Parameter Self-Adaptation for Industrial Wireless Sensor-Actuator Networks

JUNYANG SHI and MO SHA*, State University of New York at Binghamton, USA

Wireless Sensor-Actuator Network (WSAN) technology is gaining rapid adoption by the industrial Internet of Things (IoT) applications in recent years. A WSAN typically connects sensors, actuators, and controllers in industrial facilities, such as steel mills, oil refineries, chemical plants, and infrastructures implementing complex monitoring and control processes. IEEE 802.15.4-based WSANs operate at low-power and can be manufactured inexpensively, which makes them ideal where battery lifetime and costs are important. Recent studies have shown that the selection of network parameters has a significant effect on network performance. However, the current practice of parameter selection is largely based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials, resulting in non-optimal decisions in many cases. In this work, we develop the *Parameter Selection and Adaptation FramEwork (P-SAFE)* that optimally selects the network parameters based on the application Quality of Service (QoS) demands and adapts the parameter configuration at runtime to consistently satisfy the dynamic requirements. We implement P-SAFE and evaluate it on three physical testbeds. Experimental results show that our solution can significantly better meet the application QoS demand compared to the state of the art.

CCS Concepts: • **Networks** → **Network performance modeling**; **Network management**; **Sensor networks**; *Network simulations*; *Wireless mesh networks*; • **Computing methodologies** → *Modeling methodologies.*

Additional Key Words and Phrases: Industrial Wireless Sensor-Actuator Networks, Parameter Selection

ACM Reference Format:
Junyang Shi and Mo Sha. 2020. Parameter Self-Adaptation for Industrial Wireless Sensor-Actuator Networks. *ACM Trans. Internet Technol.* 1, 1, Article 1 (January 2020), 25 pages. https://doi.org/10.1145/3388240

1 INTRODUCTION

Wireless Sensor-Actuator Network (WSAN) technology is gaining rapid adoption in process industries in recent years. According to a McKinsey's report, industrial IoT will contribute up to $47 trillion in added value globally by 2025 [41]. Emerson Process Management, one of the leading process automation suppliers, has deployed more than 51,234 WSANs globally and gathered the experience of 17.6 billion operating hours [19]. A WSAN typically connects sensors, actuators, and controllers in industrial facilities, such as steel mills, oil refineries, chemical plants, and infrastructures implementing complex monitoring and control processes. IEEE 802.15.4-based WSANs operate at low-power and can be manufactured inexpensively, which make them ideal where

*Corresponding author

Part of this article was published in Proceedings of the INFOCOM [51].
Authors' address: Junyang Shi; Mo Sha, State University of New York at Binghamton, 4400 Vestal Parkway East, Binghamton, NY, 13902, USA, {jshi28,msha}@binghamton.edu.

1533-5399/2020/1-ART1 $15.00
https://doi.org/10.1145/3388240

battery lifetime and costs are important. Battery-powered wireless modules easily and inexpensively retrofit existing sensors and actuators in industrial facilities without running cabling for communication and power. The stringent *reliability* and *real-time* requirements of industrial process applications distinguish industrial WSANs from traditional Wireless Sensor Networks (WSNs) designed for best-effort services. To meet the stringent requirements, industrial WSAN standards such as WirelessHART [61] made a set of specific design choices. For instance, WirelessHART adopts a centralized network architecture and employs the Time Slotted Channel Hopping (TSCH) technology [57]: Time is divided into time slots, which are long enough for packet transmission and its acknowledgment; All devices in a network are time synchronized and hop channels to exploit frequency diversity; A centralized network manager computes the TSCH schedule and determines how each device uses every time slot: transmit, receive, or sleep. Compared to the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA), the Time-Division Multiple Access (TDMA) based TSCH offers time-deterministic packet deliveries, which makes it attractive for real-time communication. With a decade of real-world deployments, industrial standards have demonstrated the feasibility for the TSCH-based WSANs to achieve reliable low-power wireless communication in industrial environments. Therefore, the TSCH technology was adopted by the leading industrial WSAN standards (WirelessHART [61] and ISA100 [29]) and the one being standardized by IETF (6TiSCH [27]) and amended into the IEEE 802.15.4 standard in 2012 [1].

Recent studies have shown that the selection of network parameters such as the Packet Reception Ratio (PRR) threshold for link selection, the number of channels used in the network, and the number of transmission attempts for each packet has a significant effect on the performance of industrial WSANs [24, 25, 51]. However, the current practice of parameter selection is largely based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials. For instance, WirelessHART has specified the use of all available channels after the human network operator manually blacklists noisy ones [61] and Emerson Process Management suggests using a constant value (i.e., 60%) as the PRR threshold to select links for routing [18]. Unfortunately, recent studies show that these specifications are error prone [25, 51]. For example, using more channels is not always desirable in industrial WSANs, since more channels mean more channel diversity but a large number of channels may reduce route diversity with negative effects on routing and scheduling.

Thanks to some recent work [15, 16, 22, 47, 58, 68], we are confident that we are just seeing the tip of the iceberg in terms of how much performance can be improved through enabling parameter adaptation. However, to fully realize the benefits offered by the parameter adaptation, two fundamental challenges must be overcome: (i) *Conceptual gap*: There exists a large conceptual gap between the high-level application Quality of Service (QoS) requirements and the low-level network parameters. It requires expert knowledge to find the parameters whose performance satisfies given requirements. Although most network parameters have been studied individually in the context of WSNs, there still exist phenomena that are unknown under an industrial WSAN setting. For example, a recent study shows that the performance of WSANs does not improve monotonically with more channels used because of the tradeoff between channel diversity and route diversity [25]. Owing to the lack of understanding of the underlying functional form of the relationships between high-level requirements and low-level parameters, the selection of suitable parameters becomes challenging. (ii) *Complex QoS demand*: Most industrial process applications today pose multiple (sometimes conflicting) QoS requirements on information exchange to their underlying networks. Learning the QoS demand of process applications that truly reflects their needs is particularly challenging, as multiple requirements must be met and tradeoffs have to be made among conflicting ones. The traditional solutions, which require users to order their QoS

requirements or rely on a coarse-grained weighted sum calculation, always result in non-optimal decisions in practice.

To address the above-stated challenges, we develop the *Parameter Selection and Adaptation FramEwork (P-SAFE)* that optimally selects the network parameters based on the application QoS demand and adapts the parameter configuration at runtime to consistently satisfy the dynamic requirements. Specifically, this paper makes the following contributions:

- We design a rigorous modeling method that relates the high-level application QoS requirements to the low-level network parameters;
- We formulate the parameter selection into a multi-objective optimization problem and employs the NSGA-II algorithm to identify the best tradeoff decisions;
- We develop a novel approach that learns the QoS preferences from the control application based on its specified ranges of desirability and translates the control metric into the network metric. The translated preference ranges can be used by the linear physical programming technique to identify the single (most attractive) best tradeoff decision;
- We implement P-SAFE and evaluate it on three physical testbeds under different application demands, different network settings, and environmental dynamics. Experimental results show our solution can significantly better meet the application demand compared to the state of the art.

The remainder of the paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the background of WSANs. Section 4 presents the design of our P-SAFE. Section 5 and 6 describe our QoS learning approach and control system analysis engine, respectively. Section 7 evaluates P-SAFE and Section 8 concludes the paper.

2 RELATED WORK

Recently, there has been significant research on real-time industrial WSANs spanning transmission scheduling, routing algorithms, and network protocols. For instance, Watteyne et al. presented an implementation of IP-based real-time communication over TSCH [59]. Duquennoy et al. proposed autonomous scheduling for TSCH in RPL networks [17] and Accettura et al. developed distributed traffic-aware scheduling in 6TiSCH networks [2]. Jocob et al. presented a distributed protocol that considers the complete transmission chain including peripheral buses, memory accesses, networking interfaces, and the wireless real-time protocol [30]. Saifullah et al. presented a schedulability analysis under graph routing in WirelessHART Networks [49] and Gunatilaka et al. proposed two channel selection approaches [24, 25]. Wu et al. and Shi et al. developed real-time routing protocols [52, 62, 63]. Lu et al. provided a comprehensive survey of recent advances in this increasingly important class of wireless networks [40]. Yet, a key missing piece in industrial WSANs is a self-adaptation component, which allows WSANs to optimally configure themselves based on specific QoS requirements and adapt the configurations at runtime to consistently satisfy the dynamic requirements in uncertain environments. This paper aims to accomplish this and advance the state of the art of real-time industrial WSANs through creating a new paradigm of parameter adaptation.

The characteristics of IEEE 802.15.4 wireless links have been studied extensively in the context of WSNs. There has been a vast array of research that empirically studied the link quality with different platforms, under varying experimental conditions, assumptions, and scenarios [5]. For instance, Zhao et al. [65] and Srinivasan et al. [55] investigated the packet delivery performance in dense WSNs in indoor and outdoor environments. Zhou et al. studied the impact of radio irregularity on the communication performance in WSNs and established the radio irregularity model (RIM) [66]. Liang et al. proposed a Medium Access Control (MAC) layer solution that enables

the IEEE 802.15.4 devices to coexist with WiFi devices by using multi-headers technique [38]. To improve the wireless simulation, Lee et al. took a step forward in simulating packet delivery by modeling different noise signatures [35]. There also has been extensive research investigating the benefit of multi-channel communication in WSNs and mesh networks. For example, Zhou et al. [67] and Kim et al. [31] developed multi-frequency MAC protocols for WSN applications. Control theory has been applied to multi-channel WSNs to optimize the throughput [33, 34]. Alicherry et al. optimized the throughput in wireless mesh networks by using joint channel assignment and routing [3]. Doddavenkatapp et al. proposed the intermediate quality link transformation protocol (ILTP) to exploit channel diversity in WSNs [14]. Kodialam et al. further characterized the capacity region in multi-radio multi-channel wireless mesh networks [32]. The interference-aware channel assignment algorithm has been designed for wireless mesh networks to address the interference problem [48]. Lin et al. developed a distributed scheduling algorithm for the channel assignment in wireless mesh networks [39]. The extensive studies produced valuable guidelines on selecting parameters but also caused a perception that those parameters can be selected manually during the deployment based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials. As a result, WirelessHART has specified the use of all available channels after the human network operator manually blacklists noisy ones [61] and Emerson Process Management has specified to use 60% as a threshold to select links [18]. Unfortunately, recent studies show that these specifications are error prone [25, 51]. Thanks to some recent works, we are confident that we are just seeing the tip of the iceberg in terms of how much performance can be improved through enabling runtime parameter adaptation. Leveraging the semantically-enriched models, Seeger et al. proposed a rule-based translation of application-level QoS constraints into the software-defined network (SDN) configurations [50]. Zimmerling et al. developed the pTunes framework that reduces the packet loss when facing network changes through enabling adaptation of radio on and off timings and demonstrated its performance through applying it to X-MAC [8] and LPP [46] protocols [68]. Peng et al. [47] and Wang et al. [58] developed methods to reduce energy consumption by adapting the sleep intervals in duty-cycled MACs. Dong et al. proposed to adjust the packet length towards the same goal [15, 16]. Fu et al. highlighted the challenges of adapting multiple parameters simultaneously because of their joint effect on performance [22]. Although these works have some fundamental limitations, such as only adapting deployment-independent parameters, requiring precise knowledge of their effect on performance, and optimizing towards a single requirement, they have shed light on the promising opportunity for constructing a self-adaptive network. However, there is hardly any precedent for a rigorous scientific method to model the effect of deployment-dependent parameters and generate a robust set of strategies to support network parameter decisions. This motivates our work to enable the parameter adaptation in WSANs.

Part of this article was published in Proceedings of the INFOCOM [51]. Compared to the conference version [51], this paper presents our new design of a control analysis engine that translates the control performance requirements into the network QoS requirements. The control analysis engine leverages accurate mathematical models and simulations of the real-world control system to provide meaningful ranges of desirability for the network system. We use the control of an interacting two-tank system consisting of two interacting liquid tanks as an example to demonstrate the desirability range translation process. Besides, this paper also presents our experimental studies on the impact of interference on the performance of P-SAFE and the overhead introduced by the schedule updates.

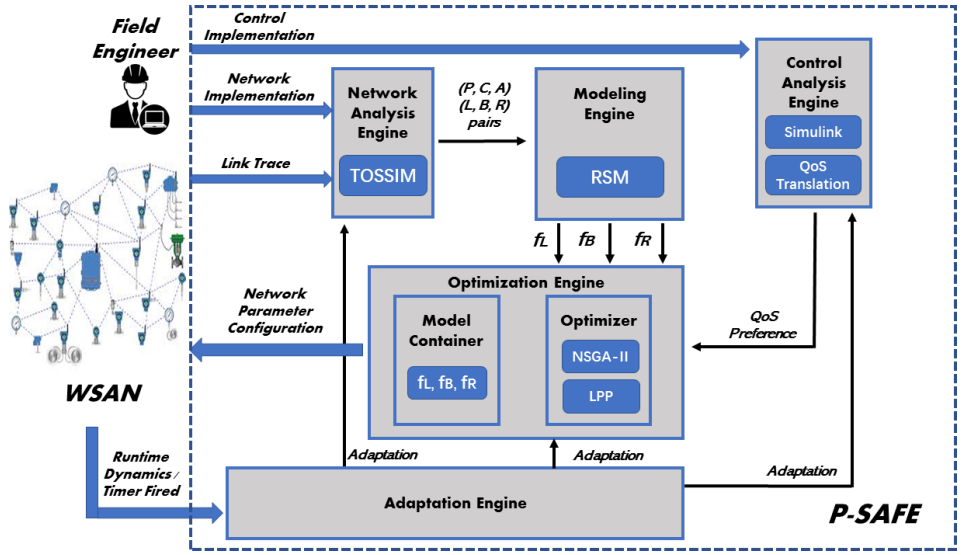


Fig. 1. System overview.

3 BACKGROUND OF WSANS

Recent studies show that the operating environments in industrial facilities are often harsh for low-power wireless communication due to the existence of strong ambient noise and metal objects [40]. To ensure reliable and real-time data deliveries in such environments, industrial WSAN standards, such as WirelessHART [61], made a set of unique design choices, such as employing a centralized network architecture, TSCH, and reliable graph routing. Each field device periodically sends a health report (e.g., in every 15 minutes), which includes information on link conditions and indications of any problems the device is having with a neighbor, to the network manager. The network manager uses the health reports collected from all field devices to determine the links which can be used for routing. To configure the network and update the transmission schedule, the network manager disseminates the network management messages by using the broadcast graph (a graph connecting the gateway downward to all field devices) [26]. TSCH technology combines time-slotted medium access, channel hopping, and multi-channel communication to provide time-deterministic packets deliveries. Inheriting from WirelessHART, TSCH has been implemented as a MAC protocol and was introduced as part of the IEEE 802.15.4e standard for industrial automation and control processes. Under TSCH, time is divided into fixed-length time slots which group into slotframes. Each time slot can be used to transmit a data packet and receive an acknowledgment between a pair of devices. To combat narrow band interference, TSCH requires each device to hop its operating channel in every time slot. Graph routing is designed to enhance network reliability by providing redundant routes between field devices and access points. A packet may be transmitted through the backup routes if the links on the primary path fail to deliver it. Only the links whose PRRs are higher than the PRR threshold on all channels used in the network can be selected for routing.

4 P-SAFE DESIGN

Figure 1 shows the design of our P-SAFE. After the engineers deploy a WSAN in the field, the **Network Analysis Engine** in P-SAFE guides them to implement the deployed network in it and also feed in the collected link (PRR) traces. The engine then simulates network performance under

Table 1. Data flows set on the BU Testbed used in Section 4.1 (Illustration and Example) and Section 7.1

Flow ID	Source	Destination	Period (ms)	Priority
1	147	146	800	1
2	144	143	800	2
3	105	104	800	3
4	149	102	800	4
5	136	135	1600	5
6	137	108	1600	6

each parameter configuration and forwards the results to the **Modeling Engine**. The Modeling Engine generates the empirical models that relate the parameter configurations to the performance of WSANs. The **Optimization Engine** stores the empirical models and selects the best-suited network parameters based on the QoS preferences learned from the **Control Analysis Engine**. The Control Analysis Engine guides the users to implement the target control application and convert control performance requirements to network performance requirements. The engine then forwards the network requirements preference to the Optimization Engine. The **Adaptation Engine** allows the Network Manager (a software module specified by WirelessHART to manage the network) and control application to update the network setting, link traces, and QoS preferences at runtime and adapts the parameters accordingly.

4.1 Network Analysis Engine

The design goal of our Network Analysis Engine is to analyze the network performance under each parameter configuration. It is impractical to perform test runs on a physical WSAN due to the significant overhead. Fortunately, the state-of-the-art wireless control simulators such as WCPS [60], Truetime [9], NCSWT [21], and Gisso [4] are capable of holistic studies of CPU scheduling, communication, and control algorithms [40]. Our Network Analysis Engine adopts WCPS (wireless cyber-physical simulator), which employs a federated architecture that uses TOSSIM [37] for simulating WSANs. TOSSIM has been widely used in the WSN community to simulate WSANs based on wireless link models that have been validated in diverse real-world environments [36]. WCPS also provides a WirelessHART implementation in TOSSIM.

After the field engineers deploy a physical WSAN, our Network Analysis Engine guides them to (i) implement the deployed network in TOSSIM (specifying the data sources and destinations, sampling rates, routing and scheduling algorithms), (ii) feed the link traces collected from the deployment into TOSSIM. The Network Analysis Engine then performs simulations under each parameter configuration. Three key network parameters, identified in the recent study [25], including (i) PRR threshold for link selection P, (ii) number of channels used in the network C, and (iii) number of transmission attempts scheduled for each packet A are considered simultaneously in the simulations. Assuming the pool of candidate parameters contains n_P for P, n_C for C, n_A for A, our Network Analysis Engine measures the network performance including (1) end-to-end latency L, (2) battery lifetime B, and (3) end-to-end reliability R, under all $n_P \times n_C \times n_A$ combinations among those three parameters (i.e., P, C, and A). The simulated performance together with its associated parameter configurations are forwarded to the Modeling Engine. We will next use an example to illustrate the process.

Illustration and Example:

In the example, we configure six data flows on the Binghamton University (BU) Testbed consisting of 50 TelosB motes [43] placed throughout several office areas including student offices, lounge,

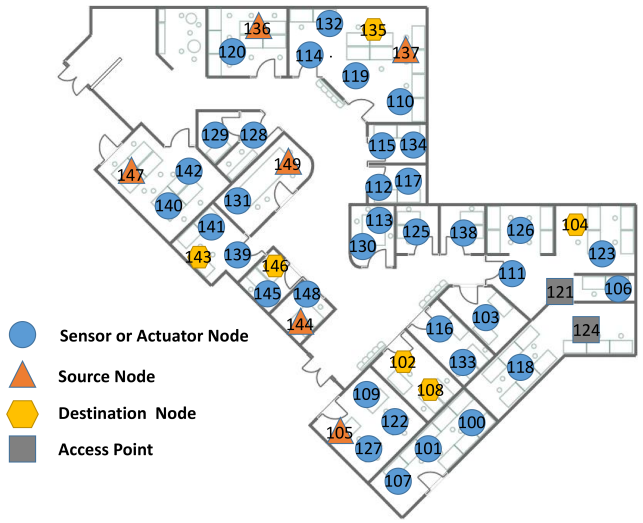
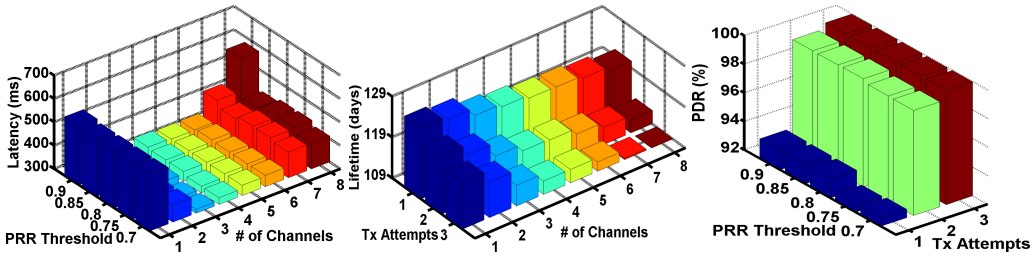


Fig. 2. Network setting on the BU Testbed used in Section 4.1 (Illustration and Example) and Section 7.1.



(a) Average latency with three attempts per packet.
(b) Average battery lifetime with a 85% PRR threshold.
(c) Average PDRs with 5 channels used.

Fig. 3. 3D-plot of network performance with different parameter configurations.

labs, and conference rooms [56]. The testbed consists of three tiers: wireless devices, switches, and a centralized server. At the bottom tier, wireless devices are placed throughout the physical environment to perform wireless experiments. Each wireless device is a Raspberry Pi 3 Model B integrating with a TelosB mote and a Power over Ethernet (PoE) splitter. The TelosB mote is connected to the Raspberry Pi through a USB cable. Messages can be exchanged between them over this interface in both directions. The wireless devices are powered by PoE and connected to the PoE switches in the middle tier through Ethernet cables. At the top tier exists a dedicated server which connects to the PoE switches through an Ethernet cable. This server is used to host, among other things, a database containing information about the different TelosB motes and Raspberry Pis they are connected to. This database minimally contains information about the connections that have been established between the TelosB motes and Raspberry Pis, as well as their current locations. The server is also used to provide a workable interface between the testbed and any end-users. Figure 2 shows the deployment of field devices and access points, and Table 1 lists the period and priority of each data flow. The packet delivery deadline is equal to the period. The graph routing and priority scheduling are employed in the simulations. We

consider $P \in \{0.7, 0.75, 0.8, 0.85, 0.9\}$, $C \in \{1, 2, 3, 4, 5, 6, 7, 8\}$, and $A \in \{1, 2, 3\}$. The performance is simulated under 120 different parameter configurations.

Since it is not feasible to show the data in a four dimension view, we fix a dimension and present three 3-D plots in Figure 3. Figure 3(a) shows the end-to-end latency under different PRR threshold P and number of channels used C combinations. The results confirm the observation reported in the early study that more number of channels used cannot always provide lower latency because of the tradeoff between route diversity and channel diversity [25]. We also observe that the effect of PRR threshold on latency gets stronger when more channels are used in the network. Figure 3(b) shows that the battery lifetime decreases when either the number of channels used C or the number of attempts per packet A increases. The significance of the effects from two parameters is different. Figure 3(c) shows the end-to-end reliability in terms of Packet Delivery Rate (PDR) that increases slightly with PRR threshold P and significantly with the number of attempts per packet A. While Figure 3 shows some interesting observations helping us understand the joint effect of those three parameters, it also highlights an important challenge posed by the interplay among them, which will be addressed by our Modeling Engine.

4.2 Modeling Engine

The design goal of our Modeling Engine is to generate the empirical models that relate the parameter configurations to the performance of WSANs. It is a significant challenge to empirically model the joint effect of three interplaying parameters without an understanding of the underlying functional form of the relationships being modeled. Modeling the relations theoretically is not efficient since the models can be deployment-dependent (e.g., depending on the particular network topology, setting, and protocols). Therefore, our Modeling Engine takes a black-box modeling approach and adopts the widely used Response Surface Methodology (RSM) [6] to construct the models. RSM is a black-box modeling technique and uses polynomial functions to approximate the model functions between the independent variables (inputs) and the response (outputs) without comprehending the underlying physical meaning between inputs and outputs, thus it provides a tractable and inexpensive approximation of the actual system behavior using polynomial functions.

Our Modeling Engine takes a tuple of performance metrics (L, B, R) and corresponding parameters (P, C, A) to construct three performance functions:

$$\begin{aligned} L &= f_L(P, C, A) + \varepsilon_1 \\ B &= f_B(P, C, A) + \varepsilon_2 \\ R &= f_R(P, C, A) + \varepsilon_3 \end{aligned} \tag{1}$$

where ε_i is a random experimental error assumed to have a zero mean. It is important to note that our Modeling Engine allows a P-SAFE user to replace the default RSM with another modeling technique. As an example, we replace RSM with a Kriging surrogate modeling approach [53] in the following example and show the models constructed by RSM and Kriging, respectively. Kriging is a type of spatial interpolation that uses complex mathematical formulas to estimate values at unknown points based on the values which are already sampled. The estimation of the value is denoted as Z_0 and the observed values are $\{Z_1, ..., Z_N\} = Z^T$, so the estimated value can be expressed as:

$$Z_0 = \sum_{i=1}^{N} w_i Z_i \tag{2}$$

where w_i denotes the influence weight. Kriging uses the minimum variance method to calculate the weights w_i.

Illustration and Example (continued):

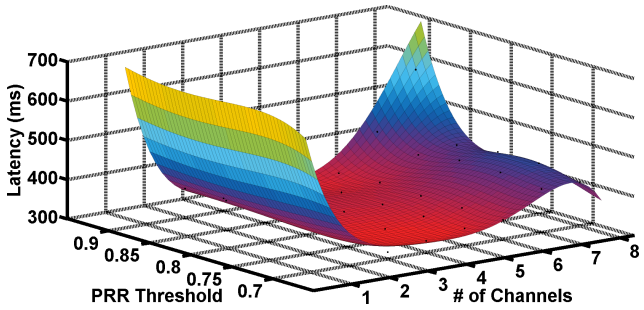


Fig. 4. The surface plot of latency when applying RSM. The number of transmission attempts per packet is set to 3.

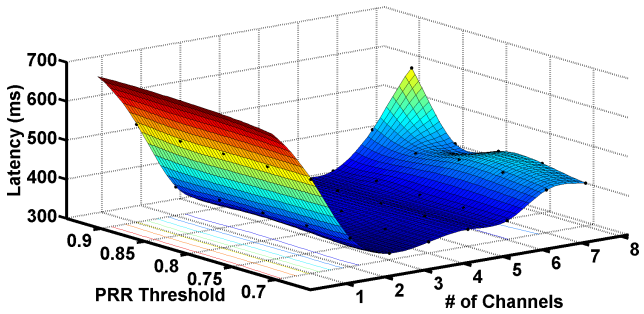


Fig. 5. The surface plot of latency when applying Kriging. The number of transmission attempts per packet is set to 3.

We use the implementations provided by Design Expert10 [20] and Matlab SUMO toolbox [23] for RSM and Kriging modeling, respectively. Figure 4 and Figure 5 show the surface plots of latency (Eq. 1) when applying RSM and Kriging on the data trace plotted in Figure 3(a). The generation of Kriging models consumes much more time compared to RSM but introduces no error between the resulting mathematical functions and samples. For example, we run the modeling process on a Dell Linux laptop with the 2.8GHz Intel Core E3-1505M with 40 samples plotted in Figure 3(a). The modeling time when applying RSM and Kriging is 160ms and 4.27s, respectively. The average modeling errors under RSM and Kriging are 3.19% and 0%.

With our open design, a P-SAFE user is free to choose any modeling technique based on the need of target application. Please note that the modeling overfitting may happen, which motivates us to design the Adaptation Engine (see Section 4.5) to overcome the modeling inaccuracy at runtime.

4.3 Optimization Engine

After obtaining the empirical models in Section 4.2, the next step is to generate a novel set of decision-making strategies to select the best-suited network parameters based on the given QoS requirements specified by the control application. Specifically, for each given (L, B, R), the parameters can be obtained by solving an optimization problem based on Eq. 1. The challenge is that most industrial process applications today pose multiple (sometimes even conflicting) QoS requirements on information exchange to their underlying networks. The traditional solutions, which require network users (e.g., a control engineer) to order their QoS requirements or rely on a coarse-grained weighted sum calculation, simplify the problem but result in non-optimal decisions in many cases. To address this problem, we develop a novel approach that learns the QoS demand of a given

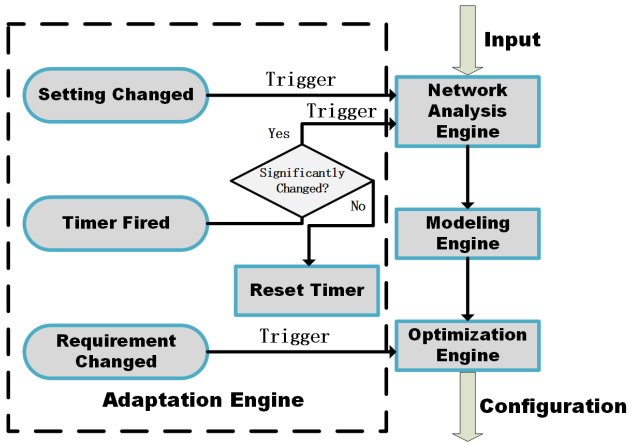


Fig. 6. Adaptation Engine's responses to different changes.

process application that truly reflects its needs and simultaneously applies them to the parameter selection process. The detailed design will be presented in Section 5. After the optimization engine selects the best-suited network parameters, the network manager disseminates them to all field devices by using network management messages.

4.4 Control Analysis Engine

The Control Analysis Engine is designed to analyze the control system performance under different network conditions and translates the control performance requirements into the network QoS requirements. The network QoS requirements learned from the control system are then provided to the Optimization Engine (See Section 4.3). It is important to note that it is a significant challenge to translate the control performance metrics to the network performance metrics without knowing the underlying control system mathematical models. To address the challenge, our Control Analysis Engine leverages WCPS (wireless cyber-physical simulator) [60] to analyze the control system and translate the control performance metrics to the network performance metrics. The detailed design will be presented in Section 6.

4.5 Adaptation Engine

The Adaptation Engine allows the Network Manager and control application to update the network setting, link traces, and QoS preferences at runtime and adapts the parameters accordingly. Since changing network parameter introduces significant communication and computation overhead, our engine employs a hybrid approach that combines event-driven and time-driven adaptations and responds differently when facing different changes. Figure 6 shows the actions of our Adaptation Engine in response to different kinds of changes. For example, the engine invokes the Network Analysis Engine and Modeling Engine to remodel the network and then reperform the optimization if the network setting (e.g., data sources and destinations) changes. It skips the modeling process and reruns the optimization if the application requirements change but the network setting stays the same. The modeling and optimization processes are invoked to examine the network by a timer if no event-driven adaptation is triggered during a long period. If the new optimized parameter configuration is not significantly better than the current one (i.e., smaller than a threshold), it is retained; else the network switches to a new configuration.

5 QOS LEARNING APPROACH

As discussed in Section 4.3, most industrial process applications today pose multiple (sometimes even conflicting) QoS requirements on information exchange to their underlying networks. Learning the QoS demand of process applications that truly reflects their needs is particularly challenging, as multiple requirements must be met and tradeoffs have to be made among conflicting ones. The traditional weighted sum approach merging multiple QoS requirements into a single objective function suffers from three significant limitations when applied in industrial WSANs:

- It is difficult to specify good weights that truly reflect the QoS preferences of a process application;
- A change in the QoS preferences of a process application does not readily translate into a change in specified weights;
- When a process application has non-linear QoS requirements, if the Pareto frontier [12] (i.e., the collection of non-dominated or best tradeoff solutions) is non-convex and/or disjointed, it can fail to obtain the best tradeoff solutions, and even higher-order weighted combinations (i.e., $\sum w_i f_i^n$, where n is an even number such as 2, 4, ...) could face difficulty in leading to the Pareto frontier [11].

The difficulty is pervasive in the context of conflicting QoS requirements, where blindly optimizing a weighted aggregate of the multiple QoS requirements provides limited to no information regarding the tradeoffs that exist among them. For example, is it beneficial to improve the reliability by A% in exchange of degradations of B% latency and C% energy consumption?

To avoid using the rules of thumb QoS orders or weights, we formulate our optimization problem into a multi-objective optimization problem and apply a widely used evolutionary algorithm to identify the best tradeoff solutions. We then develop an approach using the physical programming technique[1] to identify the most attractive tradeoff decision. We also provide the entire set of best tradeoff solutions to the P-SAFE users in case they want to see all available choices and tradeoffs.

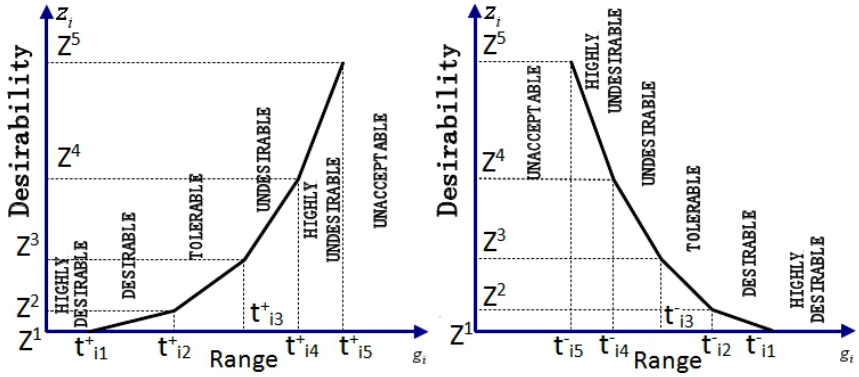
5.1 Problem Formulation and Optimization

The objective of selecting the best parameter configuration is to (i) minimize the end-to-end latency L, (ii) maximize the lifetime B, and (iii) maximize the network reliability R. Thus the problem can be formulated as

$$\begin{aligned} &min/max: f_L(P,C,A), f_B(P,C,A), f_R(P,C,A) \\ &subject\ to: P \in [P_{min}, P_{max}] \\ &C \in [C_{min}, C_{max}] \\ &A \in [A_{min}, A_{max}] \end{aligned} \tag{3}$$

where $[P_{min}, P_{max}]$, $[C_{min}, C_{max}]$, and $[A_{min}, A_{max}]$ denote the feasible ranges of the PRR threshold P, the number of channels used C, and the number of attempts for each packet A, and $f_L(P,C,A)$, $f_B(P,C,A)$, $f_R(P,C,A)$ represent the vector of objectives that should be minimized or maximized subject to a number of bounds. We adopt the NSGA-II algorithm [13], one of the most widely used multi-objective evolutionary algorithms, to solve the problem. Since Eq. 3 defines three different objectives, there does not exist a single best solution which simultaneously optimizes all objectives. NSGA-II gives a large number of best tradeoff solutions lying on or near the Pareto frontier, which can serve as the parameter selection candidates. Our implementation of NSGA-II

[1]The physical programming technique was developed in the area of multidisciplinary design optimization to address engineering design problems such as aircraft and automobile design. It provides a powerful methodical approach to obtain the most attractive best tradeoff decision from the set of best tradeoff solutions [44, 45]. The physical programming approach can be applied as a post-process if the set of Pareto solutions are obtained using a multi-objective optimization algorithm.



(a) Class-1S. (b) Class-2S.

Fig. 7. Different class functions ith objective.

has the complexity of $O(N^2)$, where N is the population size. To measure the time consumption, we run the NSGA-II algorithm on a Dell Linux laptop with the 2.8GHz Intel Core E3-1505M. It takes an average of 6.28s to solve our three objective functions with three constraints of P, C, and A. The next step is to obtain the single (most attractive) best tradeoff decision based on the needs of the target process application.

5.2 LPPA: A Linear Physical Programming based Approach

Instead of requesting the network users to specify the weights among different QoS requirements (with which they are less familiar), our Linear Physical Programming [42] based Approach (LPPA) allows the users to specify meaningful ranges of desirability on the network performance metrics (with which they are familiar). For the decreasing preference metrics (e.g., latency), we use g_i to denote the ith generic criterion value, so the range of desirability can be defined as: (i) *Highly Desirable Range* ($g_i \leq t^+_{i1}$): an acceptable range over which the improvement that results from further reduction of the criterion is desired but is of minimal additional value; (ii) *Desirable Range* ($t^+_{i1} \leq g_i \leq t^+_{i2}$): an acceptable range that is desirable; (iii) *Tolerable Range* ($t^+_{i2} \leq g_i \leq t^+_{i3}$): an acceptable range that is tolerable; (iv) *Undesirable Range* ($t^+_{i3} \leq g_i \leq t^+_{i4}$): an acceptable range that is undesirable; (v) *Highly Undesirable Range* ($t^+_{i4} \leq g_i \leq t^+_{i5}$): an acceptable range that is highly undesirable; and (vi) *Unacceptable Range* ($t^+_{i5} \leq g_i$): the range of values that is not acceptable (can be perceived as a hard constraint). Similar ranges of desirability, t^-_{ij}, can be defined for the increasing preference metrics (e.g., reliability). Therefore, we define two different class-functions as follows:

- Class-1S: Smaller-Is-Better, i.e., minimization.
- Class-2S: Larger-Is-Better, i.e., maximization.

It is important to note that unlike weights in the weighted sum method, the parameters t^+_{ij} and t^-_{ij} defined above are physically meaningful constants that are specified by control applications (See Section 6) in light of user-supplied preferences associated with the ith metric (e.g., latency, battery lifetime, or reliability.).

The ranges of all QoS metrics are then to be exploited by physical programming through an inter-criteria rule called "One Versus Others (OVO)," where a full improvement of g_i across a given range of preference is over a full reduction of all the other criteria across the next better range of preference. This is accomplished through a mapping of the preferences to a transformed class

function space. Figure 7 shows the functions for Class-1S and Class-2S. The L, B, and R values are mapped to the desirability values z_i (called z-value). A lower z_i is always more desirable. By using the class functions, our approach converts three different criteria (i.e., L, B, and R) on the horizontal axis to z-value on the vertical axis for comparison. Then the aggregated z is used as a metric for desirability. The mathematical relations are as below:

$$z^s \equiv z_i(t_{is}^+) \equiv z_i(t_{is}^-)\ \forall i; (2 \leq s \leq 5); z^1 \equiv 0 \tag{4}$$

where s denotes a generic junction and i denotes the criterion number, and z^s means the z-value in each junction point on the vertical axis as shown in Figure 7. Eq. 4 guarantees that different metrics are treated equally when they are in the same desirability region.

The increasing value of z_i for ith criterion between adjacent junction points can be calculated by:

$$\bar{z}^s \equiv z^s - z^{s-1}; (2 \leq s \leq 5); z^1 \equiv 0 \tag{5}$$

showing that different criteria increase uniformly across the same desirability region.

Following the OVO rule, we apply the relationship:

$$\bar{z}^s = \beta(n_c - 1)\bar{z}^{s-1}; (3 \leq s \leq 5); n_c > 1; \beta > 1 \tag{6}$$

where n_c denotes the number of criteria which equals to 3 based on our design. β is used as a convexity parameter. And $\bar{z}^2$ should be initialized manually with a small positive number. However, it cannot guarantee the convexity of the class function only based on Eq. 6. The following functions should be satisfied in order to meet with convexity property:

$$\bar{t}_{is}^+ = t_{is}^+ - t_{i(s-1)}^+; \bar{t}_{is}^- = t_{is}^- - t_{i(s-1)}^-; (2 \leq s \leq 5) \tag{7}$$

where $\bar{t}_{is}^+$ and $\bar{t}_{is}^-$ denote the sth range of the ith criterion on the horizontal axis. So the magnitude of the slopes of each line (w_{is}^+ and w_{is}^-) takes the following form:

$$w_{is}^+ = \bar{z}^s / \bar{t}_{is}^+; w_{is}^- = \bar{z}^s / \bar{t}_{is}^-; (2 \leq s \leq 5) \tag{8}$$

Based on Eq. 8, the difference between the slope of each line $\bar{w}_{is}^+$ and $\bar{w}_{is}^-$ is:

$$\bar{w}_{is}^+ = w_{is}^+ - w_{i(s-1)}^+; \bar{w}_{is}^- = w_{is}^- - w_{i(s-1)}^-; (2 \leq s \leq 5) \tag{9}$$

The convexity requirement can be achieved by the relationship:

$$\bar{w}_{min} = \min_{i,s}\{\bar{w}_{is}^+, \bar{w}_{is}^-\} > 0; (2 \leq s \leq 5) \tag{10}$$

indicating that the slope of lines should increase monotonically. An iteration of increasing β by a step of 1 is needed until it satisfies Eq. 10 to meet with convexity.

We use the deviation value (d_{is}^-,d_{is}^+) to calculate the aggregated z and the final decision making among the best tradeoff solutions is selected by calculating the below expression:

$$\begin{aligned} &\min_{d_{is}^-, d_{is}^+} \sum_{i=1}^{n_c} \sum_{s=2}^{5} (\bar{w}_{is}^- d_{is}^- + \bar{w}_{is}^+ d_{is}^+) \\ &subject\ to: \\ &g_i - d_{is}^+ \leq t_{i(s-1)}^+; d_{is}^+ \geq 0; g_i \leq t_{i5}^+\ (i = 1, 2, 3, s = 2, ..., 5) \\ &g_i + d_{is}^- \geq t_{i(s-1)}^-; d_{is}^- \geq 0; g_i \geq t_{i5}^-\ (i = 1, 2, 3, s = 2, ..., 5) \end{aligned} \tag{11}$$

All the best tradeoff solutions on the Pareto frontier computed by NSGA-II are fed into Expr. 11. The final best-suited network parameter configuration results in the minimum aggregated z (Expr. 11).

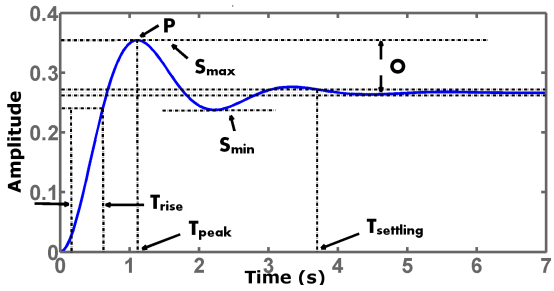


Fig. 8. Control performance metrics marked in a step response example.

6 CONTROL ANALYSIS

LPPA allows users to specify the desirability ranges of network performance metrics instead of using meaningless weights. However, the relations between the network performance metrics and control performance metrics are still unknown. After the engineers finish the deployment in the field, our Control Analysis Engine guides them to implement the target control application in it and then translates the desirability ranges of the control performance into the desirability ranges of the network performance by performing control simulation. Our Control Analysis Engine employs Simulink [54] to simulate the physical system dynamics in automation processes. Simulink is widely used by control engineers to design and study control systems and it provides accurate mathematical models and realistic simulation of various wireless control systems. Figure 8 illustrates seven example control metrics: overshoot O, settling time $T_{settling}$, rise time T_{rise}, peak time T_{peak}, peak P, settling max S_{max}, and settling min S_{min}. There are three main steps to translate the above-mentioned control requirements to network requirements: (i) identifying the control metrics of interest, (ii) generating the system transfer function between input and output, and (iii) performing simulation under different network conditions to obtain desirability ranges. The Control Analysis Engine bridges the gap between control performance requirements and network QoS requirements. The translated desirability ranges can be used for our LPPA to identify the appropriate network parameters. Following an open design, our Control Analysis Engine can translate different control performance metrics to various network performance metrics based on the optimization objectives.

Illustration and Example:

We use the control of an interacting two-tank system consisting of two interacting liquid tanks [10] as an example to demonstrate how our Control Analysis Engine translates the desirability ranges of a control metric (i.e., overshoot) to the desirability range of a network QoS metric (i.e., latency). Figure 8 shows the overshoot, O, in a step response, which can be defined as:

$$O = \frac{v_{out\_peak} - v_{out}(\infty)}{v_{out}(\infty)} \tag{12}$$

where v_{out_peak} denotes the highest peak of time response and $v_{out}(\infty)$ denotes the magnitude of its steady-state.

Figure 9 shows the diagram of an interacting two-tank control system. In the two-tank system, sensors are deployed to monitor the liquid level of Tank 2 (h_2), actuators are used to control the velocity of flow Q_{in} into Tank 1, and a controller is employed to generate control decisions. A WSAN is formed to forward the sensor readings to the controller and then send the control commands to the actuator. In Figure 9, A_1 (m^2) and A_2 (m^2) denote the base area of Tank 1 and Tank 2, respectively. The h_1(m) and h_2(m) present the height of liquid level in Tank 1 and Tank 2.

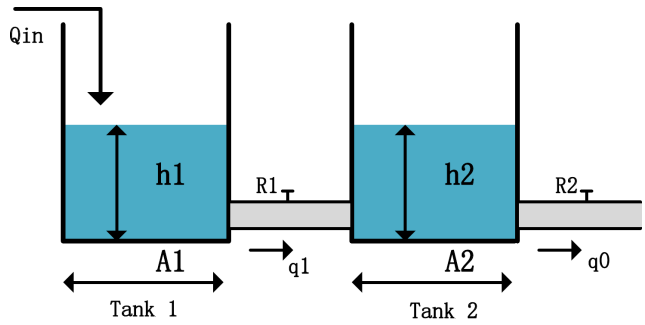


Fig. 9. Interacting two-tank system.

The velocity of liquid flow into Tank 1 is Q_{in} (m^3/min). The liquid flow velocity between Tank 1 and Tank 2 is q_1 (m^3/min). The velocity of flow leaving Tank 2 is q_0 (m^3/min). $R1$ and $R2$ are water resistance factors maintaining constant values.

The control input is Q_{in} and the controlled variable is the liquid level of Tank 2 h_2. Here is how to compute the system transfer function. For Tank 1, we have

$$A_1 \frac{dh_1}{dt} = Q_{in} - q_1 \tag{13}$$

Here we assume a linear resistance to a liquid flow:

$$q_1 = \frac{h_1 - h_2}{R_1} \tag{14}$$

After simplifying Eq. 13 with Eq. 14, we have

$$A_1 R_1 \frac{dh_1}{dt} = R_1 Q_{in} - h_1 + h_2 \tag{15}$$

We can get the time constant of Tank 1 T_1 as

$$T_1 = A_1 R_1 \tag{16}$$

After taking the Laplace transform on both sides on Eq. 15, we have

$$h_1(s) = \frac{R_1 Q_{in}(s)}{(T_1 s + 1)} + \frac{h_2(s)}{(T_1 s + 1)} \tag{17}$$

For Tank 2, we have

$$A_2 \frac{dh_2}{dt} = q_1 - q_0 \tag{18}$$

We again assume a linear flow resistance:

$$A_2 \frac{dh_2}{dt} = \frac{h_1 - h_2}{R_1} - \frac{h_2}{R_2} \tag{19}$$

Similar to T_1 of Tank 1, the time constant of Tank 2 T_2 is:

$$T_2 = A_2 R_2 \tag{20}$$

After taking the Laplace transform on both sides of Eq. 19, we have

$$(R_1 T_2 s + R_2 + R_1) h_2(s) = h_1(s) R_2 \tag{21}$$

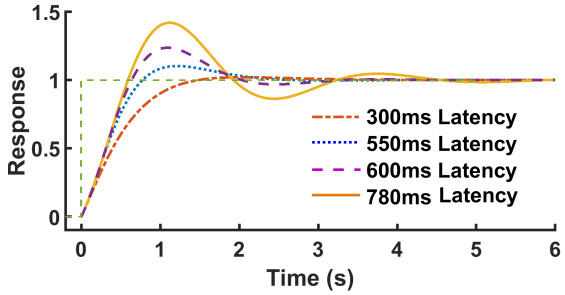


Fig. 10. Control performance with varying latency.

We then put the value of h_1 in Eq. 17 to Eq. 21. We finally get the transfer function between the input (Q_{in}) and output (h_2) as:

$$\frac{h_2(s)}{Q_{in}(s)} = \frac{R_2}{T_1T_2s^2 + s(T_1 + T_2 + A_1R_2) + 1} \tag{22}$$

where T_1 and T_2 denote the time constant of Tank 1 and Tank 2 computed by Eq. 16 and Eq. 20. We implement the above transfer function and a PID controller in WCPS. Figure 10 shows the control response to a step input under different network latency. With the control simulations in WCPS, the model that relates the overshoot and latency are constructed. We assume that the desirability ranges of overshoot are defined as: highly desirable $(0, 2\%]$; desirable $(2\%, 3\%]$; tolerable $(3\%, 4\%]$; undesirable $(4\%, 5\%]$; highly undesirable $(5\%, 6\%]$; and unacceptable $(6\%, \infty)$. The resulting desirability ranges of latency are: highly desirable $(0, 188]$ms; desirable $(188, 263]$ms; tolerable $(263, 325]$ms; undesirable $(325, 379]$ms; highly undesirable $(379, 430]$ms; and unacceptable $(430, \infty)$ms. Please note that we only show an example here and a P-SAFE user should select his/her own control metrics of interest and define the desirability ranges based on the actual application needs in practice.

7 EVALUATION

To validate the efficiency of P-SAFE in optimally configuring the network parameters and adapting them at runtime to consistently satisfy the application QoS demand, we perform a series of experiments. We first examine the capability of P-SAFE to effectively adapt the parameters to accommodate QoS demand changes. We then evaluate P-SAFE's performance under different network settings. Finally, we study the effect of interference on the performance of P-SAFE. We compare our P-SAFE against three baselines: (i) the method specified in *WirelessHART*, (ii) the *CR+CP* approach [25], and (iii) the *optimal* solution using a brute-force method[2] and repeat the experiments on three physical testbeds located in different cities: (1) the *BU* Testbed consisting of 50 TelosB motes deployed on a single floor of a building [56]; (2) the *CPSL* Testbed with 60 motes spanning three floors of a building [64]; and (3) the *Indriya* Testbed, an open access 105-node testbed deployed in a 3-floor building at National University of Singapore [28]. In all experiments, we empirically set β to 15 and $\bar{z}^2$ to 0.1 in P-SAFE to satisfy the convexity and OVO requirements in LPPA and assume that two Lithium Ion AA batteries with a total capacity of 22,100J are used to power each node for battery lifetime calculation.

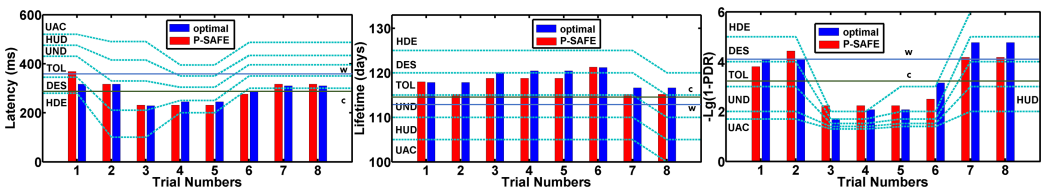
[2]The brute-force method cannot be used in practice because of its heavy computation overhead. We run it offline and use it only for the comparison purpose.

Table 2. Parameters selected by P-SAFE.

Set #	# of Channels	PRR Threshold	# of Attempts
1	2	89%	3
2	3	73%	3
3	3	71%	2
4	3	71%	2
5	3	71%	2
6	2	70%	2
7	3	72%	3
8	3	72%	3

Table 3. Parameters selected by optimal.

Set #	# of Channels	PRR Threshold	# of Attempts
1	3	76%	3
2	3	76%	3
3	3	72%	2
4	3	85%	2
5	3	85%	2
6	2	88%	2
7	3	71%	3
8	3	71%	3



(a) Latency. (b) Battery lifetime. (c) PDR. $-lg(1-PDR)$ is used for y-axis for clarity.

Fig. 11. Resulting performance under P-SAFE, CR+CP, WirelessHART and optimal.

7.1 Adaptation to QoS Demand Changes

To test P-SAFE's capability to consistently satisfy the application QoS demand, we perform a series of controlled experiments where the control application specifies different ranges of desirability on three QoS attributes: latency, battery lifetime, and PDR. We set up six data flows with periods ranging from 800ms to 1600ms and two access points (node 121 and 124) on the BU Testbed. Figure 2 shows the network setting and Table 1 lists the source, destination, data period, and priority of each data flow. We employ an interacting two-tank control system (see Section 5.2 and 6) running on top of the network. The control application uses a timer to issue eight different sets of real-world desirability ranges, provided by our industry partner, to P-SAFE one by one with a 1 hour time interval. Only one QoS attribute is changed at a time. Table 2 and Table 3 show the parameters

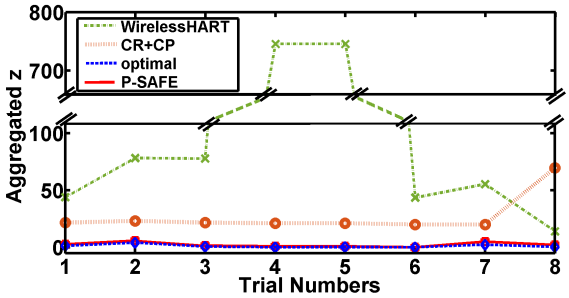


Fig. 12. Aggregated z under different methods.

selected by P-SAFE and optimal for each set of desirability ranges. The selections made by P-SAFE and optimal are alike resulting in similar performance, as shown in Figure 11. P-SAFE effectively selects the best tradeoffs among three QoS attributes and always keeps the latency, battery lifetime, and PDR in the tolerable range or better. The performance of WirelessHART and CR+CP is marked as two straight lines with "w" and "c" in Figure 11, respectively. WirelessHART and CR+CP do not consider the desirability ranges, thus they select the same parameters for all eight sets. Specifically, WirelessHART decides to use 8 channels, 60% as the PRR threshold, and 3 attempts per packet, while CR+CP selects 4 channels, 85% as the PRR threshold, and 75% threshold for the backup route, and 3 attempts per packet. Figure 12 shows the comparisons on the aggregated z-value z, the metric indicating how well the performance meets the application QoS demand, among four approaches. We use LPP to map the latency (L), battery lifetime (B), and reliability (R) to the desirability values z_i (z-value). A lower z-value means that the performance (L, B, R) is more desirable by the control system. We sum up the z-value to get aggregated z as an objective for optimization. P-SAFE significantly outperforms WirelessHART and CR+CP. The maximum aggregated z under P-SAFE is no more than 5.78, very close to what optimal provides (4.22). WirelessHART has the worst performance, with an average aggregated z of 225.55 and a worst-case value of 745.55. This is because WirelessHART uses a predetermined PRR threshold and operates on all available channels. CR+CR reduces the average aggregated z to 26.40 and the worst-case to 69.85, since it considers the effect of PRR threshold and number of channels used on network performance. However, CR+CP fails to make tradeoffs when facing conflicting QoS requirements resulting in substantially higher aggregated z values compared to what P-SAFE and optimal offer.

7.2 Performance under Different Network Settings

To explore the consistency of P-SAFE's performance, we run the experiments under different network settings on three testbeds. We create thirty different network settings by varying the sources, destinations, data periods, and priorities of data flows on the BU Testbed. Figure 13 plots the Cumulative Distribution Function (CDF) of aggregated z under WirelessHART, CR+CP, P-SAFE, and optimal, respectively. As Figure 13 showed, the performance of P-SAFE is very close to the one under optimal. The worst-case (maximum) values are 6.78 and 3.69 under P-SAFE and optimal, while the maximum values under WirelessHART and CR+CP are 89.32 and 62.41. P-SAFE achieves an average aggregated z of 2.76, representing 14.1X and 8.9X lower compared to WirelessHART and CR+CP, respectively.

We also repeat the experiments on the other two testbeds. On each testbed, we perform the experiments under 30 different network settings. Figure 14 shows the deployment of 60 TelosB motes on the CPSL Testbed spanning three floors of a building. Figure 15 plots the CDF of aggregated

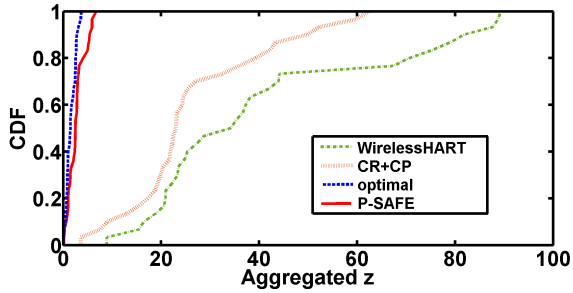


Fig. 13. Aggregated z with 30 different network settings on the BU Testbed.

Fig. 14. The deployment of the CPSL Testbed. Blue circles denote sensors and actuators and red squares represent two access points.

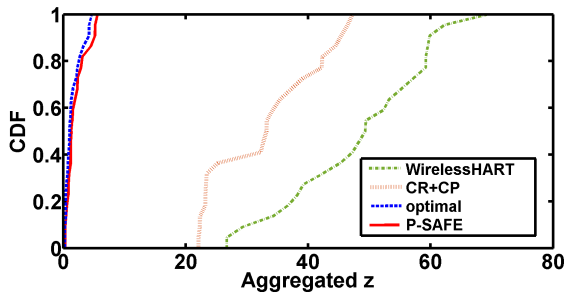


Fig. 15. Aggregated z with 30 different network settings on the CPSL Testbed.

z under different approaches. We observe similar performance. The maximum aggregated z values are 5.61 and 4.67 under P-SAFE and optimal, while the worst-case values under WirelessHART and CR+CP are 69.28 and 47.23, respectively. P-SAFE achieves an average aggregated z of 2.08, representing 22.5X and 15.0X lower compared to WirelessHART and CR+CP, respectively.

Figure 16 shows the deployment of Indriya Testbed consisting of 105 nodes spanning three floors. Figure 17 plots the CDF of aggregated z under different approaches. The maximum z under P-SAFE is 6.24, while the one under optimal is 5.62. WirelessHART and CR+CP have substantially higher aggregated z, with an average aggregated z of 41.13 under WirelessHART and 32.73 under CR+CP. The consistent results collected from various network settings under all three testbeds show that P-SAFE consistently better meets the application QoS demand, benefiting from our modeling method, multi-objective optimization, and QoS learning approach discussed in Section 4.

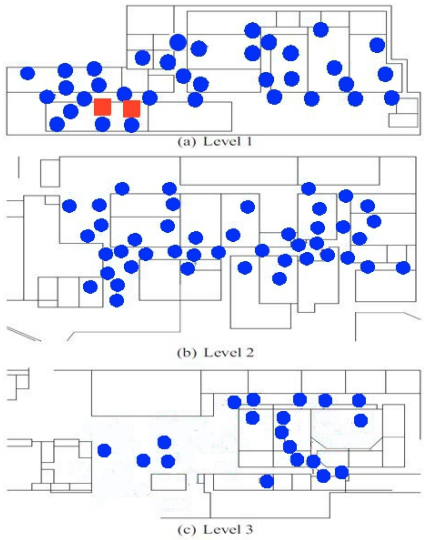


Fig. 16. The Indriya Testbed at National University of Singapore. Blue circles denote sensors and actuators and red squares represent two access points.

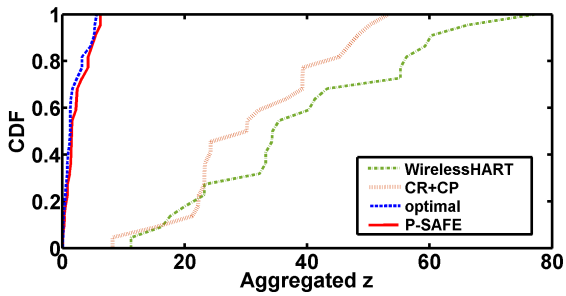


Fig. 17. Aggregated z with 30 different network settings on the Indriya Testbed.

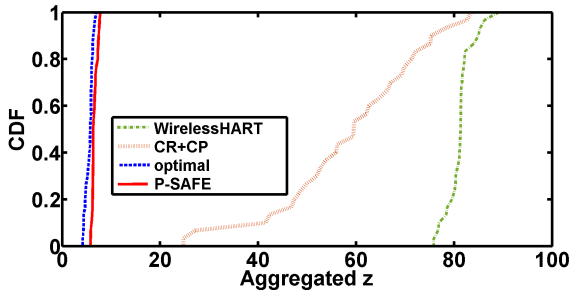


Fig. 18. Aggregated z under 30 different wireless conditions on the BU Testbed.

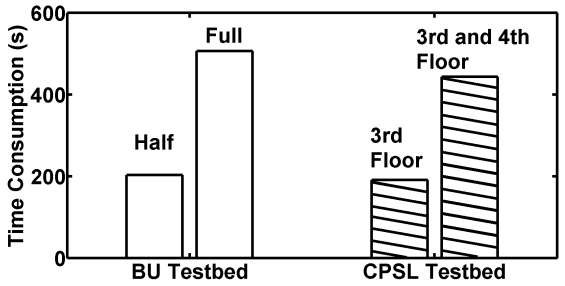


Fig. 19. Time consumed by the network manager to collect link traces, generate routes and transmission schedule, and disseminate the schedule to field devices.

Table 4. Comparison of P-SAFE, CR+CP and WirelessHART.

Method Name	Adjustable Parameters	QoS requirements	Trade-off Decision
P-SAFE	√	√	√
CR+CP	√	×	×
WirelessHART	×	×	×

7.3 Performance under Interference

We also study the impact of interference on the performance of P-SAFE. We create 30 different wireless operating conditions by using JamLab [7] to generate controlled interference and repeat the experiments under each condition. Figure 18 plots the CDF of aggregated z under 30 different wireless conditions on the BU Testbed. The maximum z under P-SAFE is 7.83, while the one under optimal is 6.99. The average z under P-SAFE is 11.28X and 7.95X lower than WirelessHART and CR+CP, respectively. We observe similar results on the other two testbeds and omit the results here due to the page limit. The consistent low z values provided by P-SAFE demonstrate P-SAFE's capacity of adapting the network parameters to consistently satisfy the application demand under various wireless ambient conditions.

7.4 Time Consumption on Schedule Updates

Finally, we measure the time consumed by the network manager to collect link traces, generate the routes and transmission schedule, and disseminate the schedule to field devices on two testbeds. We repeat the experiments by using half of the nodes on our testbed. We use a Dell Linux laptop with a 2.8 GHz Intel Core E3-1505M to run the network manager. As Figure 19 shows, it takes 506s and 443s for all nodes in the BU testbed and CPSL testbed to receive a new schedule, respectively. After disabling several nodes, it takes 203s and 191s for the nodes to obtain the new schedule on two testbeds. From the results, we can see that the time consumption of schedule updates largely depends on the number of nodes in the network. The results also indicate that the schedule updates should not happen frequently due to the considerable overhead.

8 CONCLUSIONS

Recent years have witnessed the rapid adoption of IEEE 802.15.4-based real-time WSANs in process industries, since they operate at low-power and can be manufactured inexpensively, which makes them ideal where battery lifetime and costs are important. Battery-powered wireless modules

easily and inexpensively retrofit existing sensors and actuators in industrial facilities without running cabling for communication and power. Recent studies have shown that the selection of network parameters such as the PRR threshold for link selection, the number of channels used in the network, and the number of transmission attempts for each packet has a significant effect on the network performance. However, the current practice of parameter selection is based on experience and rules of thumb involving a coarse-grained analysis of expected network load and dynamics or measurements during a few field trials, resulting in non-optimal decisions in many cases. This paper presents the P-SAFE, a framework that optimally configures the network parameters based on the application QoS demand and adapts the configuration at runtime to consistently satisfy the dynamic requirements. Table 4 summarizes the major differences between our approach and two existing solutions. CR+CP has considered the effect of PRR threshold and the number of channels used in the network to optimize the network performance, but it fails to make trade-off decisions to fully consider the QoS requirements by the control system or user preferences. WirelessHART only employs a predetermined PRR threshold and operates on all available channels which lead to the worst performance under different QoS requirements, different network settings, and environmental dynamics. Leveraging the empirical network models, NSGA-II, and LPPA, P-SAFE always provides the best-suited network parameters based on the application requirements. P-SAFE has been implemented and evaluated on three physical testbeds. Experimental results show our P-SAFE can significantly better meet the application performance demand compared to the state of the art.

ACKNOWLEDGMENT

This work was supported by the NSF through grant CRII-1657275 (NeTS).

REFERENCES

[1] 802.15.4e. 2012. IEEE-802.15.4e WPAN Task Group. Retrieved September 28, 2018 from http://www.ieee802.org/15/pub/TG4e.html

[2] Nicola Accettura, Elvis Vogli, Maria Rita Palattella, Luigi Alfredo Grieco, Gennaro Boggia, and Mischa Dohler. 2015. Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation. In *IEEE Internet of Things*, Vol. 2. IEEE, 501 Hoes Ln, Piscataway, NJ 08855, 17.

[3] Mansoor Alicherry, Randeep Bhatia, and Li Li. 2005. Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom '05)*. ACM, New York, NY, USA, 58–72. https://doi.org/10.1145/1080829.1080836

[4] Mikael Johansson B. Aminian, Jose Araujo and Karl H. Johansson. 2013. GISOO: A Virtual Testbed for Wireless Cyber-Physical Systems. In *Annual Conference of the IEEE Industrial Electronics Society (IECON)*. IEEE, Vienna, Austria, 6.

[5] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. 2012. Radio Link Quality Estimation in Wireless Sensor Networks: A Survey. In *ACM Transactions on Sensor Networks*. ACM, New York, NY, USA, Article 34, 33 pages. https://doi.org/10.1145/2240116.2240123

[6] Marcos Almeida Bezerra, Ricardo Erthal Santelli, Eliane Padua Oliveira, Leonardo Silveira Villar, and Luciane Amélia Escaleira. 2008. Response Surface Methodology (RSM) as a Tool for Optimization in Analytical Chemistry. *Talanta* 76, 5 (2008), 965 – 977.

[7] Carlo Alberto Boano, Thiemo Voigt, Claro Noda, Kay Römer, and Marco Zúñiga. 2011. JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation. In *IPSN*. IEEE, Chicago, IL, USA, 12.

[8] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. 2006. X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*. ACM, New York, NY, USA, 307–320. https://doi.org/10.1145/1182807.1182838

[9] Anton Cervin. 2018. TrueTime: Simulation of Networked and Embedded Control Systems. http://www.control.lth.se/truetime/

[10] Miral Changela and Ankit Kumar. 2015. Designing a Controller for Two Tank Interacting System. *International Journal of Science and Research* 4, 5 (2015), 589–593.

[11] Indraneel Das and John Dennis. 1997. A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. In *Structural Optimization*, Vol. 14. Springer, New York,

NY, USA, 7.
[12] Kalyanmoy Deb. 2014. Multi-objective Optimization. In *Search Methodologies*, Edmund K. Burke and Graham Kendall (Eds.). Springer, New York, NY, USA, Chapter 15, 403–449.
[13] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwa, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. In *IEEE Transactions on Evolutionary Computation*, Vol. 6. IEEE, 501 Hoes Ln, Piscataway, NJ 08855, 16.
[14] Manjunath Doddavenkatapp, Mun Choon Chan, and B. Leong. 2011. Improving Link Quality by Exploiting Channel Diversity in Wireless Sensor Networks. In *IEEE Real-Time Systems Symposium (RTSS)*. IEEE, Vienna, Austria, 11.
[15] Wei Dong, Chun Chen, Xue Liu, Yuan He, Yunhao Liu, Jiajun Bu, and Xianghua Xu. 2014. Dynamic Packet Length Control in Wireless Sensor Networks. In *IEEE Transactions on Wireless Communications*, Vol. 13. IEEE, 501 Hoes Ln, Piscataway, NJ 08855, 10.
[16] Wei Dong, Jie Yu, and Pingxin Zhang. 2015. Exploiting Error Estimating Codes for Packet Length Adaptation in Low-Power Wireless Networks. In *IEEE Transactions on Mobile Computing*, Vol. 14. IEEE, 501 Hoes Ln, Piscataway, NJ 08855, 14.
[17] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. 2015. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*. ACM, New York, NY, USA, 337–350. https://doi.org/10.1145/2809695.2809714
[18] Emerson. 2016. System Engineering Guidelines IEC 62591 WirelessHART by Emerson Process Management. http://www2.emersonprocess.com/siteadmincenter/PM%20Central%20Web%20Documents/EMR%5fWirelessHART%5fSysEngGuide.pdf
[19] Emerson. 2019. Emerson Wireless-technology. https://www.emerson.com/en-us/expertise/automation/industrial-internet-things/pervasive-sensing-solutions/wireless-technology
[20] Design Expert. 2018. Design-Expert 10. https://www.statease.com/soft-ftp
[21] Emeka Eyisi, Jia Bai, Derek Riley, Jiannian Weng, Yan Wei, Yuan Xue, Xenofon D. Koutsoukos, and Janos Sztipanovits. 2012. NCSWT: An Integrated Modeling and Simulation Tool for Networked Control Systems. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*. Elsevier, New York, NY, USA, 22.
[22] Songwei Fu, Yan Zhang, Yuming Jiang, Chengchen Hu, Chia-Yen Shih, and Pedro Jose Marron. 2015. Experimental Study for Multi-layer Parameter Configuration of WSN Links. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Columbus, OH, USA, 10.
[23] Dirk Gorissen, Ivo Couckuyt, Piet Demeester, Tom Dhaene, and Karel Crombecq. 2010. A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design. *J. Mach. Learn. Res.* 11 (Aug. 2010), 2051–2055.
[24] Dolvara Gunatilaka and Chenyang Lu. 2018. Conservative Channel Reuse in Real-Time Industrial Wireless Sensor-Actuator Networks. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Vienna, Austria, 10.
[25] Dolvara Gunatilaka, Mo Sha, and Chenyang Lu. 2017. Impacts of Channel Selection on Industrial Wireless Sensor-Actuator Networks. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Atlanta, GA, USA, 9.
[26] Song Han, Xiuming Zhu, Aloysius K. Mok, Deji Chen, and Mark Nixon. 2011. Reliable and Real-Time Communication in Industrial Wireless Mesh Networks. In *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, Chicago, IL, USA, 3–12. https://doi.org/10.1109/RTAS.2011.9
[27] IETF. 2013. 6TiSCH: IPv6 over the TSCH mode of IEEE 802.15.4e. Retrieved September 28, 2018 from https://datatracker.ietf.org/wg/6tisch/documents/
[28] Indriyatestbed. 2011. INDRIYA: A Wireless Sensor Network Testbed. https://indriya.comp.nus.edu.sg/motelab/html/index.php
[29] ISA100. 2009. ISA100. Retrieved September 28, 2018 from http://www.isa100wci.org/
[30] Romain Jacob, Marco Zimmerling, Pengcheng Huang, Jan Beutel, and Lothar Thiele. 2016. End-to-End Real-Time Guarantees in Wireless Cyber-Physical Systems. In *IEEE Real-Time Systems Symposium (RTSS)*. IEEE, Porto, Portugal, 12.
[31] Youngmin Kim, Hyojeong Shin, and Hojung Cha. 2008. Y-MAC: An Energy-Efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE, St. Louis, MO, USA, 11.
[32] Murali Kodialam and Thyaga Nandagopal. 2005. Characterizing the Capacity Region in Multi-radio Multi-channel Wireless Mesh Networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom '05)*. ACM, New York, NY, USA, 73–87. https://doi.org/10.1145/1080829.1080837
[33] Hieu Khac Le, Dan Henriksson, and Tarek Abdelzaher. 2007. A Control Theory Approach to Throughput Optimization in Multi-channel Collection Sensor Networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)*. ACM, New York, NY, USA, 31–40. https://doi.org/10.1145/1236360.1236365

[34] Hieu Khac Le, Dan Henriksson, and Tarek Abdelzaher. 2008. A Practical Multi-Channel Media Access Control Protocol for Wireless Sensor Networks. In *ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE, St. Louis, MO, USA, 12.

[35] HyungJune Lee, Alberto Cerpa, and Philip Levis. 2007. Improving Wireless Simulation Through Noise Modeling. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)*. ACM, New York, NY, USA, 21–30. https://doi.org/10.1145/1236360.1236364

[36] HyungJune Lee, Alberto Cerpa, and Philip Levis. 2007. Improving Wireless Simulation Through Noise Modeling. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)*. ACM, New York, NY, USA, 21–30. https://doi.org/10.1145/1236360.1236364

[37] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*. ACM, New York, NY, USA, 126–137. https://doi.org/10.1145/958491.958506

[38] Chieh-Jan Mike Liang, Nissanka Bodhi Priyantha, Jie Liu, and Andreas Terzis. 2010. Surviving Wi-fi Interference in Low Power ZigBee Networks. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*. ACM, New York, NY, USA, 309–322. https://doi.org/10.1145/1869983.1870014

[39] Xiaojun Lin and Shahzada Rasool. 2007. A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-hoc Wireless Networks. In *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, Barcelona, Spain, 9.

[40] Chenyang Lu, Abusayeed Saifullah, Bo Li, Mo Sha, Humberto Gonzalez, Dolvara Gunatilaka, Chengjie Wu, Lanshun Nie, and Yixin Chen. 2016. 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. IEEE, 501 Hoes Ln, Piscataway, NJ 08855, 12.

[41] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, and Alex Marrs. 2013. Disruptive technologies: Advances that will transform life, business, and the global economy. http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/disruptive-technologies

[42] C.D. McAllister, T.W. Simpson, K. Hacker, K. Lewis, and A. Messac. 2005. Integrating Linear Physical Programming within Collaborative Optimization for Multiobjective Multidisciplinary Design Optimization. *Structural and Multidisciplinary Optimization* 29, 3 (2005), 178–189.

[43] MEMSIC. 2004. TelosB Datasheet provided by MEMSIC Inc. http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

[44] Achille Messac. 1996. Physical Programming: Effective Optimization for Computational Design. In *AIAA Journal*, Vol. 34. AIAA, Salt Lake City, UT, USA, 11.

[45] Achille Messac. 2006. Multiobjective Decision-Making Using Physical Programming. In *Decision Making in Engineering Designs*, Kemper E. Lewis, Wei Chen, and Linda C. Schmidt (Eds.). ASME, New York, NY, USA, Chapter 15, 155–172.

[46] Razvan Musaloiu-E., Chieh-Jan Mike Liang, and Andreas Terzis. 2008. Koala: Ultra-low Power Data Retrieval in Wireless Sensor Networks. In *International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE, St. Louis, MO, USA, 12.

[47] Yang Peng, Zi Li, Daji Qiao, and Wensheng Zhang. 2013. I2C: A Holistic Approach to Prolong the Sensor Network Lifetime. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Turin, Italy, 9.

[48] Bhaskaran Raman. 2006. Channel Allocation in 802.11-Based Mesh Networks. In *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, Barcelona, Spain, 10.

[49] Abusayeed Saifullah, Dolvara Gunatilaka, Paras Tiwari, Mo Sha, Chenyang Lu, Bo Li, Chengjie Wu, and Yixin Chen. 2015. Schedulability Analysis under Graph Routing in WirelessHART Networks. In *IEEE Real-Time Systems Symposium (RTSS)*. IEEE, San Antonio, TX, USA, 10.

[50] Jan Seeger, Arne Bröring, Marc-Oliver Pahl, and Ermin Sakic. 2019. Rule-Based Translation of Application-Level QoS Constraints into SDN Configurations for the IoT. *2019 European Conference on Networks and Communications (EuCNC)* 5 (2019), 432–437.

[51] Junyang Shi and Mo Sha. 2019. Parameter Self-Configuration and Self-Adaptation in Industrial Wireless Sensor-Actuator Networks. In *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, Paris, France, 658–666.

[52] Junyang Shi, Mo Sha, and Zhicheng Yang. 2018. DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Vienna, Austria, 11.

[53] Timothy Simpson, Timothy Mauery, John Korte, and Farrokh Mistree. 2001. Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization. In *AIAA Journal*, Vol. 39. AIAA, Salt Lake City, UT, USA, 9.

[54] Simulink. 2019. Simulink. https://www.mathworks.com/products/simulink.html

[55] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. 2006. Understanding the Causes of Packet Delivery Success and Failure in Dense Wireless Sensor Networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*. ACM, New York, NY, USA, 419–420. https://doi.org/10.1145/1182807.1182885

[56] BU Testbed. 2017. BU testbed at Binghamton University. http://www.cs.binghamton.edu/%7Emsha/testbed

[57] TSCH. 2015. IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH). Retrieved April 10, 2019 from https://tools.ietf.org/html/rfc7554

[58] Jiliang Wang, Zhichao Cao, Xufei Mao, and Yunhao Liu. 2014. Sleep in the Dins: Insomnia Therapy for Duty-cycled Sensor Networks. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, Toronto, ON, Canada, 9.

[59] Thomas Watteyne, Vlado Handziski, Xavier Vilajosana, Simon Duquennoy, Oliver Hahm, Emmanuel Baccelli, and Adam Wolisz. 2016. Industrial Wireless IP-Based Cyber-Physical Systems. In *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems*, Vol. 104. IEEE, 501 Hoes Ln, Piscataway, NJ 08855, 14.

[60] Wireless Cyber-Physical Simulator (WCPS). 2013. WCPS Simulator. http://wsn.cse.wustl.edu/index.php/WCPS:_Wireless_Cyber-Physical_Simulator

[61] WirelessHART. 2004. WirelessHART. https://fieldcommgroup.org/technologies/hart

[62] Chengjie Wu, Dolvara Gunatilaka, Abusayeed Saifullah, Mo Sha, Paras Babu Tiwari, Chenyang Lu, and Yixin Chen. 2016. Maximizing Network Lifetime of WirelessHART Networks under Graph Routing. In *IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, Berlin, Germany, 11.

[63] Chengjie Wu, Dolvara Gunatilaka, Mo Sha, and Chenyang Lu. 2018. Real-Time Wireless Routing for Industrial Internet of Things. In *IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, Orlando, FL, USA, 6.

[64] Wustltestbed. 2015. CPSL Testbed at Washington University in St. Louis. http://cps.cse.wustl.edu/index.php/Testbed

[65] Jerry Zhao and Ramesh Govindan. 2003. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*. ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/958491.958493

[66] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. 2004. Impact of Radio Irregularity on Wireless Sensor Networks. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services (MobiSys '04)*. ACM, New York, NY, USA, 125–138. https://doi.org/10.1145/990064.990081

[67] Gang Zhou, Chengdu Huang, Ting Yan, Tian He, and John A. Stankovic. 2006. MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks. In *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, Barcelona, Spain, 13.

[68] Marco Zimmerling, Federico Ferrari, Luca Mottolay, Thiemo Voigty, and Lothar Thiele. 2012. pTunes: Runtime Parameter Adaptation for Low-power MAC Protocols. In *Proceedings of the 11th International Conference on Information Processing in Sensor Networks (IPSN '12)*. ACM, New York, NY, USA, 173–184. https://doi.org/10.1145/2185677.2185730