since nodes use a real-time scheduling policy to schedule packets locally, the worst-case delay experienced by a packet at different nodes is independent of each other. Thus, we express the total delay (experienced by a flow) as the sum of delays experienced at each node on the primary path. For implicit deadline flows, the end-to-end delay R_i experienced by a control loop F_i under DistributedHART with DM scheduling is given by Equation (7) with a probability $\mathbb{P}'_i = \prod_{v \in V_i} \mathbb{P}_{v,i}$.

$$R_i(\mathbb{P}'_i) = \sum_{v \in V_i} \delta_v(F_i, \mathbb{P}_{v,i}) \tag{7}$$

The delay bound computation, described above, generates the end-to-end delay with a probability of $\mathbb{P}'_i$. To compute the end-to-end delay for a selected value of $\mathbb{P}_i$, we first compute the end-to-end delay for all combinations of $\mathbb{P}_{v,i}$ that result in a end-to-end probability guarantee of $\mathbb{P}_i$. From the list of all possible combinations, we select the maximum delay for a probability $\mathbb{P}_i$ as the end-to-end delay for flow F_i. Note that the number of combinations of $\mathbb{P}_{v,i}$ is 2, since a node can use at most 2 time slots to make a transmission along a link on the primary path. Since the possible combinations of $\mathbb{P}_{v,i}$ for each link are bounded by a fixed constant 2, the number of delay bound calculations required for one flow is $O(n^2)$, where n is the number of nodes in the network.

Note that the computation of the transmission delay gives an upper bound on the maximum delay experienced at a node considering both consecutive and non-consecutive availability of time slots at a node. For example, consider a non-consecutive availability of time slots at node v with time slots 2,8 and $c_{v,i}(\mathbb{P}_{v,i}) = 2, \gamma = 5, w = 2$; there are two possible cases that result in a maximum transmission delay: (1) when packet arrives at time slot 3 and (2) when packet arrives at time slot 9. When a packet arrives at time slot 3, the maximum waiting time is five, and waiting between the transmission slots is five; thus, the total transmission delay is 10 time slots. When a packet arrives at time slot 9, the maximum waiting time is three, and waiting between the transmission slots is five; thus, the total transmission delay is 10 time slots.

6 NON-UNIFORM TIME WINDOW ASSIGNMENT IN DISTRIBUTEDHART

The previous sections present the design and analysis of DistributedHART with uniform time window allocation. Intuitively, assigning a fixed number of time slots to each node can cause large delays at nodes with high traffic, like the access point. Assigning time window lengths based on the flow of traffic passing through a node can potentially reduce the delays experienced by the flows. In this section, we present the design of DistributedHART with non-uniform time window allocation and extend the end-to-end delay analysis for non-uniform time window allocation. Furthermore, we show a comparison between uniform and non-uniform time window allocation.

6.1 Algorithm to Select Non-uniform Time Window Length

Here, we present a simple heuristic to select the number of time slots per window at a node. The intuition behind the heuristic is to assign time slots based on the rate of flows passing through a node. The heuristic initializes each node with a primary path of a route passing through it with α time slots, where α represents the minimum number of time slots allocated to a node. For every ϕ units of rate passing through a node, the heuristic adds β time slots. Considering a node has σ^v

units of rate passing through a node v, the length of transmission time window for a node v is given by Equation (8).

$$w_v = \alpha + \left\lceil \frac{\sigma^v}{\phi} \right\rceil \times \beta \tag{8}$$

Note that α, β, and ϕ are design parameters, and σ^v is calculated as shown in Equation (9), where V_i is the set of nodes in the route of F_i.

$$\sigma^v = \sum_{F_i \ if \ v \in V_i} \frac{1}{T_i} \tag{9}$$

6.2 Non-uniform Time Window Allocation

Non-uniform time window allocation can cause higher contention for smaller time windows and underutilization of larger time windows. To mitigate this challenge, we represent each time slot by a unique color. Each node then uses vertex multi-coloring to select multiple transmission time slots and locally define their transmission time window. For example, one node can select time slots 1, 4, and 7 as its transmission time window, and another can select time slots 3 and 7 as its transmission time window. For distributed vertex multi-coloring, we choose a variant of DRAND [26] where each node selects multiple colors instead of one. Although there exist many distributed vertex multi-coloring algorithms, we pick DRAND for its simplicity in implementation.

One key issue with using DRAND is that it requires the maximum number of available colors (or $\hat{\gamma}$) *a priori*. The central manager can estimate $\hat{\gamma}$ from the product of the maximum degree of a node and maximum window length of all nodes, similar to uniform time window assignment. However, this estimation is very loose, as the central manager assumes each node selects a fixed/maximum window length. Such a loose estimation can result in idle time slots within an epoch, significantly increasing the latency. To overcome this issue, we propose to use a leader election algorithm after DRAND to determine the leader or the node with the last used time window within the epoch. The leader can reduce the epoch's length ($\hat{\gamma}$) and re-broadcast the new epoch value to all nodes. Typically, the leader election is a polynomial-time algorithm for a mesh network and needs to be executed only during network initialization and network/workload dynamics.

6.3 End-to-end Delay Analysis for Non-uniform Time Window Assignment

To compute the end-to-end delay analysis for non-uniform time window assignment, we use a similar approach as described in Section 5. The network manager can estimate an upper bound on $\hat{\gamma}$, which facilitates the schedulability test. Similar to Equation (2), delay experienced by a packet of F_i at node v with a probability $\mathbb{P}_{v,i}$ consists of four sources: (1) delay experienced between the arrival of a packet and the start of the first transmission time window ($\hat{\gamma} - w_v$), (2) number of time slots required to successfully transmit a packet ($c_{v,i}(\mathbb{P}_{v,i})$), (3) delay caused by the unavailability of consecutive time slots $\hat{\gamma}$ and (4) delay caused by interrupting high priority flows ($\delta^{hp}_{v,i}(\mathbb{P}_{v,i})$). Considering $\hat{\gamma}$ represents the epoch length and w_v represents the v's time window length, the total delay experienced by a packet of F_i at node can be given by Equation (10).

$$\delta_{v,i}(\mathbb{P}_{v,i}) = \hat{\gamma} - w_v + c_{v,i}(\mathbb{P}_{v,i}) + \hat{\gamma} + \delta^{hp}_{v,i}(\mathbb{P}_{v,i}) \tag{10}$$

The number of time slots required to successfully transmit a packet is computed similar to that in uniform time window assignment and is given by Equation (4). The delay caused by interrupting high priority flows is given by Equation (11).

$$\delta^{hp}_{v,i}(\mathbb{P}_{v,i}) = \sum_{F_j \in HP_v(F_i)} \left\{ \left\lceil \frac{\delta_v(F_i) - D_j}{T_j} \right\rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w_v} \right\rceil (\hat{\gamma} - 1) + \left\lceil \frac{\delta_v(F_i) - D_j}{T_j} \right\rceil \frac{C_{v,j}(\mathbb{P}_{v,j})}{w_v} \Bigg\} \tag{11}$$

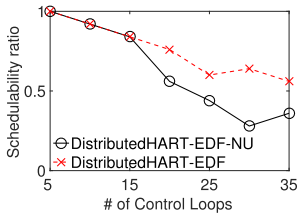


Fig. 8. Performance of non-uniform time window assignment in DistributedHART.

For implicit deadline flows, the response time R_i experienced by a control loop F_i under DistributedHART with DM scheduling is given by Equation (12) with a probability $\mathbb{P}'_i = \prod_{v \in V_i} \mathbb{P}_{v,i}$.

$$R_i(\mathbb{P}'_i) = \sum_{v \in V_i} \delta_v(F_i) \tag{12}$$

6.4 Performance Evaluation of Non-uniform Time Window Assignment

We performed simulation in TOSSIM [16] to evaluate the performance of DistributedHART with EDF local scheduling on non-uniform time windows. We used a 148-node network topology and varied the number of control loops from 5 to 35. For further details about the simulation setup, please refer to Section 9. We present the aggregate result from 50 random test cases. For each test case, we randomly selected sensor and actuators and assigned random harmonic periods in the range of $2^{11\sim13}$ time slots. To decrease the workload on the network, we doubled the range after adding every 10 flows. For this simulation, we used $\alpha = 1$ and $\beta = 1$. We selected $\phi = (\min \frac{1}{T_i} + \max \frac{1}{T_i}) \times \frac{7}{8}$. For the test cases with five control loops, the maximum rate was 8 times the minimum rate, and hence, we chose $\frac{7}{8}$ as the smallest increase to the non-uniform time window assignment. Schedulability ratio comparison between non-uniform (DistributedHART-EDF-NU) and uniform time window assignment (DistributedHART-EDF) is shown in Figure 8.

In this simulation, we have observed that non-uniform time window assignment performs better for some and poorly for other test cases. We have observed that when the number of control loops is 15, some test cases schedulable under uniform time window assignment were not schedulable under non-uniform time window assignment, and vice versa. When the number of control loops is greater than 20, we observed that the average schedulability ratio of non-uniform time window assignment was poor when compared to uniform time window assignment. We observed that this result is due to the increase in the epoch length $\hat{\gamma}_n$, which resulted in a significant increase in the end-to-end delay. From this simulation result, we can conclude that for a smaller number of control loops, non-uniform time window assignment performs better in some cases and uniform time window assignment performs better under some other. For larger number of control loops uniform time window assignment performs better in more cases.

7 LATENCY UNDER DISTRIBUTEDHART

We performed simulations in TOSSIM to evaluate the performance of DistributedHART under latency of each flow. We used a 148-node network topology with 25 flows. For further details about the topology, please refer to Section 9. We assigned a period of 4s for all the flows. The same period assignment is for this simulation alone, and we used different periods for all other simulations and experiments. We assign same periods, because it allows us to compare the latency of flows.

Figure 9 shows the latency of each flow under DistributedHART with EDF local scheduling. We observed an average latency of 3.825s for all flows. Figure 10 shows that the maximum latency

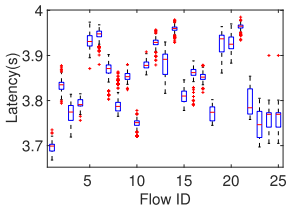


Fig. 9. Latency under DistributedHART.

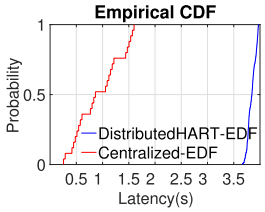


Fig. 10. Latency comparison between Centralized-DM and DistributedHART-DM.

observed under DistributedHART-EDF to be 3.98s while that of centralized is 1.7s. These results show that latency in DistributedHART is higher than centralized algorithms.

In DistributedHART, DRAND facilitates time window selection through a random selection of colors. Intuitively, a random selection of colors induces long latencies, as shown in the following example: Assume a flow F_i has a primary path $v_i - > v_j - > v_k$, where the source is v_i and destination is v_k. For this flow, if the time window assignment for $v_i, v_j,$ and v_k is 1, 2, and 3, respectively, then latency of the path is $3w$ (where w is the length of each window). On the contrary, if the time window assignment for $v_i, v_j,$ andv_k is 3, 2, and1, respectively, then a packet has to wait for an epoch (γw) at each node, and hence, latency is $3w \times \gamma$.

To improve the latency, we propose a heuristic called DistributedHART-IL. The intuition behind the heuristic is to prioritize nodes during distributed time window selection such that the first node of the highest priority flow selects the first time slot. In DistributedHART-IL, we prioritize nodes by assigning a different probability to start the selection of transmission time window. The probability of v depends on the highest priority flow passing through v and its position on the highest priority route (ϖ_v). To capture the maximum priority of flow passing through a node v, we use a normalized flow priority to each node, given by Υ_v, and is computed as the ratio of the highest priority flow period in the network and period of the highest priority flow passing through v. We use a weighted summation of the two parameters, ϖ_v and Υ_v, to compute a node's probability and is given by $\Upsilon_v - \iota\varpi_v$, where ι represents the weight, which is typically less than 0.1.

We ran simulations on TOSSIM to evaluate the performance of DistributedHART-IL. We used a 148-node network with 25 control loops and a 4s period. Latency of each flow under DistributedHART is shown in Figure 11. We observed that DistributedHART-IL results in similar performance as DistributedHART. We observed the average latency is 3.99s. To further evaluate the impact on schedulability, we ran more simulations by varying the number of flows in the network. Figure 12 shows the comparison between DistributedHART-IL and DistributedHART under the schedulability ratio. We observed that Distributed-HART-IL performs better than DistributedHART. However, we have observed that some cases that were schedulable under DistributedHART were not schedulable under Distributed-HART-IL. Although DistributedHART-IL offers better schedulability, DistributedHART is easy to implement and converges faster. From this simulation, we can conclude that both DistributedHART and DistributedHART-IL are good choices. Since both Distributed-HART and DistributedHART-IL perform similarly, we only evaluate DistributedHART through experiments and simulations.

When compared to centralized algorithm, both DistributedHART and DistributedHART-IL may perform poorly under latency. However, it achieves a similar schedulability ratio (which is a more important metric for real-time networks) as centralized algorithms in most of the cases. Furthermore, the centralized approach relies on global information to achieve low latency. Acquiring global information is challenging, and hence centralized algorithm is not scalable. However,

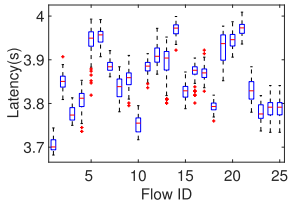


Fig. 11. Latency under DistributedHART-IL.

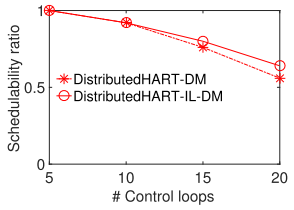


Fig. 12. Comparison between DistributedHART-IL and DistributedHART.

DistributedHART uses local information to make scheduling decisions, thereby providing advantages such as supporting any type of traffic (periodic/aperiodic), handling network-dynamics (which is frequent in industrial environments), and scalability (which is important for Industrial Internet of Things). Under network-dynamics, a centralized approach re-calculates and re-distributes schedules among nodes. Since network dynamics are frequent in industrial environments, frequently calculating and re-distributing the schedules degrades the performance of centralized approach. Thus, over a long period of time (with frequent network-dynamics), the overall performance of the centralized approach can be worse than DistributedHART (even with long latencies).

8 TESTBED EXPERIMENTS

We implemented DistributedHART on TinyOS 2.2 [1] and evaluated on a 130-node TelosB mote testbed [4] for real experiments. TelsoB devices use Chipcon CC2420 radios, which are compliant with the IEEE 802.15.4 standard. Note that the physical layer of WirelessHART is similar to 802.15.4 physical layer. We deployed the nodes in a 20×10 ft room of the Maccabees building. To create a multi-hop network, each node used a transmission power of $-28.7dBm$. The topology of the testbed is shown in Reference [4]. Our DistributedHART implementation consists of multi-channel TDMA MAC protocol. Time is divided into 10 ms slots, and clocks are synchronized using the **Flooding Time Synchronization Protocol (FTSP)** [19]. We ran FTSP algorithm frequently to avoid issues with capture effect. For a fair comparison, we used centralized version of the graph routing algorithm proposed in Reference [22]. For simplicity in implementation and experimentation, network manager computed the channel and time window allocations and disseminated them using TinyOS dissemination protocol library. We then evaluated the online scheduling of DistributedHART.

8.1 Evaluation Metrics

We evaluated DistributedHART using four metrics (1) energy, (2) memory, (3) convergence time, and (4) schedulability performance and then compared the performance with centralized EDF [29] and DM [2] (with spatial re-use). *Schedulable ratio* is defined as the fraction of test cases that were schedulable among all cases. Each test case corresponds to a set of flows and is said to be *schedulable* if all packets from all flows met their deadlines (i.e., max latency $\leq$ deadline). The deadlines of the flows were set equal to their periods. Note that for any flow that is observed to be schedulable, its packets were delivered to the destination (actuator), and they were delivered within deadlines. *Memory consumption* is the average memory consumed per node to store a schedule. *Convergence Time* is the average time taken for all nodes to disseminate a schedule excluding the time taken to generate routes. *Energy consumption* is the average energy consumed per node in Joules to disseminate a schedule. In our testbed, we use USB cables to power the nodes and hence, we can not obtain the actual energy consumed. We use a product of the number of transmissions and energy

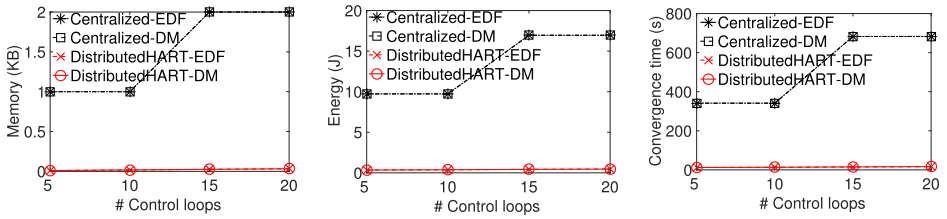


Fig. 13. Experimental result under varying number of flows considering harmonic periods.

consumed for each communication ($0.22mJ$, calculated from TelosB datasheet [3] with $5.5ms$ Tx time) to estimate energy consumption per node. We used a CSMA/CA protocol to disseminate schedule for DistributedHART and centralized algorithms. Thus, duty-cycle of operation (equal to convergence time) was very large, and hence, was not a good metric for comparison.

8.2 Results

The performance comparison between DistributedHART and centralized scheduler proposed in WirelessHART standard is shown in Figure 13 and Figure 14. The fixed priority and dynamic priority centralized schedulers are labeled "Centralized-DM" and "Centralized-EDF," respectively. Similarly, DistributedHART with fixed priority and dynamic priority local scheduling are labeled "DistributedHART-DM" and "DistributedHART-EDF," respectively. As the performance of Orchestra [10] and DiGS [35] is expected to be worse, we evaluated them only in simulation. Figure 13 and Figure 14 show the aggregate result from 25 test cases under varying number of flows in the network. In this experiment, we varied the number of flows between 5 and 40. For each test case, we generated flows by randomly selecting source and destination nodes. For test cases with 5 flows, we assign harmonic periods in the range $2^{13\sim16}$ time slots. To decrease the workload on the network, we double the range after adding every 10 flows. Since the confidence interval is very close to the average, the confidence intervals are not visible in the average energy, time, and memory consumption results shown in the article (including Figure 13).

8.2.1 Memory Consumption. Typically, memory consumption in centralized algorithms is proportional to the hyper-period. In some special cases, a compact schedule may be feasible. However, we considered a general scenario where memory consumption is proportional to the length of the hyper-period. Figure 13(a) shows a step increase in memory consumption, since we double the hyper-period for every 10 control loops. Since the transmission schedule repeats after every time epoch, time window information during the first epoch and time epoch length is sufficient. This information is subsequently smaller than centralized transmission schedule. We observed a small increase in worst-case chromatic number with the increase number of control loops. We observed that DistributedHART consumes at least 75% less memory than both centralized EDF and DM.

8.2.2 Energy Consumption and Convergence Time. The centralized algorithms use a dissemination protocol to broadcast schedules to all nodes in the network. Hence, average energy consumption at a node is dependent on the length of the schedule. Thus, Figure 13(b) shows a step increase in average energy consumption similar to memory consumption result. In this experiment, for the sake of simplicity, the central managed computed channel and time window allocation and disseminated the information even for DistributedHART. In DistributedHART, the length of the schedule was only dependent on the worst-case chromatic number and hence, Figure 13(b) shows the energy consumption is close to constant. We observed that DistributedHART consumes at least 95% less

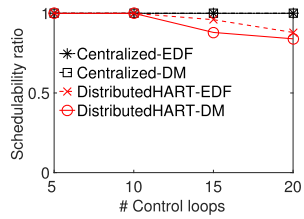


Fig. 14. Experiment results showing schedulability ratio under varying number of flows.

energy than EDF. Similar to energy consumption, convergence time for centralized algorithms is also dependent on hyper-period while convergence time for DistributedHART is almost constant. Figure 13(c) shows that DistributedHART consumes at least 95% less convergence time than EDF.

8.2.3 Schedulability Ratio. Centralized algorithms rely on global knowledge of channel/link quality information and harmonic periods that made it feasible for EDF to achieve high schedulability ratio (this may not be feasible with arbitrary periods), as shown in Figure 14. In this experiment, we used a very dense deployment. Thus, the worst-case chromatic number or γ for DistributedHART was very high. We also observed that γ increases with an increase in the number of flows, since more nodes require time windows. This increase in γ increases the end-to-end delay and decreases schedulability ratio. Since a distributed scheduler handles networks/workload dynamics and saves energy, it is expected to perform poorly under schedulability ratio due to the lack of global information. However, DistributedHART is highly competitive in terms of schedulability ratio when compared to centralized algorithms. From this experiment, we can conclude that DistributedHART is a practical choice for wirelessHART, as it offers a competitive schedulability ratio and consumes less energy and convergence time.

9 SIMULATION

9.1 Simulation Setup

We evaluated DistributedHART on a 148-node network through simulations in TOSSIM [16]. We used a 74-node testbed topology [34] deployed over a wider area to complement the experiments. To scale with the number of nodes, we assumed all nodes are placed in a grid structure and replicated this grid. We added edges between neighboring grids to generate a connected bigger topology. We followed the fully distributed approach for allocating channel and time windows, i.e., energy consumption and convergence time metrics also include the overhead of exchanging physical interference model and computing channel and time windows at each node. We evaluated the performance of DistributedHART under varying number of flows with harmonic periods, number of nodes, varying time window lengths, number of flows with non-harmonic periods, hyper-periods, and workload dynamics. We also evaluated the performance of the proposed schedulability analysis. For the simulation results, we presented the aggregate result from 50 random test cases. For each test case, we randomly selected sensor and actuators and assigned random harmonic periods in the range of $2^{11\sim13}$ time slots. The range of the periods was doubled after 10 new flows.

9.2 Performance under Varying Window Lengths

Figure 15 shows the schedulability ratio under different w. Our results showed that $w = 1$ offered ta better schedulability ratio when compared to $w = 2$ or $w = 3$. For $w = 2$ (or $w = 3$), a packet waited for 2γ (or 3γ) time units between its arrival and the availability of the first transmission window, at each node. Such long delays increased the total latency of the packet and decreased the schedulability ratio. We also observed that, most often, packets from all flows arrived at v_i

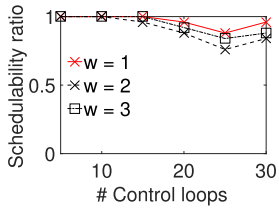
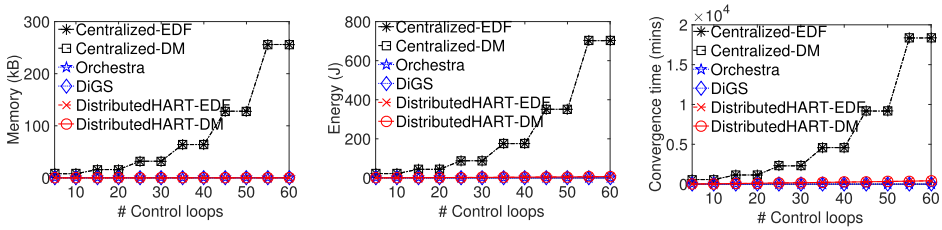


Fig. 15. Schedulability ratio vs. w.



(a) Memory consumption (b) Energy consumption (c) Convergence time

Fig. 16. Performance under varying number of flows considering harmonic periods.

during different time epochs. Therefore, the second slot in a transmission window (in most cases) remained unused. We determine $w = 1$ as the good setting for this network and traffic pattern. This result also implies that longer time windows may not necessarily increase the schedulability ratio.

9.3 Performance under Varying Number of Flows

The performance comparison between DistributedHART and centralized algorithms under varying number of flows with harmonic period assignments is shown in Figure 16. We also compare the performance of DistributedHART with Orchestra [10] and DiGS [35]. Orchestra and DiGS assign time window based on the nodeId. We use the same routing protocol and local scheduling algorithm (EDF) for Orchestra and DiGS as DistributedHART, for a fair comparison. Since Orchestra does not have a dedicated and shared slot assignment, we consider that both transmissions happen within the time window. We used $w = 1$ for DistributedHART, Orchestra, and DiGS. We varied the number of flows from 5 to 60. Simulation results for this setup are shown in Figure 16.

Memory Consumption. Similar to the experiment result, memory consumption of centralized algorithms doubles for every 10 flows, as shown in Figure 16(a). For DistributedHART, memory consumption depends only on γ, which has a small/negligible increase. This simulation shows that DistributedHART consumes a minimum of 95% less memory than centralized algorithms.

Energy Consumption and Convergence Time. In centralized EDF and DM, nodes consume energy during schedule dissemination. Since the number of messages transmitted by each node in centralized algorithms is proportional to the hyper-period, Figure 16(b) shows an exponential increase in average energy consumption. However, for DistributedHART, each node has to communicate only with its neighboring nodes in the conflict graph. We use controlled flooding to communicate with them, since routes to all nodes are not available. Thus, the average energy consumption of a node only increases linearly. From these simulations, we observed that DistributedHART consumes 95% less energy when compared to centralized algorithms. Similar to energy, convergence time for DistributedHART also increases linearly. DistributedHART consumes 90%

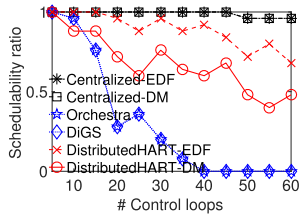
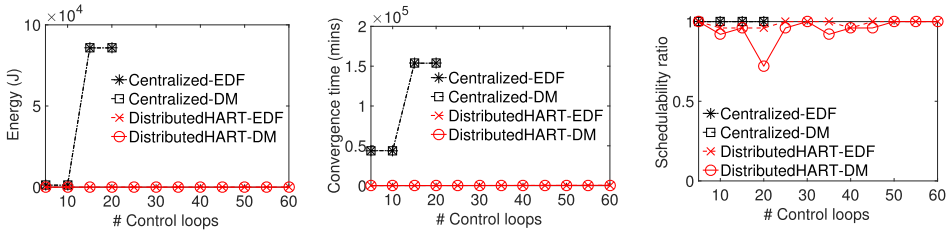


Fig. 17. Schedulability ratio under varying number of flows and harmonic periods.



(a) Energy consumption (b) Convergence time (c) Schedulability Ratio

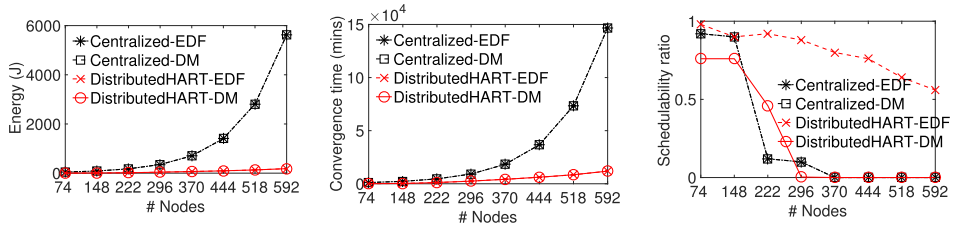
Fig. 18. Performance under varying number of flows considering non-harmonic periods.

less time than centralized algorithms. Orchestra and DiGS use an autonomous approach where each node computes its schedule locally and does not require any communication between nodes.

Schedulability Ratio. In this simulation, we consider harmonic periods that make it feasible for centralized EDF and DM to achieve very high schedulability ratio, close to optimal, because they assume all local information is available at the network manager. Thus, Figure 17 shows a high schedulability ratio for centralized algorithms. For DistributedHART, smaller γ for initial conditions results in similar schedulability as centralized algorithms. With the increase in the number of flows, γ increases linearly, which increases end-to-end delay and decreases schedulability ratio. We selected sensors, actuators, and periods randomly, which attributes to an increase in schedulability ratio with an increase in the number of flows. Although a distributed scheduler is expected to perform poorly under schedulability ratio due to the lack of global information, DistributedHART is highly competitive in terms of schedulability ratio when compared to centralized algorithms. For Orchestra and DiGS, γ is equal to the number of nodes in the network, which causes a large delay at each node of the flow. Thus, the schedulability ratio is very low compared to DistributedHART. Due to the poor schedulability ratio, we do not present results of Orchestra and DiGS in other evaluations. From these results, we can conclude that DistributedHART outperforms centralized algorithms under energy, memory, and convergence time while achieving similar schedulability ratio, and Orchestra/DiGS under schedulability ratio.

9.4 Performance under Varying Number of Flows Considering Non-harmonic Periods

The performance comparison of DistributedHART and centralized scheduler under varying number of flows with non-harmonic period assignments is shown in Figure 18. We used $w = 1$ for DistributedHART. We varied the number of flows from 5 to 60. We presented the aggregate result from 50 random test cases. For each test case, we randomly selected sensor and actuators and assigned random harmonic periods in the range of $2^{11\sim13}$ time slots. To decrease the workload on the network, we doubled the range after adding every 10 flows.



(a) Energy consumption (b) Convergence time (c) Schedulability Ratio

Fig. 19. Performance under varying number of nodes.

Energy Consumption and Convergence Time. Since energy consumption and convergence time for EDF and DM are proportional to the hyper-period, they increase exponentially, as shown in Figures 18(a), 18(b). However, energy consumption and convergence time for DistributedHART increase linearly. We observed that DistributedHART saves a minimum of 99% of energy and 99% of convergence time for flows with non-harmonic periods. We observed that DistributedHART saves a minimum of 99% of energy and 99% of convergence time for flows with non-harmonic periods.

Schedulability Ratio. For centralized algorithms, we could compute schedulability ratio up to 20 flows, since the time to generate a table driven schedule (not including simulation time) was too large. In this simulation, we observed a close to 1 schedulability for DistributedHART. From this simulation, we can conclude that for non-harmonic periods, DistributedHART outperforms centralized algorithms.

9.5 Performance under Varying Number of Nodes

Here, we show the performance of DistributedHART under varying number of nodes. Figure 19 shows the simulation results when number of flows is 20% of the number of nodes.

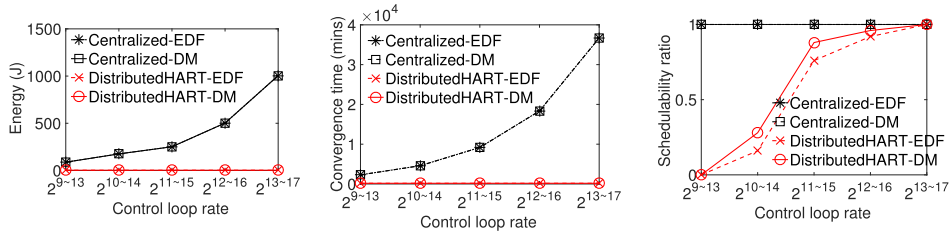
Energy Consumption and Convergence Time. Both energy and convergence time follow the similar curves as memory consumption, as they are also dependent on hyper-period. Figure 19(a) and Figure 19(b) show DistributedHART consumes at least 94% less energy and 85% less convergence time.

Schedulability Ratio. As shown in Figure 19(c), centralized algorithms are not scalable with the number of nodes due to the huge wastage of time slots. In contrast, DistributedHART offers better schedulability ratio when compared to centralized algorithms. From this result, we can conclude that DistributedHART scales with number of nodes.

9.6 Performance under Varying Hyper-periods

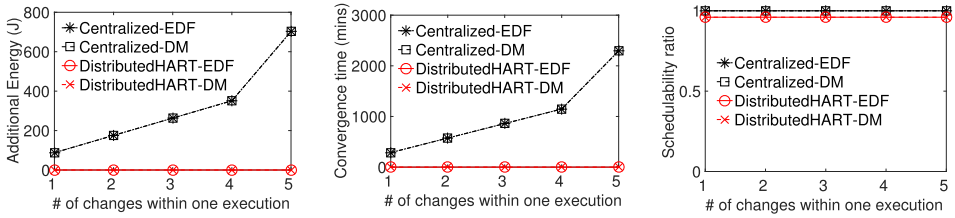
Here, we show the performance of DistributedHART under varying ranges of hyper-periods. We used the same setup as performance under varying number of flows with non harmonic periods (Section 9.4). However, we kept the number of flows constant at 30 and varied the range of periods from $2^{9\sim13}$ to $2^{13\sim17}$.

Energy Consumption and Convergence Time. As expected, energy and convergence time of centralized EDF and DM increases exponentially. However, energy (as shown in Figure 20(a)) and convergence time (as shown in Figure 20(b)) for DistributedHART is constant and remain unaffected by varying hyper-period lengths.



(a) Energy consumption (b) Convergence time (c) Schedulability Ratio

Fig. 20. Performance under varying hyper-periods.



(a) Energy consumption (b) Convergence time (c) Schedulability ratio

Fig. 21. Performance under varying workload dynamic.

Schedulability Ratio. As shown in Figure 20(c), with an increase in the range of periods, schedulability ratio remains constant at 1 for centralized algorithms and DistributedHART. For DistributedHART, we observed that increasing the period improved the schedulability ratio due to the increase in deadline, while total delay remained constant. From these results, we can conclude that under DistributedHART outperforms centralized algorithms in terms of energy, convergence time, and memory while achieving similar schedulability ratio.

9.7 Performance under Varying Workload Dynamics

Figure 21 shows the performance of DistributedHART under different network dynamics. In this simulation, we kept the number of flows constant at 30 and varied (increased or decreased) the period of random 5 flows per workload change, while ensuring hyper-period is in between $2^{14\sim18}$ time slots. We use the number of workload changes in one execution as a parameter for comparing the performance of DistributedHART and centralized algorithms.

Energy Consumption and Convergence Time. For centralized algorithms, every change in workload necessitates an update in the schedule. A centralized approach has to collect the entire topology, re-create schedules, and distribute the new schedules to all nodes in the network for each network/workload dynamic. This new schedule has to be re-disseminated to the nodes. As shown in Figure 21(a), energy consumed by centralized algorithms (for changes in workload) increases linearly with the increase in the number of workload changes. Thus, centralized approaches are inefficient in large networks with frequent network dynamics. In DistributedHART, the local scheduler at each node handles workload dynamics. Thus, DistributedHART requires $0J$ of additional energy and 0s of convergence time to generate a schedule (for every workload change). Therefore, Figure 21(b) shows a linear increase for centralized algorithms and 0 for DistributedHART.

Schedulability Ratio. In the event of a workload change, we define a test case to be schedulable if all flows in the test case meet the deadline before and after the workload change. In our simulation,

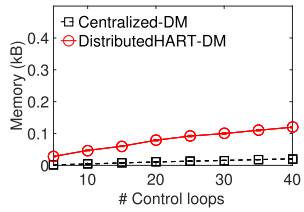


Fig. 22. Memory consumption comparison with centralized–compact scheduler.

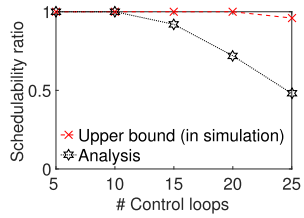


Fig. 23. Schedulability ratio comparison with an upper bound.

we observed that centralized algorithms and DistributedHART have similar schedulability ratio, where centralized algorithms perform 4% better than DistributedHART. A centralized approach has to collect the topology, re-create, and distribute new schedules to all nodes upon dynamics. Network dynamics are quite frequent in industrial environments, making a centralized approach inefficient in large-scale networks. From this simulation, we can conclude that for varying workload requirements, DistributedHART outperforms centralized algorithms in terms of energy and convergence time while offering similar schedulability ratio.

9.8 Performance Comparison between DistributedHART and Centralized Scheduler with Compact Schedules

Here, we show the performance of DistributedHART when compared to the centralized scheduler with compact schedules. For special scenarios such as harmonic periods and a rate monotonic scheduler, centralized schedulers can create a compact schedule, i.e., a node can use the first packet's schedule to generate the schedule for subsequent packets. Note that a central scheduler should assign each node the time slot and period of the first packet for each flow passing through it. For generic scenarios, centralized schedulers have to consider interference from all packets, hence this approach will not work. Figure 22 shows the memory consumption comparison between DistributedHART-DM and centralized-DM with a compact schedule. Our simulation results show that DistributedHART and centralized-EDF compact scheduler consume a similar amount of memory with DistributedHART, consuming 0.08 KB of additional memory.

9.9 Performance of End-to-end Delay Analysis

Here, we show the performance of the proposed schedulability analysis. We compare the schedulability ratio obtained by the schedulability analysis and the simulation result (which provides a conservative upper bound). We use a similar setup as Section 9.4 with periods in range $2^{12\sim16}$ time slots. We present the schedulability ratio when end-to-end delay is estimated with a probability of 0.95. As shown in Figure 23, when the number of flows is ≤ 10, all test cases were deemed to be schedulable under our schedulability analysis and simulations. When the number of flows was greater than 10, fewer test cases were deemed schedulable by our schedulability analysis. This

difference in schedulability ratio is because simulation results show a conservative upper bound on schedulability ratio while our schedulability analysis considers a pessimistic scenario (where each node requires two transmission time slots). Note that our analysis is only a sufficient test and not an exact test. From this result, we can conclude that our schedulability analysis is close to the upper bound and can be used to determine schedulability.

10 CONCLUSION AND FUTURE WORK

In this article, we have proposed DistributedHART—a distributed real-time scheduling system for WirelessHART networks. In the existing centralized approach, schedules are created centrally and in advance with high degree of redundancy, thus causing a huge waste of energy, bandwidth, time, and memory. DistributedHART obviates the need of creating and disseminating a central global schedule, thereby reducing resource waste and enhancing scalability. In addition, DistributedHART would lead to higher network utilization in the network at the expense of a slight increase in the end-to-end delay of all control loops. Through experiments on a 130-node testbed as well as large-scale simulations, we observe at least 85% less energy consumption in DistributedHART compared to existing centralized approach. In the future, we plan to address challenges arising from the mobility of the nodes in DistributedHART.

REFERENCES

[1] TinyOS. 2005. TinyOS Community Forum. Retrieved from http://www.tinyos.net/.
[2] FieldComm Group. 2007. WirelessHART. Retrieved from https://fieldcommgroup.org/technologies/hart.
[3] Texas Instruments. 2016. CC2420 RF-Transceiver. Retrieved from http://www.ti.com/lit/ds/symlink/cc2420.pdf.
[4] Embedded Systems Lab. 2019. Testbed. Retrieved from http://neteye.cs.wayne.edu/.
[5] K. Agrawal and S. Baruah. 2019. Adaptive real-time routing in polynomial time. In *RTSS*.
[6] R. Brummet, D. Gunatilaka, D. Vyas, O. Chipara, and C. Lu. 2018. A flexible retransmission policy for industrial wireless sensor actuator networks. In *ICII*.
[7] X. Cheng, J. Shi, and M. Sha. 2019. Cracking the channel hopping sequences in IEEE 802.15. 4e-based industrial TSCH networks. In *IoTDI*.
[8] B. Dezfouli, M. Radi, and O. Chipara. 2016. Mobility-aware real-time scheduling for low-power wireless networks. In *INFOCOM*.
[9] B. Dezfouli, M. Radi, K. Whitehouse, S. A. Razak, and H. Tan. 2014. CAMA: Efficient modeling of the capture effect for low-power wireless networks. *Trans. Sensor Netw.* 11, 1 (2014).
[10] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. 2015. Orchestra: Robust mesh networks through autonomously scheduled tsch. In *SenSys*.
[11] A. Ghosh, O. D. Incel, V. S. A. Kumar, and B. Krishnamachari. 2009. Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks. In *MASS*.
[12] Y. Gu, T. He, M. Lin, and J. Xu. 2009. Spatiotemporal delay control for low-duty-cycle sensor networks. In *RTSS*.
[13] D. Gunatilaka, M. Sha, and C. Lu. 2017. Impacts of channel selection on industrial wireless sensor-actuator networks. In *INFOCOM*.
[14] T. He, B. M. Blum, Q. Cao, J. A. Stankovic, S. H. Son, and T. F. Abdelzaher. 2007. Robust and timely communication over highly dynamic sensor networks. *Real-Time Syst.* 37, 3 (2007).
[15] X. Jin, F. Kong, L. Kong, W. Liu, and P. Zeng. 2017. Reliability and temporality optimization for multiple coexisting WirelessHART networks in industrial environments. *Trans. Industr. Electron.* 64, 8 (2017).
[16] P. Levis, N. Lee, M. Welsh, and D. Culler. 2003. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *SenSys*.
[17] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu. 2015. Incorporating emergency alarms in reliable wireless process control. In *ICCPS*.
[18] J. Lu and K. Whitehouse. 2008. Exploiting the capture effect for low-latency flooding in wireless sensor networks. In *SenSys*.
[19] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. 2018. The flooding time synchronization protocol. In *SenSys*.
[20] V. P. Modekurthy, D. Ismail, M. Rahman, and A. Saifullah. 2018. A Utilization-Based approach for schedulability analysis in wireless control systems. In *ICII*.
[21] V. P. Modekurthy and A. Saifullah. 2019. Online period selection for wireless control systems. In *ICII*. IEEE.

[22] V. P. Modekurthy, A. Saifullah, and S. Madria. 2018. Distributed graph routing for wirelessHART networks. In *ICDCN*.
[23] V. P. Modekurthy, A. Saifullah, and S. Madria. 2019. DistributedHART: A distributed real time scheduling system for wirelessHART networks. In *RTAS*.
[24] M. R. Palattella, P. Thubert, X. Vilajosana, T. Watteyne, Q. Wang, and T. Engel. 2014. 6TiSCH wireless industrial networks: Determinism meets IPv6. In *Internet of Things*. Springer.
[25] B. Reed. 1999. A strengthening of Brooks' theorem. *J. Combinat. Theor., Series B* 76, 2 (1999).
[26] I. Rhee, A. Warrier, J. Min, and L. Xu. 2009. DRAND: Distributed randomized TDMA scheduling for wireless ad hoc networks. *Trans. Mob. Comput.* 8, 10 (2009).
[27] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen. 2015. Schedulability analysis under graph routing in WirelessHART networks. In *RTSS*.
[28] A. Saifullah, C. Wu, P. B. Tiwari, Y. Xu, Y. Fu, C. Lu, and Y. Chen. 2014. Near optimal rate selection for wireless control systems. *Trans. Embed. Comput. Syst.* 13, 4s (2014).
[29] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2010. Real-time scheduling for WirelessHART networks. In *RTSS*.
[30] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2011. End-to-end delay analysis for fixed priority scheduling in WirelessHART networks. In *RTAS*.
[31] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2011. Priority assignment for real-time flows in WirelessHART networks. In *ECRTS*.
[32] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2014. Distributed channel allocation protocols for wireless sensor networks. *Trans. Parallel Distrib. Sys.* 25, 9 (2014).
[33] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2015. End-to-end communication delay analysis in industrial wireless networks. *Trans. Comput.* 64, 5 (2015).
[34] M. Sha, D. Gunatilaka, C. Wu, and C. Lu. 2015. Implementation and experimentation of industrial wireless sensor-actuator network protocols. In *EWSN*.
[35] J. Shi, M. Sha, and Z. Yang. 2018. DiGS: Distributed graph routing and scheduling for industrial wireless sensor-actuator networks. In *ICDCS*.
[36] J. Shi, M. Sha, and Z. Yang. 2019. Distributed graph routing and scheduling for industrial wireless sensor-actuator networks. *IEEE/ACM Trans. Netw.* 27, 4 (2019).
[37] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. 2018. Industrial internet of things: Challenges, opportunities, and directions. *Trans. Industr. Inform.* 14, 11 (2018).
[38] F. Terraneo, P. Polidori, A. Leva, and W. Fornaciari. 2018. TDMH-MAC: Real-time and multi-hop in the same wireless MAC. In *RTSS*.
[39] X. Wang, X. Wang, X. Fu, G. Xing, and N. Jha. 2009. Flow-based real-time communication in multi-channel wireless sensor networks. In *EWSN*.
[40] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. 2005. Exploiting the capture effect for collision detection and recovery. In *EmNetS*.
[41] C. Wu, D. Gunatilaka, A. Saifullah, M. Sha, P. B. Tiwari, C. Lu, and Y. Chen. 2016. Maximizing network lifetime of WirelessHART networks under graph routing. In *IoTDI*.
[42] C. Wu, D. Gunatilaka, M. Sha, and C. Lu. 2018. Real-time wireless routing for industrial internet of things. In *IoTDI*.
[43] T. Zhang, T. Gong, C. Gu, H. Ji, S. Han, Q. Deng, and X. S. Hu. 2017. Distributed dynamic packet scheduling for handling disturbances in real-time wireless networks. In *RTAS*.
[44] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu. 2018. Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks. *Trans. Mob. Comput.* 18, 11 (2018).
[45] T. Zhang, T. Gong, Z. Yun, S. Han, Q. Deng, and X. S. Hu. 2018. FD-PaS: A fully distributed packet scheduling framework for handling disturbances in real-time wireless networks. In *RTAS*.
[46] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher. 2005. RID: Radio interference detection in wireless sensor networks. In *INFOCOM*.
[47] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele. 2017. Adaptive real-time communication for wireless cyber-physical systems. *Trans. Cyber-phys. Syst.* 1, 2 (2017).

Received October 2020; revised February 2021; accepted May 2021