Conservative Channel Reuse in Real-Time Industrial Wireless Sensor-Actuator Networks

Dolvara Gunatilaka and Chenyang Lu
Cyber-Physical Systems Laboratory, Washington University in St. Louis

***Abstract*—Wireless Sensor-Actuator Networks (WSANs) are being adopted as an enabling technology for Industrial Internet of Things (IIoT) in process industries. Industrial applications impose stringent requirements in reliability and real-time performance on WSANs. To enhance reliability, industrial standards, such as WirelessHART, embrace Time Slotted Channel Hopping (TSCH) that integrates channel hopping and TDMA at the MAC layer. Within a network governed by a same gateway, WirelessHART prohibits *channel reuse*, i.e., concurrent transmissions in the same channel, to avoid interference between concurrent transmissions. Preventing channel reuse however negatively affects real-time performance. To meet the demand for both reliability and real-time performance by industrial applications, we propose a *conservative* channel reuse approach designed to enhance the real-time performance while limiting its impact on reliability in WSANs. In contrast to traditional channel reuse designed to optimize performance at the cost of reliability, our conservative approach introduces channel reuse only when needed to meet the timing constraints of flows. Finally, we present an algorithm to detect reliability degradation caused by channel reuse so that channels can be reassigned to further improve reliability. Experimental results based on two physical testbeds show that our approach significantly improves real-time performance while maintaining a high degree of reliability.**

I. Introduction

With the emergence of Industrial Internet of Things (IIoT), wireless sensor-actuator networks (WSANs) are increasingly employed as the communication technology for process monitoring and control due to their benefits in lowering deployment and maintenance cost. In contrast to traditional wireless sensor networks that require only best-effort service, industrial WSANs must meet stringent real-time performance and reliability requirements in dynamic environments. Failing to meet deadlines may lead to safety threats, production inefficiency, and financial loss. To meet such stringent requirements, industrial WSAN standards, such as WirelessHART [1], incorporate a set of salient features. It adopts a centralized network architecture and employs Time Slotted Channel Hopping (TSCH) MAC [2]. To achieve reliable communication, channel reuse is prohibited within a network based on the same gateway, i.e., in the same time slot only one transmission is allowed in each channel. However, this policy can severely affect the real-time performance and the capacity of the network.

While channel reuse has been explored in wireless sensor networks, traditional approaches are unsuitable for industrial WSANs with stringent reliability requirements. Earlier solutions either deal with interference between concurrent transmissions in a best-effort manner or schedule transmissions based on estimated interference. In practice, estimating interference incurs significant overhead and errors, especially in the presence of temporal variations in real-world environments. Moreover, traditional approaches are usually designed to *maximize* channel reuse. This *aggressive* approach to channel reuse, combined with an inaccurate interference estimate, may cause unacceptable degradation in network reliability for industrial applications.

This paper explores channel reuse for industrial WSANs that demand both real-time performance and reliability. The contributions of this work are four-fold.

- We propose a *conservative* channel reuse approach that introduces channel reuse only when necessary to meet the timing constraints of existing data flows.
- We design *Reuse Conservatively (RC)*, a conservative channel reuse algorithm that incorporates channel reuse to meet the real-time performance constraints while mitigating its impact on reliability.
- We develop a classifier to detect links with poor reliability due to channel reuse so that these links can be rescheduled to enhance network reliability.
- We present experimental results based on two testbeds that demonstrate RC significantly improves real-time performance when compared to the standard WirelessHART approach without channel reuse, while maintaining higher reliability than aggressive channel reuse.

The rest of the paper is organized as follows. Section II reviews related work. Section III describes the WirelessHART network model. Section IV introduces the problem. Section V details the channel reuse algorithm. Section VI describes the classifier to detect reliability degradation due to channel reuse. Section VII presents the experimental results. Section VIII concludes the paper.

II. Related Work

Industrial WSANs have been gaining attention in recent years. Previous works have studied different aspects of industrial WSANs including real-time scheduling, delay analysis, routing algorithms, rate selection, and channel selection. Comprehensive reviews of these efforts can be found in [3]–[5]. These works followed existing industrial standards such as WirelessHART, which does not allow channel reuse within a network with the same gateway.

There have been extensive studies on interference measurements in the past years. Masheshwari et al. [6] studied different interference models including hop-based, range-based, SINR-based, and PRR-SINR models. Liu et al. [7] modeled interference based on the PRR-SINR relationship. Qiu et al. [8] used a measurement-based model to estimate throughput, and Zhou et al. [9] proposed a radio interference detection protocol based on a thresholded-SINR model. Although SINR-based models have been shown to be more realistic than other approaches [10], estimating interference based on such models involves multiple measurements (e.g., receiver sensitivity, noise, aggregated interferences, etc.), which, in practice, incurs significant overhead and complexity. Hence, we opt for mitigating interference by increasing the hop distance between concurrent transmissions on the same channel. More importantly, we reduce the impact of channel reuse on reliability by introducing channel reuse only when needed to meet the real-time performance requirements of existing flows.

Several TDMA protocols leverage channel reuse to enhance network capacity. Gronkvist et al. [11] and Brar et al. [12] proposed scheduling policies based on spatial reuse TDMA protocol to maximize network throughput. The works presented in [13]–[16] designed real-time scheduling strategies, which incorporate channel reuse, to optimize network throughput and latency. In contrast, our approach conservatively adopts channel reuse to avoid its impact on network reliability. Notably, since WSAN traffic loads are known by the centralized network manager in industrial settings, we therefore only need enough channel reuse to meet the real-time requirements of existing workloads.

There also exist scheduling algorithms customized for TSCH networks that support channel reuse. Palettella et al. [17] proposed TASA, a centralized traffic-aware scheduling algorithm that improves network latency and power efficiency. Accettura et al. [18] developed DeTAS, a decentralized scheduling protocol for the 6TiSCH [19] network. In contrast to our work, both algorithms *always* reuse channels when transmissions are estimated not to interfere with each other, while our approach introduces channel reuse only when needed to meet the real-time requirements of the workload. Duquennoy et al. [20] presented Orchestra, an autonomous scheduling solution for 6TisCH networks. Transmissions may be mapped to the same slot on the same channel, and hence they must contend for a channel. While Orchestra incurs channel reuse in a best-effort manner, our approach manages channel reuse to reduce interferences by (1) introducing channel reuse conservatively and (2) avoiding interference by judiciously scheduling channel reuse for concurrent transmissions.

A recent direction in wireless sensor networks is to exploit synchronous transmissions. The Glossy-based approach [21] provides a promising alternative to TSCH-based networks. It offers efficient and fast network flooding by exploiting constructive interference and capture effect allowing packets to be received successfully in the presence of concurrent transmissions. Zimmerling et al. [22], to the best of our knowledge, was the first to combine Glossy-based Low-Power Wireless Bus [23] and real-time scheduling. Our solution is based on TSCH that has been adopted by the industry. Our channel reuse approach also relies on the capture effect to allow concurrent transmissions on a channel, and therefore it represents a step to incorporate capture effects into industrial standards such as WirelessHART.

III. Background of WirelessHART

We consider WirelessHART, a widely adopted industrial wireless standard for process monitoring and control. The network comprises a gateway, multiple access points, and field devices. Each device is equipped with a half-duplex radio compatible with the IEEE 802.15.4 standard. The access points are wired to the gateway, which in turn is connected to the network manager. The network manager is responsible for the construction of routes and transmission schedules, which are then distributed to the field devices. The centralized approach enhances visibility and predictability of the network operations.

The WirelessHART standard does not allow channel reuse on a network governed by the same gateway. However, channels may be reused when multiple networks connected to different gateways coexist. In this case, interferences may occur if those networks are located close to each other. Our approach enables channel reuse within a network governed by the same gateway, thereby enhancing the capacity and scalability of the network.

A. TSCH MAC

TSCH is a mode of operation of the IEEE 802.15.4e standard. It is an attractive solution for real-time communication because it offers bounded communication latencies and provides reliable communication. TSCH operates on a 2.4 GHz ISM band, and can use up to 16 IEEE 802.15.4 channels. To improve network resiliency to interference, TSCH includes a channel hopping mechanism where each pair of sender and receiver associating with a transmission switches to a new channel at every time slot. Channels with extreme noises can be blacklisted. While we focus on the WirelessHART protocol as an example for channel reuse, our approach can be generalized for other IIoT standards such as 6TiSCH [19] that runs IPv6 over the TSCH network.

The network is globally time synchronized and time is divided into 10 ms slots accommodating one transmission and its acknowledgement (ACK). A time slot can either be *dedicated* or *shared*. A dedicated slot permits exactly one transmission to be scheduled within a channel. In a shared slot, two senders contend for a channel to send a packet to a common receiver using CSMA/CA. As a result, only one transmission can be received successfully at the receiver. In contrast to a shared slot, our channel reuse policy aims to enable all concurrent transmissions on the same channel in a dedicated slot to be received successfully at their receivers, thereby improving performance and predictability over shared slots.

B. TSCH Scheduling

WirelessHART adopts centralized scheduling where a transmission schedule is generated at the network manager. The network manager has full knowledge of the network topology in all channels and the traffic loads. It first computes routes for network traffic, and then runs a scheduling policy to generate a schedule by assigning a time slot number and a channel offset to each transmission. Each time slot is subject to no *transmission conflict*, and each channel is restricted by *channel constraint*. Two transmissions *conflict* if they share a common node, either a sender or a receiver. This is because of the half-duplex nature of the IEEE 802.15.4 radio, which can perform only one operation (transmission or reception) at a time. Let M be the set of available channels. Traditionally, to avoid interference, each slot can only accommodate $|M|$ concurrent transmissions. The channel offset is within the range $[0, |M| - 1]$. It ensures that no transmission in a slot will use the same channel. Each node computes the channel hopping sequence at run-time following this formula:

$$logicalChannel = (ASN + channelOffset) mod |M|$$

where ASN (Absolute Slot Number) indicates the total number of slots elapsed since the network started. Each pair of sender and receiver then maps the logical channel to a physical channel using a common mapping table. Our approach extends WirelessHART by allowing multiple transmissions to share the same channel in the same slot.

IV. Problem Description

In this section, we first describe the flow model, and define the communication and channel reuse graphs of a network. Finally, we set the objectives of our channel reuse algorithm.

A. Flow Model

A WSAN is shared by a set of end-to-end flows $F = \{F_1, F_2, ...F_n\}$. For each flow $F_i \in F$, a source node S_i releases a packet at a periodic interval P_i. The packet must be delivered to the destination Y_i through a route ϕ_i within the deadline D_i where $D_i \leq P_i$. A route ϕ_i consists of a sequence of links $l_{i1}, l_{i2}, ...l_{ik}$. A transmission over a link l_{ij} is denoted as t_{ij}. Therefore, each flow F_i is characterized by $< S_i, Y_i, D_i, P_i, \phi_i >$. A flow F_i is schedulable if all of its packets are scheduled to meet their deadlines. A set of flows F is schedulable only if all flows in F are schedulable.

In a *fixed priority scheduling policy*, each flow F_i is assigned a fixed priority. A flow F_i has higher priority than a flow F_k if $i < k$. Priorities are commonly assigned based on deadlines or periods. The fixed priority scheduler schedules transmissions of higher priority flows before transmissions belonging to lower priority flows. It allocates the earliest possible slot to each transmission.

B. Communication Graph vs Channel Reuse Graph

We consider two types of graphs in our network: (1) the communication graph that is used for constructing routes from a source to a destination for each flow and (2) the channel reuse graph that is for estimating interferences when channel reuse is adopted.

A communication graph $G_c(V, E)$ consists of a set of network devices V, and a set of links E. We use the Packet Reception Ratio (PRR), a ratio of successfully received packets to the total number of transmitted packets, to assess link quality. Let M be a set of channels used and let $PRR(\vec{xy})_i$ be the PRR of a link $\vec{xy}$ on channel $i \in M$. A bidirectional edge uv is in E if $PRR(\vec{uv})_i \geq PRR_t$ and $PRR(\vec{vu})_i \geq PRR_t$ for all channels $i \in M$ where PRR_t is a link selection threshold. An edge must be bidirectional to support a transmission and its acknowledgement. Due to channel hopping, links hop through all channels so they must be reliable in all channels used.

Our channel reuse policy adopts a hop-based interference model in which concurrent transmissions are allowed if a sender of a transmission is at least k hops away from a receiver of a concurrent transmission. Therefore, we construct a channel reuse graph, which determines the channel reuse hop count between nodes. We represent the channel reuse graph as graph $G_R(V, E)$, where V is a set of network devices and E is a set of edges between two devices. Due to the challenge in directly measuring interference, we define an edge in the channel reuse based on its PRR, which is already collected by the network manager in order to construct the communication graph in the WirelessHART network. Specifically, a bidirectional edge uv exists in E if for ***any*** channel $i \in M$, $PRR(\vec{uv})_i > 0$ or $PRR(\vec{vu})_i > 0$. Note that we consider any channel here because each link cycles through all channels through channel hopping. As the hop count between u and v increases, interference is likely to diminish. We denote a minimum channel reuse hop count between any node u and v as ρ.

C. Objectives of Channel Reuse Policy

Given a real-time WSAN and a set of flows, the objective of our channel reuse algorithm is to introduce concurrent, channel-sharing transmissions such that all flows will meet the deadlines. Note that, if no channel reuse is needed to meet the deadlines, no concurrent transmissions will be scheduled on the same channel. Channel reuse occurs only when needed to make the deadlines. To further mitigate interference caused by channel reuse, our secondary objectives are to (1) increase the hop distance between transmissions sharing a channel and (2) reduce the number of concurrent transmissions per channel. The larger the hop distance between concurrent transmissions, the more likely for both transmissions to succeed due to capture effect. Scheduling more transmissions on the same channel can make transmission failures more likely because interferences are cumulative [6] [7]. Therefore, it is desirable to schedule fewer concurrent transmissions on each channel.

V. Channel Reuse Algorithm

In this section, we first introduce the channel reuse constraints, the concept of flow laxity, and then provide a detailed description of our channel reuse algorithm.

A. Channel Reuse Constraints

The channel reuse constraints set up conditions that a transmission must satisfy in order to be assigned a specific time slot and channel offset by a scheduler. The maximum number of slots in a schedule is determined by the hyper-period of a set of flows F, which is the least common multiple of the periods of all the flows. Let $|M|$ be the number of channels used. A channel offset is in the range $[0, |M| - 1]$. Let $H(F_i)$ be a set of flows with higher priority than F_i. At slot s, T_s denotes a set of scheduled transmissions of $H(F_i)$. For each channel offset associated with slot s, $T_{sc} \subset T_s$ is a set of scheduled transmissions assigned to channel offset c. A transmission t_{ij} can be scheduled in s, if t_{ij} does not conflict with other transmissions in T_s (see Section III-B). If two transmissions are assigned to a same slot and channel, they must be at least ρ hops apart on $G_R(V, E)$. Therefore, any transmission t_{ij} of F_i can be allocated a slot s and a channel offset c if the following *channel reuse constraints* are satisfied:

1) *Transmission conflict*: $t_{ij} = uv$ does not conflict with $xy \in T_s$: $u \neq x$ and $u \neq y$ and $v \neq x$ and $v \neq y$, for all $xy \in T_s$.
2) *Channel constraints*:
    a) If channel reuse is not allowed ($\rho = \infty$): channel offset c must not have been assigned to any transmission, $T_{sc} = \emptyset$.
    b) If channel reuse is allowed ($\rho < \infty$): given $t_{ij} = uv$ and $xy \in T_{sc}$: u must be at least ρ hops from all y and all x must be at least ρ hops from v.

B. Flow Laxity

A flow laxity helps anticipate if a packet belonging to a flow can be delivered to meet its deadline or not. It indicates how much delay a packet can tolerate. The higher the flow laxity, the more a transmission can be delayed to later slots. The concept of laxity has been used in [15] and [24] as a heuristic for real-time scheduling. Our algorithm uses flow laxity to assess if channel reuse is needed or if the current channel reuse hop distance will allow a flow to meet its deadline or not. Given a slot s that meets the *channel reuse constraints* for t_{ij}, a set of remaining transmissions of a flow F_i after t_{ij} is denoted as $T_{post,t_{ij}}$. A *flow laxity* is defined as the number of remaining time slots minus the number of slots required to schedule transmissions in $T_{post,t_{ij}}$. Let d_i be a slot corresponding to F_i's deadline D_i. For each transmission $t \in T_{post,t_{ij}}$, we estimate the number slots in an interval $[s + 1, d_i]$ that contains transmissions conflicting with t denoted as q^t_{s+1,d_i}. Hence, the total number of slots in $[s + 1, d_i]$ conflicting with transmissions in $T_{post,t_{ij}}$ is $\sum_{t \in T_{post,t_{ij}}} q^t_{s+1,d_i}$. The minimum number of slots required to schedule $T_{post,t_{ij}}$ is $|T_{post,t_{ij}}|$. Therefore, the laxity of F_i when scheduled t_{ij} at s is:

$$[s - d_i] - \sum_{t \in T_{post,t_{ij}}} q^t_{s+1,d_i} - |T_{post,t_{ij}}| \qquad (1)$$

where $s - d_i$ is the number of slots in $[s + 1, d_i]$, and $\sum_{t \in T_{post,t_{ij}}} q^t_{s+1,d_i}$ is the estimated number of conflicting slots that cannot be allocated to transmissions in $T_{post,t_{ij}}$. Our channel reuse policy attempts to select slot s for t_{ij} such that F_i's laxity is no less than 0. This means that after scheduling t_{ij} at slot s, there are enough slots to accommodate the remaining transmissions of F_i such that F_i can deliver a packet to the destination within its deadline.

C. Reuse Conservatively Algorithm (RC)

In this section, we provide a detailed description of our *Reuse Conservatively Algorithm (RC)*, which integrates our channel reuse policy with a real-time fixed priority scheduling as presented in Algorithm 1. The input of the algorithm is a set of flows F, a channel reuse graph $G_R(V, E)$, and the minimum channel reuse hop count threshold ρ_t. A larger ρ_t will lead to more reliable communication, but may reduce the capacity of the network since channel reuse will be adopted in a more restrictive fashion. In practice, to maintain reliability, a network operator may select the largest channel reuse hop count under which the workload is schedulable. When the workload is schedulable without channel reuse, RC will not introduce channel reuse (i.e., the hop count is effectively infinity). The output is a transmission schedule S. We schedule flows based on their priorities from the highest to the lowest priority. The ρ variable keeps the current minimum channel reuse hop distance. For each transmission t_{ij}, ρ is first initialized to ∞, which means no channel reuse is allowed.

For every transmission t_{ij}, the algorithm first executes $findSlot()$. This function finds the earliest slot s and channel offset c that comply with the *channel reuse constraints* (see Section V-A) given the channel reuse hop count ρ. It then calculates flow laxity following Equation 1 given t_{ij} is scheduled at s. If the computed $laxity \geq 0$, then t_{ij} is scheduled at slot s and assigned channel offset c, and the algorithm proceeds to the next transmission. Otherwise, the algorithm introduces channel reuse for t_{ij} . We define the network diameter as the maximum shortest distance between two nodes in the network. When channel reuse is adopted, ρ is set to λ_R (i.e., the network diameter of $G_R(V, E)$), which gives the maximum channel reuse distance for this network. The algorithm decreases the channel reuse hop count ρ and repeats the $findSlot()$ and $calculateLaxity()$ steps until the calculated $laxity \geq 0$ or until ρ is less than the minimum threshold ρ_t. The algorithm terminates when all transmissions of a flow set F are scheduled, or when any scheduled transmission fails to meet its deadline, which deems F unschedulable.

The algorithm adopts channel reuse conservatively, i.e., when $laxity < 0$. It introduces channel reuse with a larger hop count first to mitigate interference. Note that when there exist multiple channel offsets in slot s that meet the channel reuse constraints, the algorithm chooses a channel with the fewest number of scheduled transmissions to reduce channel contention.

Complexity: There can be at most $N * L$ transmissions scheduled where N is the size of F, and L is the maximum path length among all flows. For each transmission, $findSlot()$ is executed at most $\lambda_R + 1$ times. For each slot, the algorithm checks the channel reuse constraints, which takes $N * (|E| + |V|log|V|)$ because there can be no more than N transmissions scheduled in each slot, and channel reuse hop count computation on a graph $G_R(V, E)$ takes $|E|+|V|log|V|$. The upper-bound complexity of $findSlot()$ is $O(\lambda_R * P_{max} * N * (|E| + |V|log|V|))$ where P_{max} is the maximum possible number of slots in a schedule, which associates with the hyper-period of F. The total complexity of our channel reuse policy integrating with a fixed priority scheduling algorithm is $O(N^2 * L * \lambda_R * P_{max} * (|E| + |V|log|V|))$.

Algorithm 1: Reuse Conservatively (RC)

```
Input  : A flow set $F$ ordered by priority, a conflict
         graph $G_R(V, E)$, the minimum channel reuse
         hop distance $\rho_t$
Output: A transmission schedule $S$
for each flow $F_i$ from $F_1$ to $F_N$ do
    $\rho \leftarrow \infty$ ;
    for each transmission $t_{ij}$ of $F_i$ do
        while $\rho \geq \rho_t$ do
            $< s, c >= findSlot(t_{ij}, \rho)$;
            $laxity = calculateLaxity(t_{ij}, s)$;
            if $laxity \geq 0$ then
                break;
            else
                if $\rho = \infty$ then
                    $\rho \leftarrow \lambda_R$;
                else
                    $\rho \leftarrow \rho - 1$;
    if $s \leq d_i$ then
        Schedule $t_{ij}$ at $< s, c >$;
    else
        return $\emptyset$ ;
return $S$ ;
```

VI. Detecting Reliability Degradation Caused by Channel Reuse

While conservative channel reuse can effectively mitigate the impacts of channel reuse on reliability, it may lead to reliability degradation in some cases. In such cases, we need to detect links that suffer poor reliability caused by channel reuse, so that these links can be reassigned to different channels or time slots. A naive approach is to use a PRR threshold to identify links affected by channel reuse, i.e., if the PRRs of the links associated with channel reuse are lower than the PRR threshold, then channel reuse may have caused packet loss over these links. However, it is important to note that channel reuse is not the only possible cause of transmission failures. Dynamic changes in channel or environmental conditions and external wireless interference may lead to changes in link reliability as well. It is therefore important to distinguish whether reliability degradation is caused by channel reuse or other factors so that the network manager can resolve the link quality degradation according to the root causes. For example, if a link's PRR drops because of external interference instead of channel reuse, removing channel reuse involving that link would not necessarily improve its quality. Henceforth, we propose a method to distinguish whether a link's reliability degradation is attributed to channel reuse. We address this challenge by comparing a link's PRR in the time slots when it shares a channel with others and its PRR in the other time slots when there is no channel reuse. Intuitively, if a link is highly reliable in a contention-free channel, but performs poorly when channel reuse is introduced, then the link's reliability is degraded due to channel reuse. Moreover, we minimize detection overhead by leveraging statistics already collected by the network manager of a WirelessHART network.

Each wireless node maintains a neighbor table, which contains statistics pertaining to connectivity between a node and its neighbors, e.g., PRRs, in all channels used. A node learns these statistics based on regular data transmissions and periodic neighbor-discovery packets, which are used to update the topology of the network. The network manager must also reserve enough slots for each node to broadcast neighbor-discovery packets in all channels used. Since the network topology does not change so frequently, these periodic neighbor-discovery packets do not need to be scheduled very often. Based on the schedule computed by the network manager, each node has knowledge of time slots, channel offsets, and its links, which are associated with channel reuse. For each link involved in channel reuse, a node gathers link statistics in two cases: (1) when a transmission on the link occurs without channel reuse and (2) when a transmission over the link shares a channel with other transmissions. A node periodically reports these statistics to the network manager, which then detects links affected by channel reuse based on the following statistical test.

To detect links affected by channel reuse, our detection policy adopts the *Kolmogorov-Smirnov test (K-S test)* [25] to quantify the difference of link qualities under these two cases. *K-S* test offers the benefits of making no assumption about the distribution of data and making no restriction on the data set size. A two-sample *K-S* test evaluates the difference between two Empirical Cumulative Distribution Functions (ECDF) of two data sets. In our case, we compare the PRR distributions of a link l_i when a transmission over l_i is scheduled with channel reuse (denoted $PRR\_DIST_r(l_i)$), and when a transmission takes place without channel reuse (denoted $PRR\_DIST_{cf}(l_i)$). In a statistical test, under the *null hypothesis*, there is no significant difference between two sample sets. *K-S* test computes the maximum distance between ECDF of $PRR\_DIST_r(l_i)$ and $PRR\_DIST_{cf}(l_i)$. It calculates the *p-value* $\in (0, 1)$, which determines the result of the hypothesis test. The output of the *K-S* test is either *accepting* or *rejecting* the null hypothesis at α significance level. The null hypothesis is rejected when the p-value$< \alpha$. Therefore, for our detection policy, if the outcome of the *K-S* test is *reject*, then we conclude that channel reuse degrades the reliability of link l_i since the $PRR\_DIST_r(l_i)$ differs significantly from $PRR\_DIST_{cf}(l_i)$. Otherwise, channel reuse only has subtle or no impact on l_i.

Our detection policy considers only links associated with channel reuse. Its goal is to identify links whose reliability is degraded by channel reuse. Let $PRR_r(l_i)$ be the PRR of $PRR\_DIST_r(l_i)$. The policy first uses a PRR threshold PRR_t to identify links that fail to meet the reliability requirement, i.e., links with $PRR_r(l_i) < PRR_t$. It then performs *K-S* test on such links to determine if reliability degradation is caused by channel reuse or not. The following is the detection policy:

- If $PRR_r(l_i) < PRR_t$, then perform *K-S* test on $PRR\_DIST_r(l_i)$ and $PRR\_DIST_{cf}(l_i)$.
  - If *K-S* test returns *reject*, then channel reuse degrades link reliability.
  - If *K-S* test returns *accept*, then l_i fails to meet the reliability requirement due to other reasons.
- Otherwise, l_i meets the reliability requirement.

VII. Evaluation

We evaluate our algorithm in terms of the following aspects: (1) real-time performance (2) algorithm efficiency, i.e., the amount of channel reuse and channel reuse hop count (3) execution time (4) network reliability (5) applicability of our detection policy in identifying links whose reliability is degraded by channel reuse. To demonstrate the generality of our approach, (1)-(3) are evaluated based on the collected topologies from: (a) the 80-node Indriya testbed, deployed at the National University of Singapore [26] (b) the 60-node WUSTL testbed deploying across 3 floors. The topology information includes the PRRs of all links in the network in all 16 channels. Based on the network topology and the traffic requirements, the network manager constructs a communication graph (with PRR_t = 0.9) and a channel reuse graph. Then, it generates routes and a schedule, which all field devices must follow during run time. To measure reliability performance, evaluation (4) and (5) are conducted on the WUSTL testbed, which runs the WirelessHART protocol on TinyOS 2.1.2 and TelosB motes [27].

We randomly generate a set of flows F by varying source and destination nodes. Each flow set contains two access points, which are nodes with a high number of neighbors. Following common characteristics of process monitoring and control applications in process industries, sources of flows generate packets periodically, and the periods of flows are harmonic. The periods are uniformly selected from the range $P = [2^x, 2^y]$ where $x < y$. Given the period range $P = \{2^x, 2^{x+1}, ...s^{x+k}\}$, if a flow F_i is assigned with $P_i = 2^j$, then its deadline D_i is randomly picked from a range $[2^{j-1}, 2^j]$.

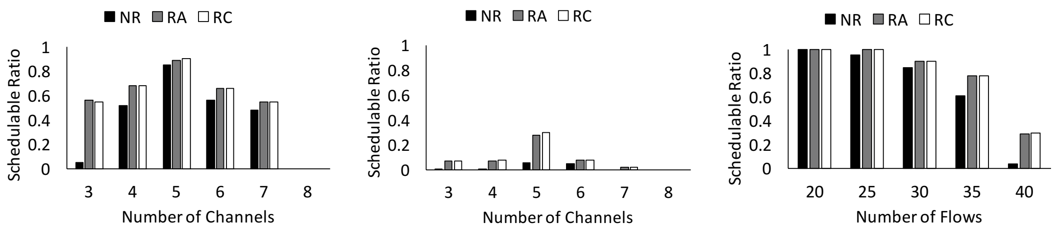
For each flow $F_i \in F$, the network manager generates a single route from a source to a destination node based on the shortest path algorithm and the types of traffic. We consider two types of traffic for control applications: (1) *centralized* traffic: a packet is generated by a source (sensor), and routed to a controller through an access point wired to a gateway; then a control message issued by a controller, running on a computer behind the gateway, is delivered to a destination (actuator) and (2) *peer-to-peer* traffic: to enhance scalability, controllers are implemented on field devices in the network, and a data packet can be delivered directly from a source to a destination node without going through the gateway. We run experiments under source routing, which enables every link along the route to retransmit a packet if the first transmission attempt fails. Hence, a scheduler must reserve one more time slot for every transmission over a link.

We adopt the *Deadline Monotonic (DM)* algorithm, which is a commonly used fixed priority scheduling policy for real-time systems. Under DM, a flow with the shortest deadline has the highest priority. Our algorithm RC is compared against: (1) DM with no channel reuse (NR): only one transmission can be scheduled on each channel, and (2) DM with aggressive channel reuse (RA): a channel is reused whenever possible, i.e., a scheduler schedules transmissions at the earliest slot and on a channel with at least ρ channel reuse hops. RA adopts similar channel reuse approaches as traditional channel reuse strategies, which always introduce channel reuse when an interference model estimates that concurrent transmissions will not interfere with each other. In addition, RA is also similar to TASA [17], a centralized TSCH scheduling approach that *always* allows transmissions to share a channel when $\rho \geq 2$. For a fair comparison, we set the minimum channel reuse distance ρ_t for RC to 2.

A. Real-Time Performance

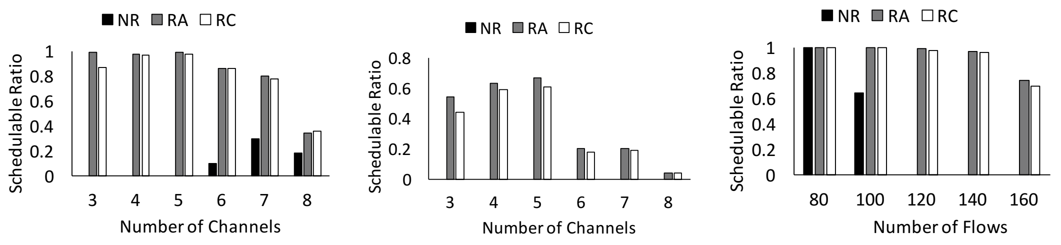
We evaluate the effectiveness of our channel reuse policy in enhancing real-time performance by using a schedulable ratio as a metric. A schedulable ratio is a fraction of flow sets that is schedulable. For each experiment, we generate 100 different flow sets. Figures 1 and 2 present schedulable ratios based on the Indriya testbed topology. We compare RC against NR and RA under a varying number of channels, flows, and periods. Figures 1(a), 1(b), 2(a), and 2(b) show RA and RC consistently outperform NR, especially when there are a limited number of channels, i.e., when the number of channels is from 3 to 5. Under the peer-to-peer traffic and the period range $P = [2^{-1}, 2^3]$, NR cannot successfully generate any schedule for any number of channels as shown in Figure 2(b). Note that increasing the number of channels does not always monotonically improve schedulability. Although more channels enhance network capacity, at the same time, they can degrade route diversity in the network and introduce more transmission conflicts as observed in our previous study [28].

Both RC and RA also yield higher schedulable ratios compared to NR as the number of flows increases. Figure 1(c) shows, under the centralized traffic, RA and RC achieve the maximum of a 7.5X higher schedulable ratio than NR when the number of flows is 40. Figure 2(c) also illustrates that with the peer-to-peer traffic, as the number of flows reaches 120, NR can no longer successfully generate schedules, while RA and RC still obtain almost a 100% schedulable ratio. We also



(a) #flows=30, $P = 2^{0\sim2}$ (b) #flows=30, $P = 2^{-1\sim3}$ (c) #ch=5, $P = 2^{0\sim2}$

Fig. 1. Schedulable ratios under varying number of channels and flows based on the centralized traffic (Indriya).

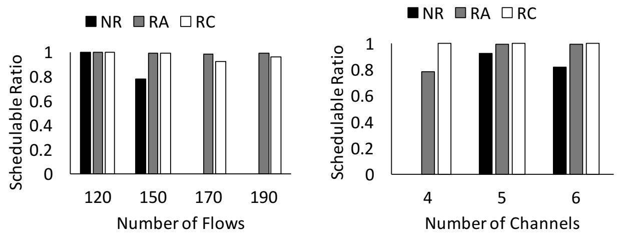


(a) #flows=120, $P = 2^{0\sim2}$ (b) #flows=120, $P = 2^{-1\sim3}$ (c) #ch=5, $P = 2^{0\sim2}$

Fig. 2. Schedulable ratios under varying number of channels and flows based on the peer-to-peer traffic (Indriya).

note that a flow path length based on the peer-to-peer traffic is approximately two times shorter than the path length of a flow under the centralized traffic. The results demonstrate the benefits of channel reuse in enhancing scalability of the network under heavy workloads and few available channels. Notably, with lower traffic load, indeed channel reuse is not needed since flows can be scheduled easily to meet their deadlines. We repeat the experiments based on the WUSTL testbed topology, and observe similar results (see Figure 3).

It can also be noticed that channel reuse is more effective under the peer-to-peer traffic since it enables both RA and RC to outperform NR with a significantly larger margin compared to the centralized approach. The centralized traffic enforces all packets to be routed through access points. As a result, it potentially introduces more transmission conflicts near the access points, which make channel reuse more difficult. Moreover, in most cases, RC can achieve a schedulable ratio comparable to RA. However, in the worst case, RC obtains a 22% lower schedulable ratio than RA as shown in Figure 3(a).



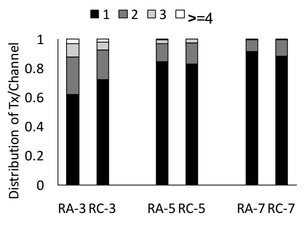
(a) #flows=150, $P = 2^{0\sim2}$ (b) #ch=5, $P = 2^{0\sim2}$

Fig. 3. Schedulable ratios under varying number of channels and varying number of flows based on the peer-to-peer traffic (WUSTL).

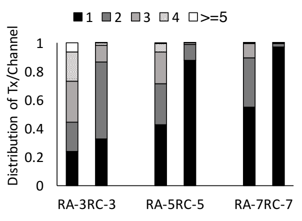
B. Algorithm Efficiency

We validate if RC indeed meets its goals in: (1) reducing channel reuse and (2) increasing channel reuse hop distance when channel reuse is adopted. We quantify the number of transmissions per channel (Tx/channel), and the minimum channel reuse hop count among senders and receivers of concurrent transmissions on each channel. The results are obtained from the experiments in Section VII-A. Figure 4 shows the distribution of different numbers of transmissions per channel of RA and RC. With the peer-to-peer traffic (Figure 4(b)), compared to RA, RC attains a higher proportion of 1 Tx/channel (no channel reuse), especially when there are more available channels. In addition, the results demonstrate that if a channel is reused, RC schedules fewer transmissions on a channel than RA. While, under the centralized traffic (Figure 4(a)), both RA and RC achieve comparable results, except when the number of channels is 3, in which RC achieves a $\sim$10% higher proportion of 1 Tx/channel than RA.

Figures 5(a) and 5(b) plot the distribution of channel reuse hop count under the peer-to-peer and centralized traffic, respectively. Again, the peer-to-peer approach allows RC to increase channel reuse hop count. For every number of channels, RC has the highest proportion for 3-hop count, while RA has the highest proportion for 2-hop count. With the centralized traffic, both RA and RC are dominated by 2-hop channel reuse. RC demonstrates that it meets its goals in mitigating interferences by reducing Tx/channel and increasing channel reuse hop count under the peer-to-peer traffic. However, it is

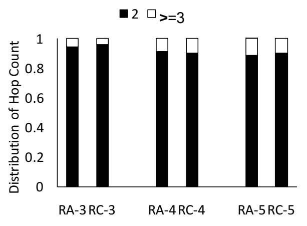


(a) Centralized traffic (#flows = 60, $P = 2^{0\sim3}$)

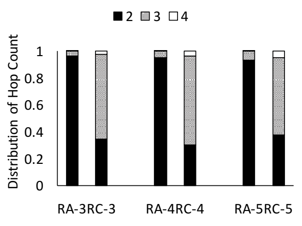


(b) Peer-to-peer traffic (#flows = 120, $P = 2^{0\sim2}$)

Fig. 4. Number of transmissions per channel under a varying number of channels (Indriya).



(a) Centralized traffic (#flows = 60, $P = 2^{0\sim3}$)



(b) Peer-to-peer traffic (#flows = 120, $P = 2^{0\sim2}$)

Fig. 5. Channel reuse hop count under a varying number of channels (Indriya).

less efficient with the centralized traffic because the centralized approach introduces more transmission conflicts, which limit the number of slots that can accommodate channel reuse.

C. Execution Time

We measure the execution time of the algorithms on a Macbook Pro laptop with 2.7 GHz Intel core i7. For NR, RA, and RC, we set the number of channels to 5 and the periods $P = [2^0, 2^2]$, and vary the traffic loads. The results are based on the peer-to-peer traffic. Figure 6 compares the execution time of NR, RA, and RC. NR obtains the smallest execution time of 0.2 to 0.4 ms, when the number of flows increases from 40 to 100. As the number of flows reaches 120, NR can no longer successfully generate schedules. The execution time of RA increases linearly from 1.2 ms to 1200 ms as the number of flows increases from 40 to 160, respectively, while RC incurs 115.1 ms to 290 ms, and sharply increases to 990 ms when the traffic loads grow from 40, 140, to 160.

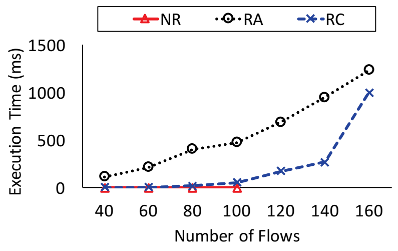


Fig. 6. Algorithm execution time (in milliseconds).

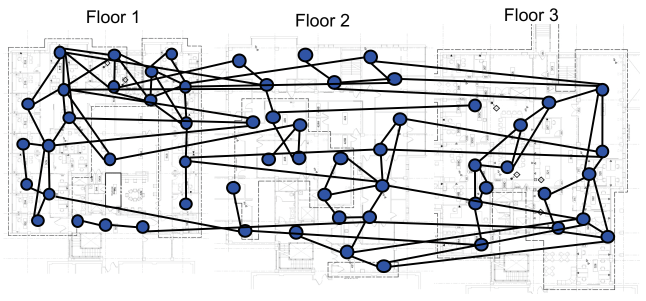


Fig. 7. WUSTL testbed topology when channels 11-14 are used.

RC requires less execution time than RA since RA attempts to adopt channel reuse for every transmission, while RC is more conservative. Although RC sparingly introduces channel reuse, it obtains longer execution time than NR because it is required to compute laxity for every transmission. A sudden increase in RC's execution time when the number of flows reaches 160 is the result of more workloads, and heavy computation for channel reuse.

RC's execution time is acceptable in the WirelessHART network because a schedule does not need to be reconfigured often. A schedule is recomputed only when the topology or the traffic demands change significantly. Accordingly, WirelessHART network maintenance is in frequent by design, e.g., it can take several minutes for a node to detect and report a link failure to the network manager.

D. Network Reliability

Despite enhancing real-time performance, channel reuse may degrade network reliability. We conduct experiments on the 60-node WUSTL testbed (see Figure 7) to obtain the reliability performance. The testbed runs TSCH MAC protocol with 10 ms time slot and source routing at the network layer. Source routing allows a sender to retransmit a packet if its first transmission fails. We generate five different flow sets consisting of 50 flows where 50% of flows release their packets every 2^{-1} s, and the rest release their packets every 2^0 s. We run experiments on 4 channels with observed good performance (channel 11 to 14) and set the transmission power to 0 dBm. For each flow set, we enable a network to execute the schedule for 100 times.

Figure 8 presents box plots comparing Packet Delivery Ratio (PDR) of NR, RA, and RC under 5 different flow sets. A PDR is a fraction of packets successfully delivered to its destination. RC achieves the median PDR at most 1% lower than NR's median PDR in flow sets 1, 4, and 5, whereas NR's median PDR surpasses that PDR of RA by a maximum of 2% under flow sets 1, 2, and 4. This shows that most of the flows of both RC and RA can achieve good reliability performance. In terms of worst-case reliability, RC clearly outperforms RA. RC achieves lower worst-case PDRs than NR by a margin of 3%, 0%, 8%, 3%, and 5% for flow sets 1 to 5, respectively, while RA decreases the worst-case PDRs by 29%, 18%, 22%, 31%, and 27% under flow sets

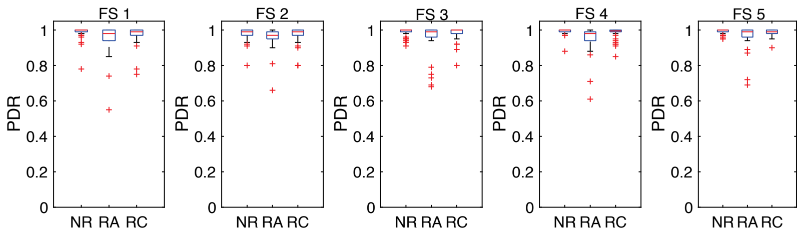


Fig. 8. Box plots of Packet Delivery Ratio (PDR) of 5 distinct flow sets where #flows=50, #ch=4, $P = 2^{-1\sim 0}$

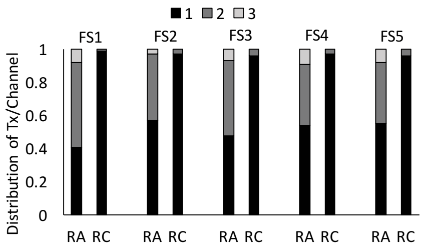
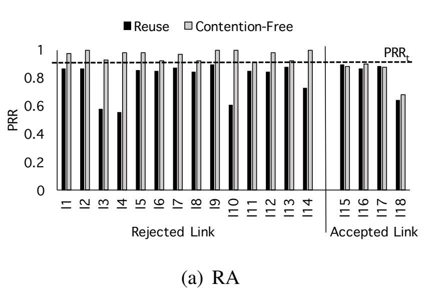
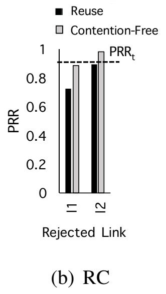


Fig. 9. Number of transmissions per channel under RA and RC of 5 different flow sets.





(a) RA (b) RC

Fig. 10. PRRs of rejected and accepted links failing to meet the reliability requirement when scheduled by RA and RC.

1 to 5, respectively, compared to NR. This is because RC adopts channel reuse more conservatively compared to RA as shown in Figure 9. It is important to note that *worst-case* reliability is important in industrial applications because severe data loss can degrade stability and control performance and may lead to system failure. Although RC may still suffer from interferences due to channel reuse, in exchange for a much higher schedulable ratio, advances in networked control have contributed control designs that can tolerate packet losses systems (e.g., [29] and [30]). Our approach is suitable for such resilient control systems.

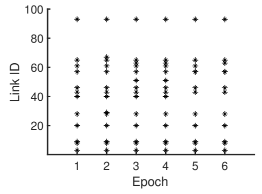
E. Links Affected by Channel Reuse

In this section, we evaluate our detection policy in terms of its applicability to identify links whose reliability is degraded by channel reuse under both RA and RC approaches. In this set of experiments, we generate 50 peer-to-peer flows, which release a packet every 1 s. By default, a WirelessHART node must deliver a health report to the network manager every 15 minutes (=1 epoch). In each epoch, we obtain the PRR distribution consisting of 18 samples. We also calculate the overall PRR of each link in each epoch. Note that we consider the PRR distribution of each link over all channels used.

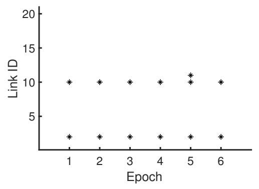
We run experiments under a clean environment on channel 11 to 14. We then generate external interference using WiFi in order to demonstrate that our detection policy can differentiate unreliable links caused by channel reuse from those caused by external interference. Because every node in the WUSTL testbed is connected to a Raspberry PI through a USB port for testbed management and instrumentation, we set up three pairs of Raspberry PIs, one pair on each floor, to generate interference on WiFi channel 1, which overlaps with 802.15.4 channels 11 to 14. For each pair of raspberry PIs, one is set up as a server (receiver), and another is a client sending 1 Mbps UDP traffic. We set the significance level α to 0.05 and PRR_t to 0.9. We run the experiments for 6 epochs. There are 95 and 20 links associated with channel reuse when the flow set is scheduled by RA and RC, respectively.

With RA under clean environment, there are 10 links whose reliability is below PRR_t under channel reuse. Because only links with $PRR \geq PRR_t$ are used for communication, this link quality degradation is therefore caused by channel reuse because there exists no external interference. After WiFi traffic is injected to cause external interference on the network, we observe 18 links with $PRR < PRR_t$. Out of the 18 links with low PRRs, our detection policy determines that only 14 cases (including all the 10 links with low PRRs in the clean environment) are attributed to channel reuse (reject), while transmission failures over the other 4 links are caused by external interference (accept). Our detection policy correctly detects all 10 links that suffered from channel reuse in the clean environment. Note that those same links are likely to remain vulnerable to channel reuse under external interference. Moreover, the additional 4 rejected links were also rejected by the *K-S* test in the clean environment. This indicates that channel reuse had a negative impact on these links. However, as their PRRs still met the reliability requirement in the clean environment, these links do not need to be rescheduled. When under external interference, however, the external interference degrades the PRRs of the links while the effect of channel reuse remains. Our policy hence includes these links among the links that require channel reuse to be avoided.

We run similar experiments using RC to generate a schedule. In the clean environment, all links satisfy the reliability requirement, while under external interference, our detection policy identifies two rejected links and no accepted link with $PRR < PRR_t$. Again, the two rejected links were also rejected by the *K-S* test in the clean environment demonstrating that they were affected by channel reuse. But because their PRRs were above PRR_t in the clean environment, we do not need to readjust the schedule for these links. Due to RC's more conservative channel reuse policy, fewer number of links suffer from link quality degradation due to channel reuse compared to RA.



(a) RA



(b) RC

Fig. 11. Rejected links failing to meet the reliability requirement in each epoch under external interference.

Figure 10 compares the PRRs of links that fail to meet the reliability requirements for both rejected and accepted cases when transmissions are scheduled by RA and RC. It can be observed that rejected links consistently achieve better performance on a contention-free channel but obtain low PRR when a channel is reused. While accepted links attain poor reliability performance for both cases. This result shows that our detection policy can effectively distinguish if link quality degradation is a result of channel reuse or external interference. Figure 11 presents rejected links in each epoch when transmissions are scheduled by RA and RC. The result demonstrates that our detection policy consistently obtains almost the same set of rejected links over the course of the experiments under external interference.

VIII. Conclusion

WSANs have become an enabling technology for many IIoT applications that impose strict demands for real-time and reliable performance. In contrast to traditional approaches designed to maximize channel reuse, we propose a *conservative* channel reuse to enhance the real-time performance of industrial WSANs while mitigating the impact of channel reuse on network reliability. We further design a classifier to detect reliability degradation caused by channel reuse. Experimental results from two testbeds demonstrate that conservative channel reuse significantly outperforms the traditional industrial WSANs without channel reuse in real-time performance, while achieving higher reliability than aggressive channel reuse.

Acknowledgment

This work was sponsored by NSF through grants 1320921 (NeTS) and 1646579 (CPS), and by the Fullgraf Foundation.

References

[1] WirelessHART, 2007. [Online]. Available: http://www.hartcomm2.org

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

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

[4] M. Nobre, I. Silva, and L. A. Guedes, "Routing and Scheduling Algorithms for WirelessHART Networks: a Survey," *Sensors*, vol. 15, no. 5, pp. 9703–9740, 2015.

[5] T. Watteyne, V. Handziski, X. Vilajosana, S. Duquennoy, O. Hahm, E. Baccelli, and A. Wolisz, "Industrial Wireless IP-Based Cyber-Physical Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1025–1038, 2016.

[6] R. Maheshwari, S. Jain, and S. R. Das, "A Measurement Study of Interference Modeling and Scheduling in Low-power Wireless Networks," in *Sensys*, 2008, pp. 141–154.

[7] S. Liu, G. Xing, H. Zhang, J. Wang, J. Huang, M. Sha, and L. Huang, "Passive Interference Measurement in Wireless Sensor Networks," in *ICNP*, 2010, pp. 52–61.

[8] L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan, "A General Model of Wireless Interference," in *MobiCom*, 2007, pp. 171–182.

[9] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher, "RID: Radio Interference Detection in Wireless Sensor Networks," in *INFOCOM*, vol. 2, 2005, pp. 891–901.

[10] H. Shokri-Ghadikolaei, C. Fischione, and E. Modiano, "On the Accuracy of Interference Models in Wireless Communications," in *ICC*, 2016, pp. 1–6.

[11] J. Grönkvist, "Interference-Based Scheduling in Spatial Reuse TDMA," Ph.D. dissertation, KTH, 2005.

[12] G. Brar, D. M. Blough, and P. Santi, "Computationally Efficient Scheduling with the Physical Interference Model for Throughput Improvement in Wireless Mesh Networks," in *MobiCom*, 2006, pp. 2–13.

[13] O. Chipara, C. Wu, C. Lu, and W. Griswold, "Interference-Aware Real-time Flow Scheduling for Wireless Sensor Networks," in *ECRTS*, 2011, pp. 67–77.

[14] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, "An Implicit Prioritized Access Protocol for Wireless Sensor Networks," in *RTSS*, 2002, pp. 39–48.

[15] S. 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, pp. 14–21.

[16] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A Global Time-synchronized Link Protocol for Sensor Networks," *Ad Hoc Networks*, vol. 6, no. 8, pp. 1201–1220, 2008.

[17] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic-Aware Scheduling Algorithm for Reliable Low-power Multi-hop IEEE 802.15.4e Networks," in *PIMRC*, 2012, pp. 327–332.

[18] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized Traffic-Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 455–470, 2015.

[19] IPv6 over the TSCH mode of IEEE 802.15.4e (6tisch). [Online]. Available: https://datatracker.ietf.org/wg/6tisch/about/

[20] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks through Autonomously Scheduled TSCH," in *Sensys*, 2015, pp. 337–350.

[21] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient Network Flooding and Time Synchronization with Glossy," in *IPSN*, April 2011, pp. 73–84.

[22] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, "Adaptive Real-Time Communication for Wireless Cyber-Physical Systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 2, pp. 8:1–8:29, Feb. 2017.

[23] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power Wireless Bus," in *Sensys*, ser. SenSys '12, 2012, pp. 1–14.

[24] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time Scheduling for WirelessHART Networks," in *RTSS*, 2010, pp. 150–159.

[25] "KolmogorovSmirnov Test (K-S test)," https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test.

[26] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed," in *TRIDENTCOM*, 2011, pp. 302–316.

[27] 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, no. 3, pp. 696–704, June 2017.

[28] D. Gunatilaka, M. Sha, and C. Lu, "Impacts of Channel Selection on Industrial Wireless Sensor-Actuator Networks," in *INFOCOM*, 2017.

[29] G. W. Irwin, J. Chen, A. Mckernan, and W. G. Scanlon, "Co-design of Predictive Controllers for Wireless Network Control," *IET Control Theory Applications*, vol. 4, no. 2, pp. 186–196, February 2010.

[30] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman Filtering with Intermittent Observations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, Sept 2004.