Real-Time Wireless Routing for Industrial Internet of Things

Chengjie Wu, Dolvara Gunatilaka, Mo Sha*, Chenyang Lu
Cyber-Physical Systems Laboratory, Washington University in St. Louis
*Department of Computer Science, State University of New York at Binghamton

***Abstract*—With the emergence of the Industrial Internet of Things (IIoT), process industries have started to adopt wireless sensor-actuator networks (WSANs) for control applications. It is crucial to achieve real-time communication in this emerging class of networks, and routing has significant impacts on end-to-end communication delays in WSANs. However, despite considerable research on real-time transmission scheduling and delay analysis for such networks, real-time routing remains an open question for WSANs. This paper presents a *conflict-aware real-time routing* approach for WSANs, leveraging a key observation that conflicts among transmissions involving a common field device can contribute significantly to communication delays in industrial WSANs, such as WirelessHART networks. By incorporating conflict delays into the routing decisions, conflict-aware real-time routing algorithms allow a WSAN to accommodate more real-time flows while meeting their deadlines. Both evaluations based on simulations and experiments on a physical WSAN testbed show that conflict-aware real-time routing can lead to as much as a three-fold improvement in the real-time capacity of WSANs.**

I. Introduction

Industrial networks connect sensors and actuators in the industrial facilities, such as steel mills, oil refineries, and chemical plants, implementing complex monitoring and control processes. The industrial Internet of Things (IIoT) is a key enabling technology to realize the vision of Industry 4.0. Wireless sensor-actuator networks (WSANs) provide an appealing solution to connect IIoT devices because they require minimal infrastructure. Moreover, wireless modules can be used to easily and inexpensively retrofit existing sensors and actuators in industrial facilities, without running cabling for communication and power [1]. Recent years have witnessed world-wide deployment of WSANs implementing industrial standards, such as WirelessHART [2] and ISA100.11a [3].

Feedback control loops for industrial automation impose stringent end-to-end delay requirements on data communication. To support a feedback control loop, the network periodically delivers data from sensors to a controller and then delivers control commands to the actuators within an end-to-end deadline. The effects of deadline misses in data communication may range from production inefficiency, equipment destruction, system failure for instability, to irreparable financial and environmental damage.

Real-time communication in industrial WSANs is challenging due to their limited bandwidth and multi-hop mesh topologies [1]. Furthermore, industrial standards, such as WirelessHART and 6TiSCH [4] employ Time Slotted Channel Hopping (TSCH), a TDMA-based MAC with channel hopping, to achieve predictable and reliable communication. End-to-end communication delays in such networks are affected by *conflicts* between transmissions involving a common device [5]. Because conflicting transmissions cannot be scheduled in a same time slot, transmission conflicts contribute significantly to the end-to-end communication delays of data flows. Recent studies demonstrated that end-to-end delays of multi-hop flows are heavily influenced by their routes [6]. While real-time routing has received attention in the research community, there has been limited work on routing algorithms specifically designed for recent industrial WSAN standards. Moreover, they generally ignore transmission conflicts in routing decisions, which negatively affect the ability to meet the delay requirements of a large number of real-time flows.

To meet this open challenge, we propose *conflict-aware routing*, a novel approach to real-time routing in WirelessHART networks, a WSAN standard widely adopted in process industries. The key novelty of conflict-aware routing is that it incorporates transmission conflicts and scheduling into its routing decisions to improve real-time performance. Experiments on a physical testbed and in numerical simulations show that conflict-aware routing can lead to as much as a three-fold improvement in the real-time capacity of a WSAN.

The rest of the paper is organized as follows. Section II reviews related work. Section III presents the network model. Section IV discusses the problem formulation. Section V provides a brief review of the existing delay analyses. Section VI presents our real-time routing algorithms. Section VII evaluates our routing algorithms through experiments and simulations. Section VIII concludes the paper.

II. Related Work

The field of wireless sensor networks produced a multitude of sophisticated routing protocols (e.g., RPL [7] and ORPL [8], just to name a few). Designed for general-purpose applications that do not demand real-time performance, these routing protocols were optimized for efficiency and adaptivity to link dynamics. There were efforts to improve the real-time performance of traditional sensor networks in a best-effort manner such as the works presented in [9]–[12]. A common approach adopted by these protocols is to employ localized algorithms to dynamically select the next hop to forward a packet. However, those decentralized approaches cannot

provide end-to-end delay guarantees and are incompatible with recent standards for industrial WSANs.

In contrast to traditional sensor networks, industrial WSAN standards adopt drastically different design choices in order to meet the stringent reliability and real-time requirements of IIoT. For example, in a WirelessHART network, links used for routing are usually more reliable and stable than those in traditional sensor networks. Furthermore, to achieve predictable latency, WirelessHART employs TSCH MAC protocol. Finally, WirelessHART adopts a centralized network manager responsible for computing routes for all flows in the network. Industrial WSANs thus need a new class of routing algorithms.

Several groups proposed algorithms [13]–[15] for reliable graph routing, a multi-path routing approach supported by WirelessHART. Many efforts [16], [17] were geared toward improving the reliability and robustness of industrial WSANs. Other routing algorithms such as [18], [19] aimed to improve energy efficiency and prolong network lifetime. However, these aforementioned algorithms were not targeted at improving the real-time performance of industrial WSANs. The recent 6TiSCH standard combines the RPL routing protocol and the TSCH MAC to support decentralized adaptation [20]. This work is in contrast to our work that is based on TSCH with centralized scheduling.

There exist real-time routing protocols for TDMA-based wireless sensor networks. Xu et al. [21] designed the PRTR protocol to minimize the delay of real-time traffic in a TDMA-based network. However, their end-to-end delay bounds are probabilistic, which is in contrast to many industrial applications that require deterministic delay bounds. Nirjon et al. [22] proposed IAA, a real-time routing algorithm that can guarantee end-to-end delay in TDMA-based networks. IAA employs heuristics to assign shorter paths to flows with tighter deadlines. It does not take into account transmission conflicts, which play a significant role in communication delays in WirelessHART networks. Moreover, in contrast to our work that is based on TSCH and WirelessHART, IAA is designed for a *single-channel* TDMA network that allows concurrent transmissions on the same channel.

III. Network Model

We consider a network model based on the WirelessHART standard [2] that has been widely adopted in process industries. A WSAN consists of a gateway, multiple access points, and a set of field devices (e.g., sensors or actuators). The access points and network devices are equipped with half-duplex radio transceivers compatible with the IEEE 802.15.4 physical layer; together they form a wireless mesh network. A WSAN can use up to 16 channels, as specified in the IEEE 802.15.4 standard. The access points are wired to the gateway and serve as bridges between the gateway and field devices.

The WSAN adopts a centralized network management approach, where the network manager (i.e., a software module running on the gateway or a host connected to the gateway) manages all devices. The network manager gathers the network topology information, and then generates and disseminates the routes and transmission schedule to all network devices. This centralized network management architecture, adopted by the WirelessHART standard, enhances the predictability and visibility of network operations at the cost of scalability.

The WSAN adopts the TSCH MAC on top of the IEEE 802.15.4 physical layer. TSCH is a TDMA-based protocol in which all devices in the network are time synchronized. Time is divided into 10 ms slots, and each slot can accommodate one packet transmission and its acknowledgment. In a slot, only one transmission is scheduled on each channel across the entire network to avoid channel contention, and enhance reliability. Moreover, TSCH supports channel hopping (i.e., each node switches to a new receiving channel in every time slot) to enhance network resiliency through channel diversity. The network operator can blacklist channels with poor quality.

The WirelessHART standard supports two types of routing: *source routing* and *graph routing*. Source routing provides a single route from a source to a destination, whereas graph routing provides multiple redundant routes in a routing graph. Hence, graph routing promotes reliability through route diversity, at the cost of longer latency and higher energy cost [6]. Given our interest in real-time communication, this paper focuses on source routing. In addition, while our algorithms are designed for the WirelessHART standard, the insights and approach may be extended to other WSANs based on TSCH.

IV. Problem Formulation

We consider a WSAN with a set of N real-time flows $\mathcal{F} = \{F_1, F_2, \cdots, F_N\}$. For each flow $F_k = (s_k, d_k, \phi_k, D_k, T_k)$, a source s_k generates a packet at a constant period T_k. A packet must be delivered to a destination d_k through a source route ϕ_k within a relative deadline D_k.

Due to its simplicity and efficiency, fixed priority scheduling is commonly adopted as the real-time scheduling policy in CPU and traditional real-time networks (e.g., Control- Area Networks). A recent study [23] has shown that fixed priority scheduling is an effective policy for real-time flows in WSANs. Hence, we will adopt the fixed priority scheduling framework in this work.

In practice, priorities are assigned based on deadlines, periods, or the criticality of the real-time flows. Priorities of flows remain constant during run-time unless the user requirements or the traffic demands are changed. In this work, we use the deadline-monotonic priority assignment policy, where flows with closer deadlines are assigned with higher priorities. Priorities of flows with the same deadline are randomly assigned. Our routing algorithms can be applied to any fixed priority assignment.

Under a fixed priority scheduling policy, the transmissions of the flows are scheduled in the following way. We assume that all flows are ordered by priorities. Flow F_i has a higher priority than flow F_j if and only if $i < j$. Starting from the highest priority flow, F_1, the following procedure is repeated for every flow F_i in decreasing order of priority. The network manager schedules transmissions of the current flow F_i in

the earliest available time slots and on available channels. A time slot is available if no conflicting transmission is already scheduled in that slot.

The goal of our routing algorithm is to find routes for the flows so that every flow can meet its deadline. The shortest path algorithms based on hop count [13]–[15] are commonly adopted in WSANs. However, as shown in our simulation results, the effectiveness of these algorithms is far from optimal. Based on the insights from end-to-end delay analysis, we propose two new heuristics to assign routes to meet real-time requirements.

V. Conflict Delay Analysis

In this section, we summarize the delay analysis for WSANs. We later use these insights to design our routing algorithms. According to the previous work [23] on delay analysis, a packet can be delayed for two reasons: conflict delay and contention delay. Due to the half-duplex radio, two transmissions conflict with each other if they share a node (sender or receiver). In this case, only one of them can be scheduled in the current time slot. Therefore, if a packet conflicts with another packet that has already been scheduled in the current time slot, it has to be postponed to a later slot, resulting in *conflict delay*. Because a WSAN does not allow concurrent transmissions on the same channel, each channel can accommodate only one transmission across the network in each slot. If all channels are assigned to transmissions of other packets, a packet must be delayed to a later slot, resulting in *contention delay*.

From the delay analyses presented in [23], [24], and our simulations, conflict delay plays a significant role in the end-to-end delays of flows. Furthermore, routing directly impacts conflict delays, whereas contention delays largely depend on the number of channels available. Therefore, in our routing design, we focus only on conflict delay. Saifullah et al. proposed Efficient Delay Analysis (EDA) [23], a state-of-the-art delay analysis algorithm for WSANs. Here, we briefly discuss the EDA algorithm.

We denote the total number of transmissions of flow F_h that conflict with flow F_l as Δ_l^h. Here, flow F_h has a higher priority than flow F_l. Δ_l^h is counted based on the routes of the two flows. Δ_l^h equals the number of links in F_h's route that share nodes with F_l's route, times the number of transmissions scheduled on each link. For example, given F_h's route is $u \to p \to q \to x \to y$, F_l's route is $v \to p \to q \to z$, and the number of transmissions over each link is one, then $\Delta_l^h = 3$, i.e., three links, $\{(u,p),(p,q),(q,x)\}$, in F_h's route share nodes with F_l's route.

Given a time interval of t slots, the number of packets of flow F_h that contribute to the delay of a packet of flow F_l during this time interval is upper bounded by $\lceil \frac{t}{T_h} \rceil$, where T_h is the period of flow, F_h. Therefore, the worst-case conflict delay of flow F_l from all flows with higher priority than F_l in a time interval t can be bounded as

$$\Theta_l(t) = \sum_{h<l} \lceil \frac{t}{T_h} \rceil \Delta_l^h \tag{1}$$

Based on Equation (1), EDA uses an iterative fixed-point algorithm to get the upper bound of F_l's conflict delay. We further break down Equation (1) to learn how much a transmission of high priority flow can delay a low priority flow. Here, we will give an approximation of conflict delay by a single transmission of high priority flow.

A packet of flow F_l can be delayed only within its lifetime D_l (the relative deadline of flow F_l). To simplify Equation (1), we use F_l's deadline as the length of the time window. We further ignore the ceiling function and approximate the conflict delay that F_l can suffer from flow F_h as

$$\Theta_l^h = \frac{D_l}{T_h} \Delta_l^h \tag{2}$$

where Θ_l^h is the total conflict delay that flow F_h brings to flow F_l. Since the total number of transmissions of flow F_h that conflict with flow F_l is Δ_l^h, we approximate the number of conflict delays from a single transmission of flow F_h as $\frac{D_l}{T_h}$. We will use this approximation in our routing design.

VI. Real-Time Routing

In this section, we propose two real-time routing algorithms: Conflict-Aware Routing (CAR) and Iterative Conflict-Aware Routing (ICAR).

A. Conflict-Aware Routing (CAR)

As Section V shows, the conflict delay that a single transmission of a high priority flow, F_h, brings to a low priority flow, F_l, is $\frac{D_l}{T_h}$. We will incorporate this idea into our Conflict-Aware Routing (CAR) algorithm, which picks routes with small conflict delays caused by high-priority flows.

Algorithm 1 presents the pseudocode of our CAR algorithm. The two inputs to the algorithm are (1) a graph $G(V,E)$, where V is the set of devices in the network and E is the set of links in the network, and (2) a flow set $\mathcal{F} = \{F_1, F_2, \cdots, F_N\}$ ordered by priority. We assign routes for flows following the priority order, from the highest to the lowest. Each link (u,v) has a link weight $w_{(u,v)}$ and a delay coefficient $c_{(u,v)}$. For each flow F_l, we update the link weights based on the routes of higher priority flows. If a link (u,v) shares at least one node with a higher priority flow F_h's route, its weight will be increased by $\frac{D_l}{T_h}$, based on Equation (2).

In our algorithm, we implement the link weight update in two steps. In the first step, once the route R_h of a high priority flow F_h is fixed, because every link on R_h will impose $\frac{D_l}{T_h}$ conflict delays on flows with lower priority, we increase the delay coefficient of any link (u,v) that shares at least one node with R_h by $\frac{1}{T_h}$. In the second step, when we calculate the route for a lower priority flow F_l, we update the weight of each link as $w_{(u,v)} = 1 + D_l \cdot c_{(u,v)}$, which takes into account all flows that have higher priority. After updating the link weights, we run Dijkstra's algorithm to find the path ϕ_l

with the smallest path weight. The algorithm terminates when the flow with the lowest priority is assigned a route ϕ_N.

Algorithm 1: Conflict-Aware Routing

```
1  Function CAR(G, F)
       Input  : A graph G(V, E), A flow set
                F = {F_1, F_2, ..., F_N} ordered by priority
                with F_l = (s_l, d_l, T_l, D_l)
       Variable: link weight w, link delay coefficient c
       Output : A route φ_l for each flow F_l
2      for each link (u, v) ∈ E do
3          w_(u,v) = 1;
4          c_(u,v) = 0;
5      for each flow F_l from F_1 to F_n do
6          if l > 1 then
7              for each link (u, v) ∈ E do
8                  w_(u,v) = 1 + D_l · c_(u,v);
9          Find the shortest path φ_l connecting s_l to d_l;
10         Assign φ_l as flow F_l's route;
11         for each link (u, v) ∈ E do
12             if (u, v) shares at least one node with F_l's
                route φ_l then
13                 c_(u,v) = c_(u,v) + 1/T_l ;
```

Now, we discuss the complexity of the CAR algorithm. We first check the complexity for each flow (one iteration within the *for* loop at lines 5-13). The complexity to update the link weights is $O(|E|)$. It takes $O(|E| + |V|\log|V|)$ to execute the Dijkstra's algorithm, and $O(|E|)$ to update the delay coefficients. Then, the total complexity of each flow is $O(|E| + |V|\log|V|)$. Finally, the complexity of our CAR algorithm is $O(N(|E|+|V|\log|V|))$, where N is the number of flows in the WSAN.

B. Iterative Conflict-Aware Routing (ICAR)

By reducing the conflict delay of low priority flows, we can accommodate more flows while meeting their deadlines. However, CAR is based on flow priorities, and high priority flows are not aware of the routes of low priority flows. We further improve the real-time capacity by introducing an approach where high priority flows also take into account the routes of low priority flows to avoid the overlapping routes. This approach gives low priority flows a higher chance to find routes that are schedulable. Hence, in this section, we introduce our Iterative Conflict-Aware Routing (ICAR) algorithm, which is an extension of CAR

ICAR is an iterative algorithm that runs in rounds. Within each round, flows compute their routes one by one. As with CAR, each flow F_l will first update link weights based on the routes of other flows, and then use Dijkstra's algorithm to find the path with the smallest path weight. ICAR then determines if flow F_l with this new route is schedulable under EDA. If yes, this new route is assigned to F_l, and flow F_l is indicated as schedulable. The delay coefficients of the links belonging to the old route of F_l are deducted by $\frac{1}{T_l}$, and the coefficients of the links on the new route are increased by $\frac{1}{T_l}$. Otherwise, flow F_l will not update its route. ICAR terminates when (1) none of the flows update their routes or, (2) all flows are schedulable under EDA or, (3) the number of iterations exceeds the preset threshold M. Otherwise, the algorithm will enter a new round.

The complexity of one round is $O(N(|E| + |V|\log|V| + D_{max}))$. Compared with the complexity of the CAR algorithm $O(N(|E|+|V|\log|V|))$, ICAR incurs an additional complexity of $O(ND_{max})$ for the delay analysis EDA, where D_{max} is the maximum deadline in $\mathcal{F}$. The total complexity of ICAR is $O(MN(|E|+|V|\log|V|+D_{max}))$, where M is the maximum number of rounds. M is no larger than 5 in our simulation.

VII. Evaluation

We evaluate our real-time routing algorithms through both experiments on a physical WSAN testbed and numerical simulations. As discussed in Section II, routing protocols designed for traditional wireless sensor networks are incompatible with the WirelessHART standard, whereas recent efforts on WirelessHART routing have focused on enhancing the reliability [13]–[15] and energy efficiency [18], [19] of multi-path graph routing which usually lead to larger latency than single-path routing like our routing algorithms. Henceforth, we compare our conflict-aware routing algorithms (CAR and ICAR) against Shortest Path Routing (SP) as a baseline for performance evaluation. Note that, while link quality is often incorporated in routing metrics for traditional wireless sensor networks as a routing metric, it is not as useful for WirelessHART networks in which links are usually highly reliable due to aggressive link blacklisting and conservative deployment. In such networks, the shortest path is a reasonable heuristic to reduce latency. Comparing conflict-aware routing against shortest path routing quantifies the benefit of considering transmission conflicts to real-time performance.

Our evaluation includes two parts: (1) experiments on a WSAN testbed and (2) simulations based on network topology traces collected from physical experiments. We evaluate routing protocols with three metrics: (a) *Acceptance ratio*: the percentage of test cases that are deemed schedulable, (b) *End-to-end delay*: the communication delay between the release of a packet from the source and the reception at the destination, and (c) *Execution time*: the total time required to compute routes at the network manager.

A. Experiments on a WSAN Testbed

We evaluate our routing designs on an indoor WSAN testbed consisting of 63 TelosB motes. Figure 1 shows the topology of the WSAN testbed, and the locations of access points, sources and destinations of flows. For each link in the testbed, we measure its *packet reception ratio (PRR)* by counting the number of received packets among 250 packets transmitted on the link. Following the practice of industrial deployment, we add only links with PRRs higher than 90% to the topology of the testbed. The motes in the testbed run a

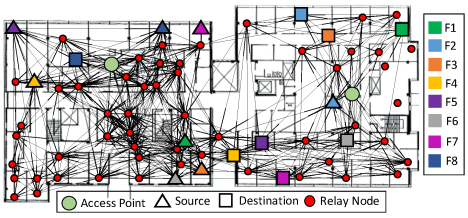
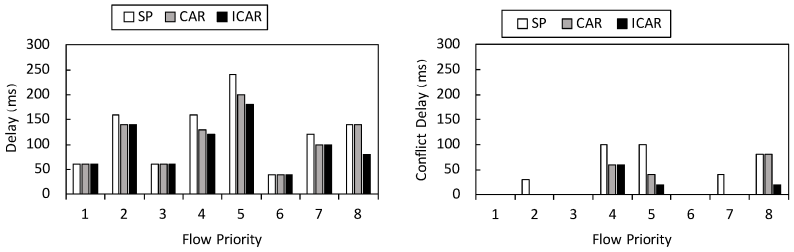


Fig. 1: WSAN testbed topology that includes the locations of access points, sources, and destinations of flows.



(a) End-to-end delays

(b) Conflict delays

Fig. 2: Delays in experiments.

protocol stack [6] implementing source routing and TSCH on top of the CC2420X radio stack.

In our experiment, we generate 8 distinct flows, and use 8 channels for communication. Due to channel hopping, each transmission of any flow can hop through all channels used. The period of each flow is in the range of $2^{4\sim7} \times 10$ milliseconds, which are typical periods used in process industries, as specified in the WirelessHART standard [2]. The length of the hyper-period is 1280 milliseconds. The relative deadline of each flow is equal to its period. We run the experiments long enough that each flow can deliver at least 100 packets.

We evaluate how much our approach improves end-to-end and conflict delays over the performances of SP. Figure 2(a) presents the worst-case end-to-end delays of each flow. CAR and ICAR consistently achieve similar or better end-to-end delays than SP. Moreover, ICAR can further improve the delays of some lower priority flows compared to CAR. Note that since the locations of a source and destination of a flow, and the network topology also impact the end-to-end delay of a flow, some of the lower priority flows may have smaller delays than those of higher priority flows.

We next investigate conflict delays of flows under our approaches and the baseline. Since conflict delay contributes to the end-to-end delay of a flow, reducing the conflict delay can enhance the end-to-end delay. As shown in Figure2(b), under CAR and ICAR, flows incur less conflict delay than under SP. For flow 2 and flow 7, CAR and ICAR can eliminate the conflict delay, while SP still incurs some conflict delay. In addition, ICAR can outperform CAR in many cases since it uses an iterative algorithm that allows higher priority flows to take into account routes of lower priority flows.

B. Simulations based on the WSAN Testbed Topology

To provide a more comprehensive evaluation, we also evaluate our routing algorithms through simulations based on the WSAN testbed topology. The simulator uses the same routing and scheduling algorithms as in our testbed experiments and is written in C++. All simulations are performed on a MacBook Pro laptop with 2.4 GHz Intel Core 2 Duo processor.

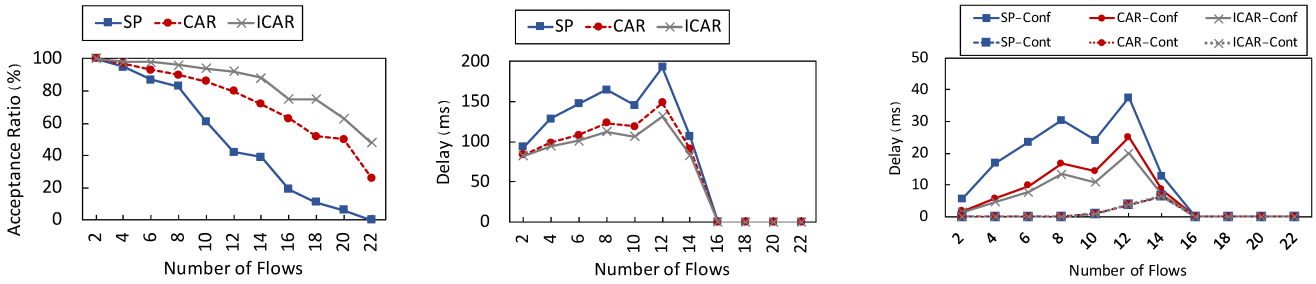
We evaluate our algorithms under different numbers of channels (from 4 to 15). With a given set of channels, we test our routing designs on different numbers of flows by increasing the numbers of source and destination pairs from 2 to 22. The period of the each flow is randomly picked within the range of $2^{4\sim7} \times 10$ milliseconds. The relative deadline of each flow is equal to its period. For the same number of flows, we run 100 tests with randomly generated pairs of sources and destinations. In summary, we perform numerical evaluations on about 10K different configurations. The following results are from simulations with 8 channels.

Figure 3(a) compares the acceptance ratios of CAR, ICAR, and SP in simulations. SP always has the lowest acceptance ratio, and both CAR and ICAR have much higher acceptance ratios. ICAR has a higher acceptance ratio than CAR, which shows the benefit of letting flows with higher priorities be aware of the routes of lower priority flows. Compared to SP, CAR and ICAR can respectively improve the acceptance ratio by 239% and 350% on average.

We then compare the delays of CAR, ICAR, and SP under 8 channels. As shown in Figure 3(c), conflict delay indeed dominates contention delay. Moreover, although CAR and ICAR may lead to routes with longer hop counts, their end-to-end delays are smaller than SP on average as presented in Figure 3(b). This is because CAR and ICAR have fewer conflict delays than SP in all cases.

We obtain similar results with 4-7 and 9-15 channels used. Within all simulations, our CAR and ICAR algorithms improve the acceptance ratio significantly, with smaller conflict delays. When the number of channels is small (4-8), the contention delays can be an important part of the end-to-end delays. However, when more (12-15) channels are used, the contention delays are zero, and conflict delays dominate.

Execution Time: We compare the execution time of SP, CAR, and ICAR in Figure 4. The execution time increases as the number of flows increases in all three algorithms. The execution times of the three routing algorithms follow the order SP<CAR<ICAR. SP has the lowest execution time since it uses the breadth-first search algorithm. ICAR has a higher execution time than CAR because it is an iterative algorithm. The execution time of ICAR is less than 200 ms when the number of flows is 22. According to the WirelessHART Standard [2], when a node loses connectivity to its neighbor for a certain period of time (timeout period), a node will send a keep-alive packet to probe the connection. This timeout period is no less than 30 seconds. If the node fails to



(a) Acceptance ratios. (b) End-to-end delays (c) Conflict (Conf) and contention (Cont) delays.

Fig. 3: Simulation results: acceptance ratio and delays.

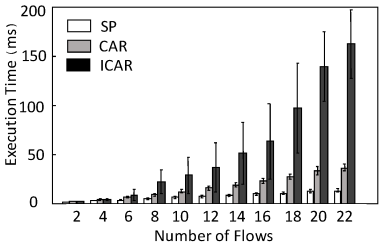


Fig. 4: Execution time

receive a response from its neighbor, a node will issue a path-down alarm to the network manager, which will recalculate a new route. This indicates that the 200 ms execution time is acceptable for the real-world operations.

VIII. Conclusions

As process industries start to embrace WSANs, it becomes critical for WSANs to support real-time communication for IIoT. Due to limited results on real-time routing for industrial WSANs, this paper proposes *conflict-aware routing*, a new approach to real-time routing in industrial WSANs that considers transmission conflict delays in routing decisions. As a result, a WSAN can accommodate more real-time flows while meeting their deadlines. Evaluations based on both testbed experiments and simulations show that conflict-aware routing can lead to up to three-fold improvement in the real-time capacity of a WSAN when compared to shortest path routing.

Acknowledgment

This work is supported, in part, by the NSF through grants 1320921 (NeTS), 1646579 (CPS), and 1657275 (CRII).

References

[1] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," in *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems*, vol. 104, pp. 1013–1024, May 2016.

[2] "WirelessHART Sfpecification," 2007. http://www.hartcomm2.org.

[3] "ISA100: Wireless Systems for Automation." https://www.isa.org/isa100/.

[4] "IPv6 Over the TSCH Mode of IEEE 802.15.4e (6TiSCH)." https://datatracker.ietf.org/wg/6tisch/about/.

[5] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHART Networks," in *RTSS*, 2010.

[6] M. Sha, D. Gunatilaka, C. Wu, and C. Lu, "Empirical Study and Enhancements of Industrial Wireless Sensor-Actuator Network Protocols," *IEEE Internet of Things Journal*, vol. 4, pp. 696–704, June 2017.

[7] "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks." https://tools.ietf.org/html/rfc6550.

[8] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the Tree Bloom: Scalable Opportunistic Routing with ORPL," in *Sensys*, 2013.

[9] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," in *ICDCS*, 2003.

[10] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, "Real-time Power-Aware Routing in Sensor Networks," in *IEEE International Workshop on Quality of Service*, 2006.

[11] J. Heo, J. Hong, and Y. Cho, "EARQ: Energy Aware Routing for Real-Time and Reliable Communication in Wireless Industrial Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 5, pp. 3–11, Feb 2009.

[12] P. T. A. Quang and D. S. Kim, "Enhancing Real-Time Delivery of Gradient Routing for Industrial Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 8, pp. 61–68, Feb 2012.

[13] J. Zhao, Z. Liang, and Y. Zhao, "ELHFR: A Graph Routing in Industrial Wireless Mesh Network," in *ICIA*, 2009.

[14] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and Real-time Communication in Industrial Wireless Mesh Networks," in *RTAS*, 2011.

[15] G. Gao, H. Zhang, and L. Li, "A Reliable Multipath Routing Strategy for WirelessHART Mesh Networks Using Subgraph Routing," *Journal of Computational Information Systems*, vol. 9, March 2013.

[16] J. Niu, L. Cheng, Y. Gu, L. Shu, and S. K. Das, "R3E: Reliable Reactive Routing Enhancement for Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 784–794, Feb 2014.

[17] L. Pradittasnee, S. Camtepe, and Y. C. Tian, "Efficient Route Update and Maintenance for Reliable Routing in Large-Scale Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 13, pp. 144–156, Feb 2017.

[18] C. Wu, D. Gunatilaka, A. Saifullah, M. Sha, P. B. Tiwari, C. Lu, and Y. Chen, "Maximizing Network Lifetime of WirelessHART Networks under Graph Routing," in *IoTDI*, 2016.

[19] S. Zhang, A. Yan, and T. Ma, "An Energy-Balanced Graph Routing Algorithm for WirelessHART Networks," in *IHMSC*, 2013.

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

[21] Y. Xu, F. Ren, T. He, C. Lin, C. Chen, and S. K. Das, "Real-time Routing in Wireless Sensor Networks: A Potential Field Approach," *ACM Transactions on Sensor Networks*, pp. 35:1–35:24, June 2013.

[22] S. M. S. Nirjon, J. A. Stankovic, and K. Whitehouse, "IAA: Interference Aware Anticipatory Algorithm for Scheduling and Routing Periodic Real-time Streams in Wireless Sensor Networks," in *INSS*, 2010.

[23] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-End Delay Analysis for Fixed Priority Scheduling in WirelessHART Networks," in *RTAS*, 2011.

[24] C. Wu, M. Sha, D. Gunatilaka, A. Saifullah, C. Lu, and Y. Chen, "Analysis of EDF Scheduling for Wireless Sensor-Actuator Networks," in *IWQoS*, 2014.