

# DistributedHART: A Distributed Real-Time Scheduling System for WirelessHART Networks

Venkata P Modekurthy<sup>†\*</sup>, Abusayeed Saifullah<sup>†\*</sup> and Sanjay Madria<sup>‡</sup>

<sup>†</sup>Department of Computer Science, Wayne State University, Detroit, MI, USA

<sup>‡</sup>Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA

**Abstract**—Industry 4.0 is a new industry trend which relies on data driven business model to set the productivity requirements of the cyber physical system. To meet this requirement, Industry 4.0 cyber physical systems need to be highly scalable, adaptive, real-time, and reliable. Recent successful industrial wireless standards such as WirelessHART appeared as a feasible approach for such cyber physical systems. 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, and 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.

## I. INTRODUCTION

Industry 4.0 is a new industry trend which relies on a data-driven business model to set the productivity requirements of the cyber-physical systems. To meet these requirements, cyber-physical systems need to be highly scalable, adaptive, 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 such cyber-physical systems [7]. 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* (i.e., a time slot when at most one transmission is scheduled to a receiver) or *shared* (i.e., a time slot when 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 another retransmission [2]. A network manager creates the

transmission schedule **centrally** and in advance for all nodes in the network 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 [26], [28] 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, 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, 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 IV). Furthermore, there can be events or emergencies that 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 [16]. 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.

Schedules dissemination in centralized algorithm consumes bandwidth, energy, and time, even for a smaller network or a smaller workload. Typically, hyper-period and length of the schedule increase exponentially with the increase in the number of flows or their periods, which hinders the scalability of the network. Note that, in general, periods can be non-harmonic to ensure stability or control performance [27]. In Industry 4.0, the data-driven business model introduces frequent changes to sampling rates, which requires re-configuration and re-dissemination of schedules. In addition, network dynamics such as a change in channel, node, or link condition requires reconfiguration and re-dissemination of schedules [28]. Such frequent re-dissemination of the schedule consumes very high energy, time, and bandwidth. Thus, a fully centralized WirelessHART scheduling is less suitable for cyber-physical systems, especially those in the domain of Industry 4.0 [6]. Besides, it is typically suitable for deterministic traffic patterns such as periodic traffics or traffics with known arrival pattern.

To address the above limitations, in this paper, we propose a distributed real-time scheduling system for WirelessHART networks. Designing a distributed TDMA protocol with scheduling

\*co-first author

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 paper.

- 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 the real-time performance of a WirelessHART network with a high probability.
- 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. To consider more experimental scenarios, we also evaluated DistributedHART through simulations on TOSSIM [15] using the topology of another testbed [33]. In both experiments and simulations, we observe at least 85% less energy consumption in DistributedHART compared to existing centralized approach.

Section II reviews related work. Section III describes the model. Section IV presents DistributedHART. Sections V and VI present experiments and simulations, respectively. Section VII is the conclusion.

## II. RELATED WORK

Existing work in [36] explored the real-time scheduling for wireless networks. TDMA-based real-time scheduling without multi-channel communication or multi-path graph routing was studied in [9], [12], [17], [20], [24], [44]. Real-time scheduling for data collection in WirelessHART network under tree topology was studied in [35], [40]. For a WirelessHART network, routing [13], [22], [38], schedule modeling [5], priority assignment [30], and distributed channel assignment [31] was studied recently. These works did not focus on the real-time scheduling of packets. Schedulability analysis for industrial wireless networks was studied in [23], [26], [29], [32].

Existing work in [28] showed that the real-time scheduling for flows in WirelessHART networks is NP-hard and proposed real-time scheduling policies that incorporate the transmission conflicts. Scheduling under multiple co-existing wirelessHART networks was studied in [14]. Both papers adopt a fully centralized scheduler that creates a schedule in advance, and they rely on the current WirelessHART scheduling approach with high redundancy. Such an approach causes a huge waste of time, bandwidth, energy, and memory, making it less suitable for dynamics and scalability. In this paper, we aim to address these limitations and propose an online and distributed real-time scheduling system for WirelessHART networks.

Orchestra [10], D<sup>2</sup>-PaS [41], [42] and DiGS [34] 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 flow is in the order of the number of nodes in the network. Such high latencies is less suitable for real-time communications with high workloads. Due to these limitations, Orchestra, D<sup>2</sup>-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.

## III. BACKGROUND AND SYSTEM MODEL

WirelessHART networks operate in the 2.4GHz band and are built based on the physical layer of IEEE 802.15.4. They form 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 both transmit and receive a packet at the same time and can 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 to a controller. The controller sends control commands to the actuators to adjust the physical processes.

Transmissions in a WirelessHART network are scheduled based on a multi-channel TDMA protocol. The network employs the channels defined in IEEE 802.15.4. In large networks spread over a wide area, two distant nodes (which do not interfere with each other) can use the same channel in parallel, 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 should happen in one 10ms time slot. A transmission 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. 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 according to the standard.

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 provide 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 consider there are  $n$  real-time flows in the system denoted by  $F = \{F_1, F_2, \dots, F_n\}$ . The period and deadline of a flow  $F_i$  are denoted by  $T_i$  and  $D_i$ , respectively, where  $D_i \leq T_i$ . Our system is applicable to fixed or dynamic priority assignment. In practice, flows may be prioritized based on deadlines, periods, or criticalities of the loops. In this paper, we use *flow* and *control loop* interchangeably.

Here we give an outline of the current centralized scheduling approach adopted in WirelessHART networks. For a control loop scheduling in the uplink graph, the network manager allocates two dedicated slots for each device on the primary path of a graph route starting from the source. The second dedicated slot on the same path handles retransmissions in case of transmission failures on the first dedicated slot. Then, to handle failures of both transmissions along a link, it allocates a third shared slot on a separate path to handle another retry. The links in the downlink graph are scheduled similarly. Fig. 1 shows an example of transmission scheduling from node  $a$  to an access point (AP) for one control loop in a network of 4 nodes. In Fig. 1, the label on a link refers to its transmission time slot. Node  $a$  is scheduled to use slots 1 and 2 for dedicated transmissions on link  $a \rightarrow b$ . Node  $b$  is scheduled to use slots 3 and 4 for dedicated transmission on link  $b \rightarrow AP$ . To handle transmission failures along  $a \rightarrow b$ , node  $a$  is scheduled to use slot 3 for a shared transmission on link  $a \rightarrow c$ , and so on.

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 our focus. We generate routes using the distributed graph routing algorithm for WirelessHART networks proposed in [22], however DistributedHART works with any graph routing algorithm.

#### IV. THE DESIGN OF DISTRIBUTEDHART

Currently, in a WirelessHART network, a central manager creates a global schedule in advance, which is split into superframes. A *superframe* is a series of time slots representing the communication pattern of a set of nodes and it repeats after the completion of the series. The manager disseminates the superframes among the nodes. Fig. 1 shows an example of transmission scheduling from node  $a$  to an access point (AP) for one control loop in a network of 4 nodes. When there is no transmission failure in the network, the packet will use slot 1 and 3 to reach AP. Although the central manager assigns redundant slots for worst-case scenarios, it leaves 80% of the slots not usable when the network conditions are good.

In an experiment conducted on 30 flows on a testbed of 69 nodes, work in [26] observed that 70% of the slots were unused for a flow in a run. Thus, the network remains largely

underutilized. Although it is crucial to real-time control for handling worst-case scenarios, such scheduling with high redundancy causes a huge waste of energy, bandwidth, time, memory (to store schedule), and is less suitable for network/workload dynamics and scalability. We propose DistributedHART which offers a distributed scheduling system for WirelessHART networks. We describe the design of DistributedHART below.

#### A. Distributed Scheduling

The essence of our proposed approach is to adopt local (node-level) scheduling through a time window allocation among the nodes. The time window allocations happen in a distributed approach and after the route generation. A node schedules its transmissions locally and online in its assigned time window using a real-time scheduling policy. If the transmission on the first slot is successful, the other slots in the time window can be used to transmit other packets, which is not feasible in centralized algorithms. Thus, our approach reduces the wastage and usage of time slots when compared to centralized algorithms. Hence, it consumes less energy and memory, and scales with the number of nodes even under frequent network dynamics. To generate a transmission schedule for nodes, 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 **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 performs a receiver based channel allocation based on vertex coloring as the one used in [25]. Each node is assigned a fixed channel to receive messages; its neighbors use this channel to send messages to it.

After channel allocation, each node performs time window allocation using distributed vertex coloring. In time window allocation, two nodes are assigned different time windows if they remain in conflict even after channel allocation. Note that, DistributedHART allows spatial reuse where many non-conflicting nodes will have the same time window and transmit simultaneously. A time window consists of  $w$  time slots allocated to a node to schedule packets that are available in its queue. We represent the total number of unique time windows by  $\gamma$ , which also represents the worst case chromatic number. We define an *epoch* as the number of time slots after which the time windows repeat, i.e.,  $\gamma \times w$ . Minimizing  $\gamma$  is one important objective of the time window allocations since it minimizes the end-to-end latency for each flow. In the assigned window for a node, it autonomously schedules packets in its queue using a real-time scheduling policy. The local scheduling of packets (within a window) obviates the need for creating a schedule in advance and significantly reduces bandwidth usage. We describe channel and time window allocation, local scheduling policy, and online scheduling as a dedicated and shared slot in the following sections.

In DistributedHART, nodes execute distributed channel and time window allocation during network initialization and under some network dynamics (e.g., when routes are affected).

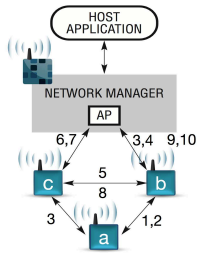


Fig. 1. An example of scheduling in WirelessHART

Workload dynamics or some network dynamics (e.g., that does not affect routes) will not trigger these algorithms, keeping the overhead of DistributedHART low. Due to the low overhead, handling network or workload dynamics is one key advantage of DistributedHART.

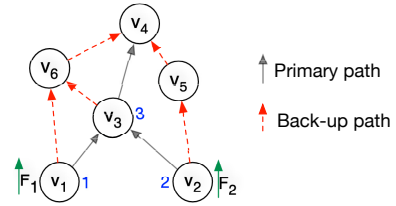
Note that, this paper builds a distributed scheduling system for WirelessHART that needs to accommodate dedicated and shared slots which is not addressed in the existing literature. In DistributedHART, DRAND [25] only performs a distributed vertex coloring to obtain channel and time windows. We can use any graph coloring approach, not necessarily DRAND [25].

#### Channel and Time Window Allocation:

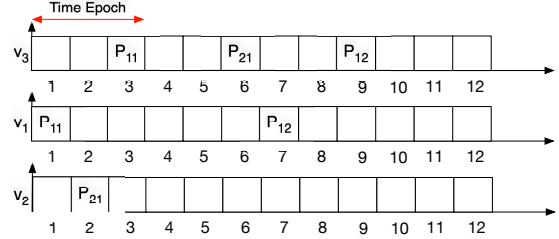
To minimize conflicts between receivers, we first perform channel allocation. We perform receiver based channel allocation for its simplicity in handling channels instead of a link based one. In *receiver based channel allocation*, each node that receives a message is allocated a channel. Neighboring nodes transmit a packet on the receivers channel.

An optimal channel allocation problem is known to be NP-Hard [11]. We use distributed vertex coloring approach called DRAND proposed in [25] on a receiver conflict graph to obtain a heuristic solution. 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 the other. In our method, we assume that each node maintains a physical interference model of the network using Signal-to-Noise plus Interference Ratio (SNIR) such as the RID protocol [43]. Furthermore, each node needs to know the channel conditions between itself and other nodes to construct the conflict graph. This requirement can be met in practice, even in scenarios with fast channel fading or network dynamics since a central manager blacklists these channels at deployment time or through maintenance cycles as specified in the WirelessHART standard. In addition, our approach can leverage existing algorithms specifically designed to detect conflict graphs efficiently in wireless sensor networks [18], [43]. These methods use RSSI measurements to determine and store conflict graphs in a distributed fashion: a node only knows its incoming/outgoing communication and interference edges based on SNIR. Hence, a conflict graph construction method is distributed and efficient in practice. The objective of generating a conflict graph is only to identify nodes that are in conflict with each other, which is done in a distributed way (without generating any tree structure).

Since the number of available channels is limited, a channel allocation does not necessarily solve all conflicts in the network. To remove all conflicts, we perform a time window allocation to each transmitting node. We assign time windows to nodes such that they can transmit to any of its neighbors without interfering (or being interfered by) other nodes. A node transmits a packet during its time window on the receiver's channel. To allocate time windows, we first represent all remaining conflicts using a transmitter conflict graph. 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



(a) Time window allocation example



(b) local scheduling at nodes  $v_1$ ,  $v_2$  and  $v_3$

Fig. 2. An example of local scheduling in DistributedHART

intended recipients. We use distributed vertex coloring using DRAND [25] on transmitter conflict graph to compute non-conflicting transmission windows at each transmitter node. After the time window allocation, each node is also aware of its neighbors time window. Thus, each node knows when to expect a packet and when to sleep.

The number of time slots allocated in a window of a node can depend on the traffic conditions through that node. For simplicity, we assume each node selects an equal length time window ( $w$ ), and  $w$  is a user parameter. However, DistributedHART can handle non-uniform time window lengths by assigning multiple equal length time windows to a node. While assigning a longer time window to a node  $v_i$  may reduce the delays on flows passing through  $v_i$ , it may increase the delay for other flows (due to the increase in  $\gamma$ ). This trade-off between uniform and non-uniform window assignment is out of the scope of this paper and will be studied in the future.

#### B. 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 for two flows  $F_1$  and  $F_2$ , with periods  $T_1 = 6$  and  $T_2 = 12$ , is shown in Fig. 2. For this example, assume that there is only one channel available in the network. Here, nodes  $v_1$ ,  $v_2$ , and  $v_3$  are in transmission conflict as simultaneous transmission from any two nodes results in a collision. To resolve the transmission conflict, each node executes distributed vertex coloring algorithm (DRAND) and selects a unique time window. One possible result of the selection is shown in Fig. 2(a). In this selection, the length of

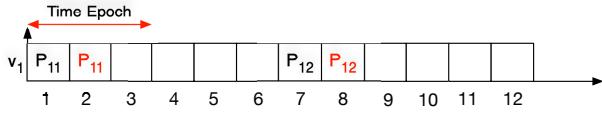


Fig. 3. An example of scheduling as dedicated and shared slot

the epoch is 3, i.e., the packet transmission schedule repeats after 3 time windows. For this selection, local scheduling at each node is shown in Fig. 2(b), where  $P_{11}, P_{12}$  represent the first and second packet of flow  $F_1$ , respectively. At time window 1,  $v_1$  transmits packet  $P_{11}$  to node  $v_3$ . At time window 2,  $v_2$  transmits  $P_{21}$  to node  $v_3$ . At time window 3, node  $v_3$  has packets  $P_{11}$  and  $P_{21}$  from flows  $F_1$  and  $F_2$ . Based on the EDF policy, node  $v_3$  selects packet  $P_{11}$  for transmission on time window 3 and packet  $P_{21}$  is held in queue of node  $v_3$  until next epoch (i.e., time window 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. In our model, the deadline is equal to the sampling period. Hence, rate monotonic policy and DM generate the same schedule.

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 will know the priorities of the packets at its buffer and schedule accordingly. Thus, the network can handle changes due to plant/network dynamics locally, and the manager need not update the entire schedule. Management and diagnostic superframes can run in parallel with the highest priority.

### C. Online Scheduling as Dedicated and Shared Slots

A key **challenge** for DistributedHART is to incorporate both dedicated and shared slots. To use a time slot as a shared slot, many nodes need to transmit to the same node simultaneously. We adopt the following technique to handle shared and dedicated slots. A node can use time slots within its time window as a dedicated or a shared slot. A node can also use time slots outside its time window as a shared slot. An example of dedicated and shared slot scheduling at node  $v_1$  (for network shown in Fig. 2(a) with periods  $T_1 = 12$  and  $T_2 = 24$  slot) is shown in Fig. 3. If the transmission of packet  $P_{11}$  fails on time window 1, then  $v_1$  can use time slots in time window 2 as shared slots even though they lie outside the transmission window of  $v_1$ .

A node  $v_1$  starts the transmission at the beginning of the slot after channel setting if it intends to use it as a dedicated slot. If node  $v_1$ 's transmission on 2 dedicated slots fails, then it does not need to wait for an explicitly assigned shared slot. A shared slot can be either in its assigned window or outside it. In either case, it waits for some time  $\theta$  in the slot during which it keeps sensing the channel. If the channel is busy, then it concludes that the corresponding receiver is involved with a dedicated slot, and it leaves the slot. If it does not sense

any busy channel within  $\theta$  time, then it can use it as a shared slot. If the current slot is within its transmission window, then waiting for  $\theta$  time allows other nodes to use it as shared. If the current slot is outside its transmission window, then waiting for  $\theta$  allows it to know if other nodes are using it as a dedicated slot. In either case, the node makes a small random back-off before transmitting. In some cases, a hidden terminal  $v_2$  may transmit to  $v_3$  as dedicated slot without  $v_1$  knowing it. In these scenarios, we need to ensure that  $v_2$ 's packet ( $P_2$ ) is received correctly as it was transmitted in  $v_2$ 's dedicated slot. We enable **capture effect** [19] of the radio at the receiver to avoid/handle collision between  $P_2$  and  $P_1$  (packet sent by  $v_1$ ).

#### 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], [8]. After this, the radio generates an interrupt and locks to payload reception mode and does not search for preambles. Therefore, **capture effect** [8] 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 1–3dB 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. On the other hand, 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  $\theta$  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  $\theta$  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.

Existing work in [37] demonstrates the effectiveness of capture effect for IEEE 802.15.4 based networks. Here, we experimentally determine values of  $\theta$  (time difference) and Tx power difference to enable the capture effect. We use a setup consisting of 3 TelosB motes (that use radios based on 802.15.4), one receiver and two time synchronized transmitters as shown in Fig. 4. We performed one experiment to determine  $\theta$  and another to determine the power difference for enabling capture effect. For the first experiment, we used a transmission power of 0dBm for both the transmitters and varied  $\theta$  and measured PRR (packet reception rate). We then found  $3ms$  as a good value of  $\theta$ , and using this value, in the second experiment, we decreased the power level of one transmitter while keeping the other transmitter's power at 0dBm. We then found  $3dBm$  as a good value of Tx power difference and using this value (and  $\theta = 3ms$ ), we performed another experiment

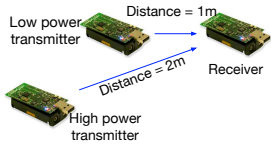
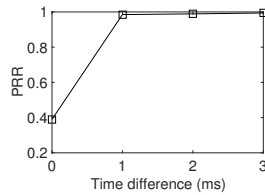
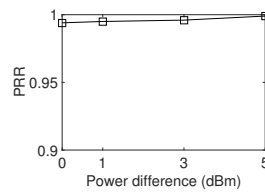


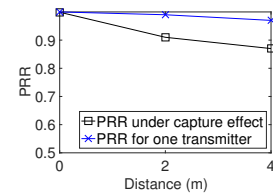
Fig. 4. Capture effect experiment setup



(a) Time difference



(b) Power difference



(c) Distance

Fig. 5. PRR through capture effect with distance, time and power differences between two transmitters

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 1500 packets with a payload of 14 bytes at a period of 10ms on channel 15. We present an aggregate result from 10 iterations. Due to the small variance in the obtained result, we limit the number of iterations to 10.

Fig. 5(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  $\theta$  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). Fig. 5(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 Fig. 5, we set the value of  $\theta$  to  $3ms$  and Tx power difference to  $3dBm$ . Note that, we do not change the time slot structure by delaying the transmission for 3ms. Rather, we capitalize on the remaining 7ms within the current time slot to successfully transmit a packet. Fig. 5(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 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  $\theta$  can be adjusted based on the node placements, which is quite feasible as long as topology changes and/or mobility are not overwhelming.

#### D. End-to-end Delay Analysis for DistributedHART

Here, we present a very simple end-to-end delay analysis as a function of  $\gamma$  and  $w$  for DistributedHART. In DistributedHART, the maximum degree of a node provides an upper bound  $\gamma$ . Thus, the network manager can compute  $\gamma$  without knowing the exact schedule of each node, and use this analysis to predict an upper bound on the delay a flow can experience.

We develop a probabilistic delay bound that considers delay experienced on the primary path of a graph route. In a graph route, the probability of successful transmission on the primary path of a graph route is very high. For example, if we consider

a 4 hop path and PRR on each link is 0.9, then probability of successful reception at actuator is 0.96, as given by Equation (1). The probability of successful reception along a backup path is very small and does not contribute much. Thus, a probabilistic delay bound considering delay caused only in the primary path represents a tight bound with high probability. To simplify the analysis, we assume that the probability of successful transmission for each link is independent of all other links. Let  $\rho_i^k$  be the probability of successful transmission for a link  $k$  in the set of links on the primary path  $E_i$  of a graph route of  $F_i$ . The probability of successful reception of a packet on two dedicated transmissions of link  $k$  is  $1 - (1 - \rho_i^k)^2$ . Then, Equation (1) gives the probability of successful reception of packet through the primary path of a graph route.

$$\mathbb{P}(F_i) = \prod_{k \in E_i} (1 - (1 - \rho_k)^2) \quad (1)$$

DistributedHART is an online scheduling algorithm where a packet experiences a different delay at each node on the path. Thus, we express the total delay (experienced by a flow) as the sum of delays experienced at each node on the primary path. In DistributedHART, nodes adopt real-time scheduling policies like EDF or DM to schedule packets autonomously. Therefore, we can leverage on existing processor demand bound analysis (or response time analysis) of EDF (or DM) algorithms to determine the worst-case delay of flows. In this section, we show the worst case delay computation for flows in DistributedHART with EDF local scheduling using processor demand function proposed in [39]. A similar approach can be used for DM to obtain the end-to-end delay bounds.

Typically, in a WirelessHART network, a link is scheduled on two dedicated slots. Thus, we can say that the worst case execution time of a flow at a node is 2 time slots. To compute the worst case delay, we need to take into account the waiting time between consecutive time windows. Since time window repeats after every time epoch or  $\gamma \times w$  slots, a packet waits for  $\gamma \times w$  time slots for every  $w$  time units delay. A packet also experiences a delay of  $(\gamma - 1) \times w$  time slots between its arrival and first transmission window. Adding these additional delays to response time equation in [39], the total delay experienced by a flow  $F_i$  at node  $v$  is given by Equation (2).

$$\delta_v(F_i) = (\gamma - 1) \times w + 2 + \sum_{F_j \in I_v(F_i)} \max \left\{ 0, 1 + \left\lfloor \frac{\delta_v(F_i) - D_j}{T_j} \right\rfloor \right\} 2 \times \gamma \quad (2)$$

where,  $I_v(F_i) = \{F_j | v \in V_j\}$ . Here  $V_j$  denotes the set of nodes in the primary path of  $F_j$  and  $I_v(F_i)$  denotes the set of flows passing through the node  $v$ .

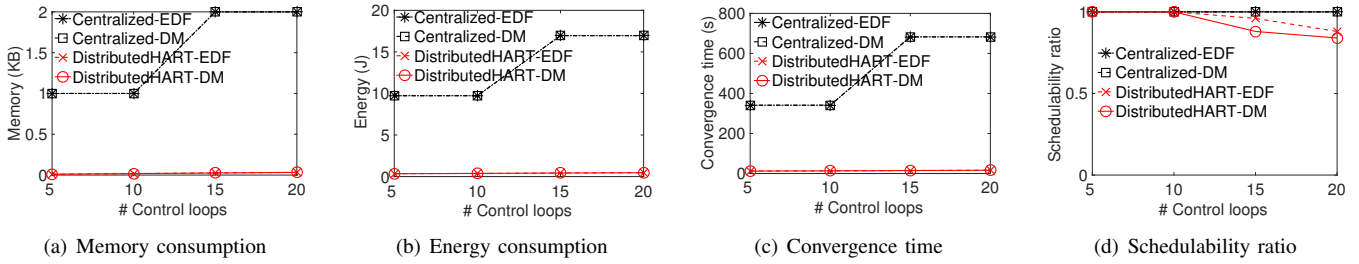


Fig. 6. Experimental result under varying number of flows considering harmonic periods

For implicit deadline flows, the response time  $R_i$  experienced by a control loop  $F_i$  under DistributedHART with EDF scheduling is given by Equation (3) and Equation (2) with a probability  $\mathbb{P}(F_i) = \prod_{k \in E_i} (1 - (1 - \rho_k)^2)$ .

$$R_i = \sum_{v \in V_i} \delta_v(F_i) \quad (3)$$

## V. TESTBED EXPERIMENTS

We implemented DistributedHART on TinyOS 2.2 [1] and evaluated on a 130 node testbed [4] of TelosB mote for real experiments. TelosB 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. The topology of the testbed is shown in [4]. We used a transmission power level of  $-28.7dBm$  to create a 4 hop network. Our DistributedHART implementation consists of multi-channel TDMA MAC protocol. Time is divided into 10ms slots, and clocks are synchronized using the Flooding Time Synchronization Protocol (FTSP) [21]. 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 [22]. For simplicity in implementation and experimentation, network manager computed the channel and time window allocations and then disseminated using TinyOS dissemination protocol library. We then evaluated the online scheduling of DistributedHART.

### A. 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 [28] 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. *Memory consumption* is the average memory consumed per node to store a schedule. *Convergence Time* is the average time taken for all nodes to obtain a schedule excluding the time taken to generate routes. *Energy consumption* is the average energy consumed per node in Joules to generate/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 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 generate/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.

### B. Results

Fig. 6 shows the performance of DistributedHART (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  $2^{13 \sim 16}$  time slots. To decrease the workload on the network, we double the range after adding every 10 flows. As the performance of Orchestra [10] and DiGS [34] is expected to be worse, we evaluated them only in simulation.

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. Fig. 6(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.

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, Fig. 6(b) shows a step increase in average energy consumption similar to memory consumption result. In this experiment, for the sake of simplicity, we computed channel and time window allocation at the central manager for DistributedHART and disseminate the information. In DistributedHART, the length of the schedule was only dependent on the worst case chromatic number and hence, Fig. 6(b) shows the energy consumption is close to constant. We observed that DistributedHART consumes at least 95% less energy than EDF. Similar to energy consumption, convergence time for

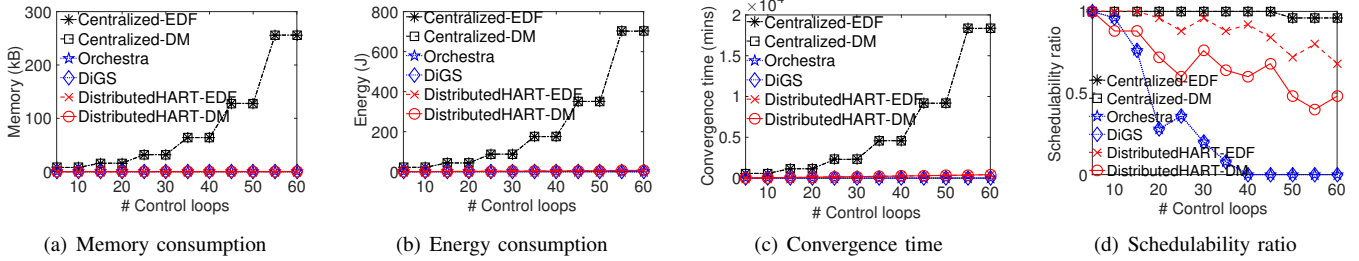


Fig. 7. Performance under varying number of control loops considering harmonic periods

centralized algorithms is also dependent on hyper-period (has a step increase) while convergence time for DistributedHART is almost constant. Fig. 6(c) shows that DistributedHART consumes at least 95% less convergence time than EDF.

3) *Schedulability Ratio*: Centralized algorithms rely on global knowledge of channel/link quality information and harmonic periods which made it feasible for EDF to achieve high schedulability ratio (this may not be feasible with arbitrary periods). In this experiment, we used a very dense deployment. Thus, the worst case chromatic number or  $\gamma$  for DistributedHART was very high. We also observed that  $\gamma$  increases with an increase in the number of flows since more nodes require time windows. This increase in  $\gamma$  increases the end-to-end delay and decreases schedulability ratio.

While a distributed scheduler handles networks/workload dynamics and save 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, convergence time and takes very low memory to store a schedule.

## VI. SIMULATION

### A. Simulation Setup

We perform evaluations considering a topology [33] of 148 nodes through simulations in TOSSIM [15]. We use a large testbed topology of 74 nodes [33]. To scale with the number of nodes, we assume all nodes of that topology are placed in a grid structure and replicate this grid. We add edges between neighboring grids to generate a connected bigger topology for large scale simulation. For the simulations, we follow the fully distributed approach for allocating channel and time windows, as mentioned in DistributedHART. We evaluated the performance of DistributedHART under varying number of control loops, number of nodes, latency, and workload dynamics. 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  $2^{11 \sim 13}$  time slots. To decrease the workload on the network, we doubled the range after adding every 10 flows.

### B. Performance under Varying Number of Control Loops

Fig. 7 shows the performance of DistributedHART (under both fixed & dynamic priority scheduling) and compares it with centralized EDF and DM under varying number of control

loops with harmonic period assignments. We also compare the performance of DistributedHART with Orchestra [10] and DiGS [34]. 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 control loops from 5 to 60. Simulation results for this setup are shown in Fig. 7.

**Memory Consumption.** For centralized EDF and DM, memory consumption at a node is dependent on the hyper-period. Fig. 7(a) shows a step increase in memory consumption since we double the hyper-period for every 10 control loops. For DistributedHART, memory consumption depends only on worst case chromatic number  $\gamma$ . Thus, there is a very small increase in the memory consumption of DistributedHART. In this simulation, we have observed 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, Fig. 7(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% 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 which 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, Fig. 7(d) shows a high schedulability ratio for centralized algorithms. For DistributedHART, smaller  $\gamma$  for initial conditions results in similar schedulability as centralized algorithms. As the number of control loops increases,  $\gamma$  increases linearly which decreases the schedulability ratio. In some cases, we observed that increasing the number of

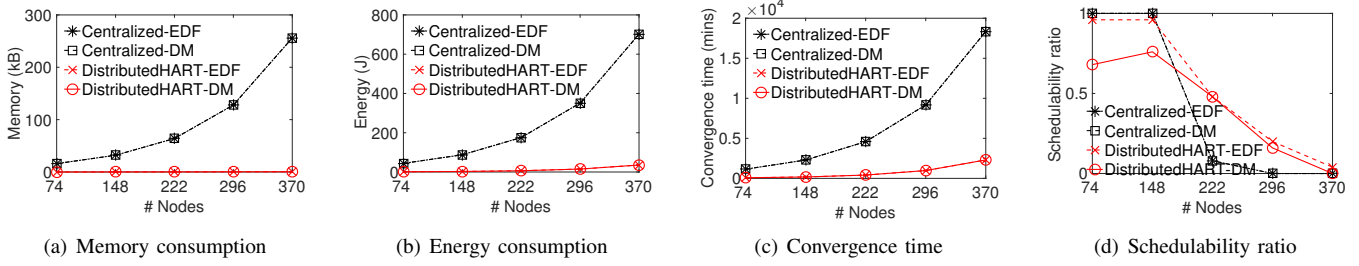


Fig. 8. Performance under varying number of nodes

control loops improved the schedulability ratio. This is due to the fact that sensors, actuators, and periods were selected at random. 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,  $\gamma$  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 Orchestra/DiGS under schedulability ratio, and centralized algorithms under energy, memory and convergence time while achieving similar schedulability ratio.

### C. Performance under Varying Number of Nodes

Here, we show the performance of DistributedHART under varying number of nodes. Fig. 8 shows the simulation results when number of control loops is 20% of the number of nodes. **Memory Consumption.** In this simulation, the number of control loops increases linearly with the increase in the number of nodes, which exponentially increases the hyper-period. Thus, Memory consumption for both centralized and distributed algorithms follow the same result as varying number of control loops with harmonic periods. Fig. 8(a) shows DistributedHART consumes 99% less memory than EDF.

**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. Fig. 8(b) and Fig. 8(c) show DistributedHART consumes at least 94% less energy and 85% less convergence time.

**Schedulability Ratio.** As shown in Fig 8(d), 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.

### D. Performance under Varying Workload Dynamics

Fig. 9 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  $2^{14\sim 18}$  time slots. We use the number of workload changes in one execution as a

parameter for comparing the performance of DistributedHART and centralized algorithms.

**Memory Consumption.** For 1 workload change, hyper-period at most doubles. For more than one change, hyper-period increases by at most 4 times as shown in Fig 9(a). For DistributedHART, change in the hyper-period does not affect the memory consumption and hence, it remains constant.

### 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 re-disseminated to the nodes. As shown in Fig. 9(b), 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, Fig 9(c) 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, 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.

### E. Latency under DistributedHART

Here, we show the latency experienced by each flow. We used a 148 node network topology with 30 flows. We assigned the same period of 4s for all the flows. Note that, we assign the same period for this simulation alone. For all other simulations and experiments, we use different but harmonic periods. We choose to assign the same periods because it allows us to compare the latency of a flow with every other flow.

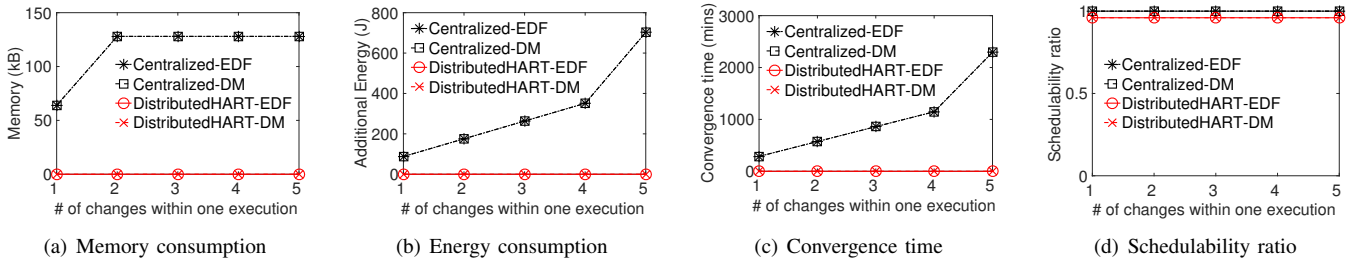


Fig. 9. Performance under varying workload dynamic

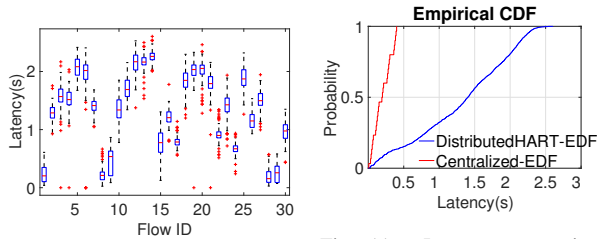


Fig. 10. Latency under DistributedHART

Fig. 11. Latency comparison between Centralized-EDF and DistributedHART-EDF

Fig. 10 shows the latency experienced by each flow in one test case under DistributedHART with EDF local scheduling. We observed an average latency of 1.5s for all flows. We observed that flow 14 experienced the maximum latency in the network of 2.5s. We also observed that flows with smaller route lengths (or routes with less number of hops on the dedicated path) typically have smaller delays, for example, flows with ID 1, 8, and 9. This dependency on the route length is because a packet has to wait for a minimum of  $\gamma$  time units at every node on the path to the destination.

Fig. 11 shows the comparison between the cumulative distribution function of latency observed under DistributedHART-EDF and centralized EDF. We observed, in the simulation, that both centralized-EDF and DistributedHART-EDF use the same number of time slots to transmit all the packets. However, time slots for DistributedHART are  $\gamma$  time units apart from each other, and hence latency of DistributedHART is more than centralized. In this simulation, we also observed packets in that DistributedHART experiences a significant jitter.

DistributedHART 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, DistributedHART uses local information to make scheduling decisions, thereby providing advantages like 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).

### F. 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 VI-B with periods in range  $2^{12\sim 16}$  time slots.

As shown in Fig. 12, when the number of control loops us  $\leq 10$ , all test cases were deemed to be schedulable under our schedulability analysis and simulations. When the number of control loops 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.

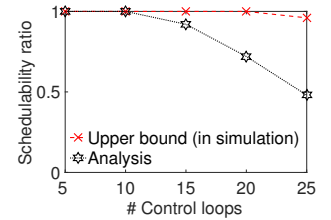


Fig. 12. Schedulability ratio comparison with an upper bound

## VII. CONCLUSION

In this paper, we have proposed DistributedHART - a distributed real-time scheduling system for WirelessHART networks. DistributedHART obviates the need of creating and disseminating a central global schedule thereby reducing resource waste and enhancing scalability. 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. The performance of our schedulability test suggests that there is still room for improvement. In the future, we will derive an improved schedulability test by deriving tighter delay bounds.

## ACKNOWLEDGMENT

This work was supported by NSF through grants CNS-1565751 and CNS-1461914, and by the Graduate School of Wayne State University.

## REFERENCES

- [1] TinyOS Community Forum. <http://www.tinyos.net/>.
- [2] WirelessHART, 2007. <https://fieldcommgroup.org/technologies/hart>.
- [3] CC2420 RF-Transceiver, 2016. <http://www.ti.com/lit/ds/symlink/cc2420.pdf>.
- [4] Testbed, 2019. <http://neteye.cs.wayne.edu/>.
- [5] R. Alur, D’Innocenzo, Johansson, Pappas, and G. Weiss. Modeling and analysis of multi-hop control network. In *RTAS ’09*.
- [6] Ramakrishna Budampati and Soumitri Kolavennu. *Industrial Wireless Sensor Networks: Monitoring, Control and Automation*. Elsevier, 2015.
- [7] Jaime Chen, Manuel Díaz, Luis Llopis, Bartolomé Rubio, and José M Troya. A survey on quality of service support in wireless sensor and actor networks: Requirements and challenges in the context of critical infrastructure protection. *Journal of Network and Computer Applications*, 34(4):1225–1239, 2011.
- [8] Behnam Dezfouli, Marjan Radi, Kamin Whitehouse, Shukor Abd Razak, and Hwee-Pink Tan. Cama: efficient modeling of the capture effect for low-power wireless networks. *ACM Transactions on Sensor Networks (TOSN)*, 11(1):20, 2014.
- [9] Diego Dujovne, Thomas Watteyne, Xavier Vilajosana, and Pascal Thubert. 6tisch: deterministic ip-enabled industrial internet (of things). *IEEE Communications Magazine*, 52(12):36–41, 2014.
- [10] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. Orchestra: Robust mesh networks through autonomously scheduled tsch. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 337–350. ACM, 2015.
- [11] A Ghosh, O Durmaz Incel, V Anil Kumar, and B Krishnamachari. Multi-channel scheduling for fast aggregated convergecast in wireless sensor networks. *MASS’09*, 2009.
- [12] Yu Gu, Tian He, Mingen Lin, and Jinhui Xu. Spatiotemporal delay control for low-duty-cycle sensor networks. In *RTSS ’09*.
- [13] Song Han, Xiuming Zhu, Aloysius K Mok, Deji Chen, and Mark Nixon. Reliable and real-time communication in industrial wireless mesh networks. In *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 3–12. IEEE, 2011.
- [14] Xi Jin, Fanxin Kong, Linghe Kong, Wei Liu, and Peng Zeng. Reliability and temporality optimization for multiple coexisting wireless networks in industrial environments. *IEEE Transactions on Industrial Electronics*, 2017.
- [15] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *SenSys*, 2003.
- [16] Bo Li, Lanshun Nie, Chengjie Wu, Humberto Gonzalez, and Chenyang Lu. Incorporating emergency alarms in reliable wireless process control. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, pages 218–227. ACM, 2015.
- [17] Ke Liu, Nael Abu-Ghazaleh, and Kyoung-Don Kang. JiTS: Just-in-time scheduling for real-time sensor data dissemination. In *PERCOM ’06*.
- [18] Shucheng Liu, Guoliang Xing, Hongwei Zhang, Jianping Wang, Jun Huang, Mo Sha, and Liusheng Huang. Passive interference measurement in wireless sensor networks. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 52–61. IEEE, 2010.
- [19] Jiakang Lu and Kamin Whitehouse. Exploiting the capture effect for low-latency flooding in wireless sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys ’08, 2008.
- [20] Rahul Mangharam, Anthony Rowe, Raj Rajkumar, and Ryohei Suzuki. Voice over sensor networks. In *RTSS ’06: 27th IEEE Real-Time Systems Symposium*, pages 291–302, Dec. 2006.
- [21] Miklós Maróti, Branislav Kusz, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys ’04*.
- [22] Venkata P Modekurthy, Abusayeed Saifullah, and Sanjay Madria. Distributed graph routing for wireless networks. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, page 24. ACM, 2018.
- [23] Venkata Prashant Modekurthy, Dali Ismail, Mahbubur Rahman, and Abusayeed Saifullah. A utilization-based approach for schedulability analysis in wireless control systems. In *2018 IEEE International Conference on Industrial Internet (ICII)*, pages 49–58. IEEE, 2018.
- [24] Maria Rita Palattella, Pascal Thubert, Xavier Vilajosana, Thomas Watteyne, Qin Wang, and Thomas Engel. 6tisch wireless industrial networks: Determinism meets ipv6. In *Internet of Things*, pages 111–141. Springer, 2014.
- [25] Injong Rhee, Ajit Warrier, Jeongki Min, and Lisong Xu. Drand: distributed randomized tdma scheduling for wireless ad-hoc networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 190–201. ACM, 2006.
- [26] Abusayeed Saifullah, Dolvara Gunatilaka, Paras Tiwari, Mo Sha, Chenyang Lu, Bo Li, Chengjie Wu, and Yixin Chen. Schedulability analysis under graph routing in WirelessHART networks. In *Real-Time Systems Symposium, 2015 IEEE*, pages 165–174. IEEE, 2015.
- [27] Abusayeed Saifullah, Chengjie Wu, Paras Tiwari, You Xu, Yong Fu, Chenyang Lu, and Yixin Chen. Near optimal rate selection for wireless control systems. In *RTAS ’12*.
- [28] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. Real-time scheduling for WirelessHART networks. In *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, pages 150–159. IEEE, 2010.
- [29] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. End-to-end delay analysis for fixed priority scheduling in wireless networks. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*, pages 13–22. IEEE, 2011.
- [30] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. Priority assignment for real-time flows in wireless networks. In *2011 23rd Euromicro Conference on Real-Time Systems*, pages 35–44. IEEE, 2011.
- [31] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. Distributed channel allocation protocols for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2264–2274, 2014.
- [32] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. End-to-end communication delay analysis in industrial wireless networks. *IEEE Transactions on Computers*, 64(5):1361–1374, 2015.
- [33] Mo Sha, Dolvara Gunatilaka, Chengjie Wu, and Chenyang Lu. Implementation and experimentation of industrial wireless sensor-actuator network protocols. In *EWSN’15*.
- [34] Junyang Shi, Mo Sha, and Zhicheng Yang. Digs: Distributed graph routing and scheduling for industrial wireless sensor-actuator networks.
- [35] Pablo Soldati, Haibo Zhang, and Mikael Johansson. Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks. In *ECC ’09*.
- [36] J.A. Stankovic, T.E. Abdelzaher, Chenyang Lu, Lui Sha, and J.C. Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7):1002–1022, 2003.
- [37] Kamin Whitehouse, Alec Woo, Fred Jiang, Joseph Polastre, and David Culler. Exploiting the capture effect for collision detection and recovery. In *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on*, pages 45–52. IEEE, 2005.
- [38] Chengjie Wu, Dolvara Gunatilaka, Abusayeed Saifullah, Mo Sha, Paras Babu Tiwari, Chenyang Lu, and Yixin Chen. Maximizing network lifetime of wireless networks under graph routing. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pages 176–186. IEEE, 2016.
- [39] Fengxiang Zhang and Alan Burns. Improvement to quick processor-demand analysis for edf-scheduled real-time systems. In *ECRTS’09*. IEEE.
- [40] Haibo Zhang, Fredrik Osterlind, Pablo Soldati, Thiemo Voigt, and Mikael Johansson. Rapid convergecast on commodity hardware: Performance limits and optimal policies. In *SECON ’10*.
- [41] Tianyu Zhang, Tao Gong, Song Han, Qingxu Deng, and X Sharon Hu. Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks. *IEEE Transactions on Mobile Computing*, 2018.
- [42] Tianyu Zhang, Tao Gong, Zelin Yun, Song Han, Qingxu Deng, and Xiaobo Sharon Hu. Fd-pas: A fully distributed packet scheduling framework for handling disturbances in real-time wireless networks. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–12. IEEE, 2018.
- [43] Gang Zhou, Tian He, John A Stankovic, and Tarek Abdelzaher. Rid: radio interference detection in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 891–901. IEEE, 2005.
- [44] Marco Zimmerling, Luca Mottola, Pratyush Kumar, Federico Ferrari, and Lothar Thiele. Adaptive real-time communication for wireless cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 1(2):8, 2017.