This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2020.2980909, IEEE Journal on Selected Areas in Communications

Reinforcement Learning-Based Control and Networking Co-design for Industrial Internet of Things

Hansong Xu*, Xing Liu*, Wei Yu*, David Griffith†, and Nada Golmie†
*Towson University, USA
Emails: {hxu2, xliu10}@students.towson.edu, wyu@towson.edu
†National Institute of Standards and Technology (NIST), USA
Emails:{david.griffith, nada.golmie}@nist.gov

***Abstract*—Industrial Internet-of-Things (IIoT), also known as Industry 4.0, is the integration of Internet of Things (IoT) technology into the industrial manufacturing system so that the connectivity, efficiency, and intelligence of factories and plants can be improved. From a cyber physical system (CPS) perspective, multiple systems (e.g., control, networking and computing systems) are synthesized into IIoT systems interactively to achieve the operator's design goals. The interactions among different systems is a non-negligible factor that affects the IIoT design and requirements, such as automation, especially under dynamic industrial operations. In this paper, we leverage reinforcement learning techniques to automatically configure the control and networking systems under a dynamic industrial environment. We design three new policies based on the characteristics of industrial systems so that the reinforcement learning can converge rapidly. We implement and integrate the reinforcement learning-based co-design approach on a realistic wireless cyber-physical simulator to conduct extensive experiments. Our experimental results demonstrate that our approach can effectively and quickly reconfigure the control and networking systems automatically in a dynamic industrial environment.**

***Index Terms*—Industry 4.0, Internet of Things, Reinforcement Learning, Control and Networking Co-design.**

I. Introduction

INDUSTRY 4.0 envisions the integration of Internet of Things (IoT) technology into industrial settings to achieve the high connectivity, reliability, efficiency, and intelligence in manufacturing facilities and plants. Two key components of an Industrial IoT (IIoT) system (i.e., cyber system and physical system) form a closed-loop architecture. The physical system represents the manufacturing and automation components, which carry out industrial production and process tasks. The cyber system consists of control, networking, and computing components for assisting operation, interconnection, and intelligence of IIoT systems [1].

In IIoT systems, control and networking systems interactively affect the performance of the physical system. For example, the control system usually increases the sampling rate to stabilize the controllability of the system when physical disturbances occur. Nonetheless, the high-volume network traffic caused by the increased sampling rate could degrade the network performance, which in turn negatively affects the performance of the control system. Moreover, from the communication perspective, the network system usually uses a more conservative modulation type, i.e., a signal constellation with fewer bits per symbol, to improve the packet delivery rate in a poor communication channel. Nonetheless, given the limited network resources, a conservative modulation type increases the amount of data to be transmitted. This could burden the network and in turn degrade the packet delivery rate and further affect the control performance.

As IIoT systems are highly coupled systems that synthesize control and networking systems, it is not feasible to conduct separate design (i.e., configuring the control system or the networking system independently). Thus, we propose a control and networking co-design approach using machine learning techniques such as reinforcement learning. Generally speaking, reinforcement learning techniques can be classified into model-based and model-free reinforcement learning [2]–[4]. Due to the complex interdependence between control and networking systems, it is impractical to derive or estimate a complete and accurate model for the integrated control and networking systems [5]–[7]. In contrast, the model-free reinforcement learning technique is capable of configuring the control and networking systems simultaneously while accounting for the interactions of both systems. The benefit of model-free reinforcement learning is that it does not require a mathematical model of the environment, so that the configurations under different circumstances can be carried out automatically [8].

The challenges of leveraging reinforcement learning to configure the control and networking systems automatically are two-fold. First, in the IIoT system, physical disturbances in control systems are difficult to predict, as well as the channel noise level in the networking system in a dynamic industrial environment. Second, the control system is designed to stabilize the system quickly, which imposes further constraints on the maximum convergence time of any reinforcement learning algorithm. Thus, we shall design a reinforcement learning approach to learn from the highly dynamic environment and make decisions quickly. Our design goal is to leverage the model-free reinforcement learning technique to enable self-configuration under the dynamic environment of an IIoT system with minimal convergence time at runtime.

To achieve our design goal, in this paper we leverage the

open source wireless cyber-physical simulator (WCPS) [9] to set up an IIoT system, which consists of a physical plant, a control center, and a network module. We use a fluid temperature control system (i.e., classic continuous stirred tank reactor (CSTR)), as the physical plant to demonstrate our idea. In an IIoT system, sensors collect fluid temperature data and send it to the control center through wireless communication channels. The controller computes actuation commands and sends them to actuators so that valves can be adjusted to achieve the control goals (i.e., maintaining a targeted temperature). We use the WCPS testbed to observe the performance of different control network parameters (i.e., sampling rate and modulation type) on the IIoT system under physical disturbances and multi-level communication noise. We observe that the high sampling rate of the control system fails to stabilize the CSTR under heavy communication noise. Instead, a lower sampling rate performs better under these conditions. This conclusion is confirmed through our experimental results, provided in Section VI-C.

Based on these observations, we implement and integrate a reinforcement learning module into the system to achieve a reinforcement learning-enabled IIoT system based on the open source WCPS, as shown in Fig. 1 [9][1]. The plant (CSTR) generates data by sensors and transmits the data to the Extended Kalman Filter (EKF) state observer through wireless networks. The EKF estimates the system state based on the sensed data received. Then, the model predictive control (MPC)-based controller computes control commands based on the system estimation and transmits the commands to actuators through wireless networks to control the plant setpoints. The commands are buffered before being sent to the actuators. The reinforcement learning-enabled control module takes the estimated system states as input and reconfigures control system and networking system simultaneously.

Particularly, for the reinforcement learning-based scheme, we consider the stability of the control system as a set of states, which can be measured by the maximum absolute error (MAE). In reinforcement learning, the scheme selects actions from action sets for control and networking systems based on a learning policy under the dynamic environment. After an action is taken, the system state will change to a new state and generate the reward for this action. At the same time, the scheme will update the learning policy and select new actions for current states. The goal of reinforcement learning is to compute an optimal policy that assigns optimal actions at given states.

To summarize, our contributions are as follows: (i) We use the WCPS testbed to capture the interactions between control and networking systems, as well as observe the control performance under varying networking environments. We investigate and evaluate the results of varying parameters (i.e., sampling rate and modulation type) under dynamic environments in a realistic CPS platform. (ii) We propose a new reinforcement learning-based scheme to self-configure the control and networking systems in a dynamic environment. In our co-design approach, the reinforcement learning-based scheme can tune the sampling rate for the control system and the modulation type of the networking system simultaneously so that near-optimal control performance can be achieved automatically. (iii) We propose several policies based on the characteristics of the control system to improve the performance of our reinforcement learning scheme. We also implement the reinforcement learning-enabled IIoT system, which automatically assigns actions to both the control and networking systems under a dynamic environment such that the performance of the IIoT system can be improved, as demonstrated through extensive experimental results derived from our real-world testbed.

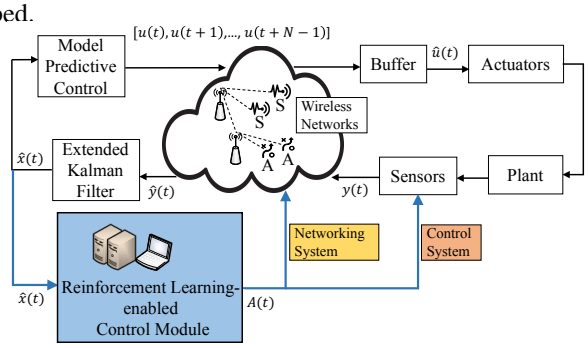

Fig. 1. Reinforcement Learning-enabled IIoT System Architecture

The remainder of this paper is organized as follows: We review research efforts on IIoT, wireless control systems, and machine learning for IIoT in Section II. In Section III, we provide a preliminary review of control and networking systems and reinforcement learning. In Section IV, we introduce our approach in detail. In Section V, we present our algorithms and the analysis of our approach. In Section VI, we present the experimental methodology, system implementation, and results. In Section VII, we discuss several future directions. We conclude the paper in Section VIII.

II. Related Work

As envisioned by Industry 4.0, industrial manufacturing systems are facing new opportunities and challenges to achieve highly flexible, efficient, and intelligent manufacturing production via advances in the IoT. A number of research efforts have been conducted on the applicability, efficiency, extensibility, security, and privacy of leveraging IoT in the context of industrial manufacturing systems, also called IIoT [1], [10]–[12]. For example, Xu *et al.* [10] systematically investigated the applicability of IoT technologies in the context of industry, including the key enabling technologies (e.g., radio-frequency identification (RFID)), architectures, and IIoT applications. Likewise, Xu *et al.* [1] surveyed IIoT from a CPS perspective, which consists of physical plants and machinery on the physical side and control, networking, and computing systems on the cyber side.

In addition, there have been a number of research efforts on wireless control systems, as one key component in an IIoT system [5], [13]–[15]. For example, Li *et al.* [13] implemented

[1]Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Symbols	Descriptions
A	Action set
a	An action selected from A (i.e., a combination of a sampling rate and a modulation type)
X	System state set
x	Current system state
x'	System state after an action
π	Reinforcement learning policy
$P(x, a, x')$	State transition probability function
$R(x, a, x')$	Reward function
$Q(x, a)$	Q-factor (denotes the value of action a at state x)
$r(t)$	Discount factor
MAE	Maximum absolute error
ΔMAE	Change of the system state
$ItLe_{MAX}$	Maximum iterations in learning phase
α^k	Learning rate
η	A scaling constant close to 1 (i.e., 0.99)
p^k	Exploration rate
Th_{STAB}	Stability threshold for system state (i.e., a MAE value)
Th_{Lexp}	Threshold for a low exploration
Th_{Hexp}	Threshold for a high exploration
Ω^k	Average reward received with the optimal policy
$ItFr_{MAX}$	Maximum iterations in frozen phase
REW_{TOT}	Total reward received in system runtime
TIM_{TOT}	Length of system runtime

TABLE I. Reinforcement Learning Parameters

an open source WCPS testbed to observe the interactions of control and networking systems. Ma *et al.* [14] proposed a cyber-physical management framework with a holistic controller to generate actuation signals sent to the physical plant and the reconfiguration and coordination of wireless sensor and actuator network (WSAN) at system runtime.

Adopting machine learning technology to address various challenges in IIoT (spectrum resource allocation, energy efficiency, data analytics, and prediction, among others) has attracted significant research attention [16]–[20]. For example, Oyewobi *et al.* [16] adopted a reinforcement learning technique to assess scarce spectrum resources dynamically in an IIoT environment. Also, research efforts have leveraged reinforcement learning techniques to conduct resource allocation in IIoT with consideration for high reliability and low latency requirements, such as Q-learning for channel allocation [17], Q-learning for computation offloading [18], and multi-armed bandit algorithms for channel hopping [20]. To overcome the high cost of Q-learning with a large state space, deep neural networks have been adopted to reduce the complexity and improve the convergence via approximating values for state-action pairs [17], [18].

To the best of our knowledge, no existing research efforts have leveraged state-of-the-art reinforcement learning techniques to address the control and networking co-design problem in highly coupled IIoT systems.

III. Preliminary

In this section, we briefly introduce the background and interactions of control and networking systems and provide an overview of reinforcement learning.

Control and Networking Systems: Industrial factories and plants typically employ a closed-loop system architecture, which consists of sensing and actuation phases to control objects to achieve certain goals. In the sensing phase, the industrial sensors collect measurement data (e.g., the fluid temperature in the fluid temperature control system), and send their measurements to the controller. In the actuation phase, the controller computes the actuation commands and sends them to actuators (e.g., valves), as shown in Fig. 1.

Control Systems: We use a linear time-invariant (LTI) system to model industrial plants to demonstrate the feasibility of our approach. Given an arbitrary input, the LTI system will produce an output (the observation) by convolving the input signal with the impulse response of the LTI system. Based on the observation, the controller computes a control or feedback signal (i.e., control commands) to stabilize the LTI system around a setpoint. We leverage the EKF as the state observer to estimate the system states. Specifically, the state observer first predicts the current system states based on its knowledge of previous states. Then, the EKF updates its prediction by comparing newly received sensing data (i.e., actual system states) to its predicted state values. Note that, when the sensing data is not received (perhaps due to packet loss), the observer outputs only the predicted state values [13].

We use the model predictive control (MPC) based controller to solve the optimal control problem on a given set of actuation values u (as a vector) in a finite horizon time. The MPC controller computes a sequence of control commands, where the length is determined by the finite horizon. These commands drive the system towards the setpoint (i.e., control goal) based on the current system state. Then, the MPC controller applies the first control action in the sequence u, to the LTI system, and reads the new system state. With the update of the system state, the MPC controller recomputes a new sequence of control commands for the next time interval. We also leverage a buffer for the MPC controller to store the subsequent control commands in the sequence, up to the buffer length, N. When no new control command sequence is generated due to loss of observations (sensing data), the stored control commands will be applied to the system, one at a time, until new a control command sequence is generated.

Networking Systems: Networking systems transmit sensing measurements and actuation commands in packets transmitted through the closed-loop system from sensors to controller and from the controller to actuators. IIoT networking systems must satisfy the real-time requirement and high-reliability requirement before widespread deployment [1]. Networking systems for IIoT can be wireless sensor actuator networks (WSANs), which interconnect components (sensors, controllers and actuators) and satisfy real-time requirements in industrial automation [5]. Note that the real-time requirement means that every packet needs to be delivered before its latency requirement deadline expires.

The WSAN in WCPS uses WirelessHART[2] as an open standard designed to satisfy the requirements of industrial applications [15]. Particularly, WirelessHART specifies multi-channel time division multiple access (TDMA) in the MAC layer and IEEE 802.15.4 in the PHY layer. A centralized

[2]WirelessHART is based on the Highway Addressable Remote Transducer Protocol (HART) protocol.

real-time scheduler assigns channel resources to sender and receiver pairs per time slot of 10 ms. Note that the time slot can be shared and dedicated for transmission. The shared time slot allows multiple senders to compete for the time slot while the dedicated time slot is reserved by a single sender. One time slot is sufficient to accommodate both data transmission and acknowledgment. In the PHY layer, WirelessHART utilizes the 2.4 GHz band with up to 16 channels. Also, channel-hopping technology is leveraged to avoid channel interference. Furthermore, the network layer provides source routing and graph routing schemes. The source routing creates a signal path from the sender to the destination, while the graph routing creates an additional backup path to improve the resilience of transmitting critical data.

Interactions Between Control and Networking Systems: In IIoT systems, the interactions of the control and networking systems affect the overall system performance. On one hand, the MPC controller can increase the sampling rate to improve the frequency of system state observations, which leads to finer control. Furthermore, the MPC controller with a low sampling rate can directly degrade the control performance, regardless of the communication performance [5]. On the other hand, due to the fact that the network resources in WSANs are limited, a higher sampling rate could increase the amount of data that needs to be delivered to the MPC controller during the sensing phase, increasing resource contention and real-time performance in the networking system. Thus, a sampling rate that is too high would, in turn, degrade control performance too. Thus, the dynamics of interactions between control and networking systems call for the co-design approach, which considers both control and networking systems simultaneously. The detail of our approach will be presented in Section IV.

Reinforcement Learning: The intelligent operation and automation of manufacturing factories and plants is a key objective of IIoT, which consists of autonomous sensing, learning, operations, control, and decision-making, among others, without human intervention in the IIoT system life cycle [1], [21], [22]. Specifically, in IIoT systems, decision-making (e.g., setting the sampling rate for the control system and choosing the modulation type for the networking system) needs to occur automatically according to the dynamics and interactions of the control and networking systems. In this way, IIoT systems are capable of adapting to the dynamic industrial environment quickly, efficiently, and automatically.

Generally speaking, the relationship of control commands to IIoT system states can be modeled as a Markov decision process (MDP). In our study, we use the observation signals as the system states to describe the LTI system. The system states are quantified by how close the current measurements are to the setpoints, measured by MAE. We use a set of actions (e.g., changing the sampling rate) to control the system in an optimal manner. Due to the network dynamics, an action (e.g., higher sampling rate) does not always lead to better states (closer to the setpoint).

The reinforcement learning model receives the observation signals from the LTI system as the input and selects an action (i.e., sampling rate and modulation type) as the output. After each action per time step, the LTI system will transit to a new state. We can observe the state transition by collecting the system observation signals. Accordingly, reinforcement learning will receive the immediate reward for each action taken based on the state transition, such as a positive reward for a good transition (closer to the setpoints) and a negative reward for a bad transition (farther from the setpoints). During runtime, our reinforcement learning model will choose actions that increase the overall rewards based on the sum of the rewards over time. By interacting with the dynamic industrial environment, reinforcement learning can achieve near-optimal results automatically by systematic trial and error.

Algorithms	Existing Research	Pros	Cons
Q-Learning	Li *et al.* [23], Wang *et al.* [24]	Q-Learning is an off-policy algorithm; Q-learning can find an optimal policy for any MDPs to maximize the total reward	High computing complexity when state or action spaces are large; Unable to deal with unseen states
SARSA	Arvind *et al.* [25], Zou *et al.* [26]	SARSA is an on-policy algorithm	Slower convergence than Q-learning
DQN	Zeng *et al.* [27], Qiu *et al.* [28], Liu *et al.* [29]	Good for high dimensional state or action space; Generalization to unseen states End-to-end training	Unstable performance

TABLE II. A Comparison of Model-free Reinforcement Learning Algorithms

We discuss and compare the pros and cons of key model-free reinforcement learning algorithms in Table II, including Q-learning, State-Action-Reward-State-Action (SARSA), and Deep Q Networks (DQN). The Q-learning algorithm uses a Q-table to store the state, action, and Q-value, and updates the Q-table through a trial and error process with the environment [23], [24]. Via systematic trial and error, Q-learning learns an optimal policy to maximize the total reward. The main issue of Q-learning is the high computing cost when the state or action space is large. DQN can reduce the cost by using a neural network to approximate the Q-value function in place of the Q-table. The DQN is a function approximation method that can effectively reduce the number of states and generalize to unseen states [27]–[29]. Similar to Q-learning, SARSA uses a Q-table to store data and updates the Q-table through trial and error. The main difference of SARSA is the updating process, where SARSA updates the Q-value based on the current policy (i.e., on-policy) instead of the greedy policy (i.e., off-policy) [25], [26]. Overall, for the sake of simplicity, we select Q-learning instead of other model-free reinforcement learning algorithms.

We summarize the key components of reinforcement learning-enabled IIoT systems and their relationships as follows: (i) the CSTR module models the physical phenomena in a classic continuous stirred tank reactor, (ii) the WSAN transmits timely sensing and actuation data, (iii) the EKF conducts state estimation with the received sensing data, (iv) the MPC controller computes a new control command sequence based on the state estimation results, (v) the reinforcement learning technique (in this case Q-Learning) conducts automatic reconfiguration on both the control system and the networking system. We present the detailed design of our reinforcement learning scheme in the next section.

IV. Our Approach

In this section, we present our approach in detail. We first introduce the design rationale and problem formalization. We then detail our Q-learning scheme and three learning policies.

A. Design Rationale

It is complicated and costly to obtain closed-loop formulae to model the control and networking co-design and derive mathematical solutions [5]–[7]. This is because there are a number of dynamic factors from both the control system (e.g., control policy, sensor accuracy, and disturbance) and the networking system (e.g., network configuration, channel conditions, and noise) as well. When integrating one factor from the networking system with the control system, a number of other factors in the control system will be impacted directly or indirectly. For example, when we integrate the channel noise into the control system, the control command transmission will be impacted. In this way, the current control model, which did not consider the added impacts, has become infeasible and needs to be modified. In addition, a number of control systems are modeled as time-invariant systems to which control laws are applied [30], [31]. When the networking system parameters are introduced (e.g., delay and packet loss) to the control system, the control system model could be no longer time invariant. In short, the co-design approach is very complex to achieve from a strictly analytical perspective.

Thus, in this paper we propose to leverage machine learning techniques to automate IIoT systems, as they have been demonstrated as feasible for the transformation of complex datasets into accurate knowledge as output [32]–[34]. Machine learning techniques have proven effective and been utilized in the context of IIoT in several ways [35]–[37]. Particularly, model-free reinforcement learning interacting with unknown environments through systematic trial and error has proven effective for decision-making tasks [8], [16]. Nonetheless, the standard model-free reinforcement learning usually needs a sufficient number of samples and learning processes to achieve a near-optimal policy [4]. Thus, it is challenging to apply model-free reinforcement learning in control systems, as the convergence time is generally intolerable in real control systems. To address this issue, we introduce several new policies in Section IV-D to improve the convergence time of reinforcement learning schemes that are tailored for control systems.

B. Problem Formalization

We model the co-design problem (i.e., selecting control and networking parameters with regards to the system dynamics) as an MDP problem. At each time step, reinforcement learning selects actions to move the system between different states. The reinforcement learning scheme will assign each state transition a positive or negative reward depending on whether the system state is changed towards or away from the setpoints. The objective is to find policy π that maximizes the cumulative reward to ensure the quick stabilization of control systems. The MDP is defined with a tuple as (A, X, P, R). Here, A is the set of actions, including the sampling rate and modulation type as control and networking parameters. The reinforcement learning selects an action a, including a sampling rate and a modulation type combination from the action set $\{A = a_1, a_2, a_3, \ldots, a_m\}$.

The state of the system is defined as a set of variables to describe the different levels of stability of the system. System state is represented by $\{X = (x_1, x_2, x_3, \ldots, x_n)\}$ with n states in total. At each time step, the current system state is denoted as x, where $x \in X$. The transition probability is denoted by function $P(x, a, x')$, which represents the probability of taking action a at state x and transiting to state x'. $R(x, a, x')$ denotes the reward function that determines the reward value for the state change from x to x' under action a. The objective is to find an optimal policy that maximizes the cumulative reward $\sum_{t=0}^{\infty} r(t) * R(x(t), \pi(x(t)), x'(t+1))$, where $r(t)$ is the discount factor and satisfies $0 \leq r(t) < 1$.

Note that we use discount factor $r(t)$ to discount the time value of rewards. Instead of a constant discount factor, we set the discount factor function to be computed by $r(t) = 1 - 0.5^{(t)}, t \geq 0$. This is because we care less about future rewards towards computing an optimal policy in the first few time steps of reinforcement learning. We use the change of MAE (i.e., system state) to formalize the reward function. The change of system state in the starting phase is not important in comparison to the change of system state in later time steps of reinforcement learning (i.e., stable phase). Thus, we set a small initial discount factor to reduce the value of future rewards in initial time steps, and increase the discount factor to increase the value of future rewards in later time steps.

In our setting, the reward is derived by

$$R(x(t), a, x'(t+1)) = f(\Delta MAE, t), \tag{1}$$

where δMAE is used to measure the change of the system state. As the transition probability $P(x, a, x')$ is unknown, we chose a Q-learning scheme rather than using a model-based reinforcement learning.

C. Q-learning Scheme

We leverage Q-learning to find the optimal policy, which consists of two phases: a learning phase followed by a frozen phase. The Q-learning scheme generally assigns a value to each state-action pair in the system, denoted by the Q-factor. The Q-factor is represented by $Q(x, a)$, indicating the value of action a at state x. There are a total of $|X| \times |A|$ Q-factors if there are $|X|$ system states and $|A|$ action sets in the designated system. The Q-factors are first given small values, equal or close to 0.

In the learning phase, the Q-learning scheme will continuously update the Q-factors. Particularly, the immediate reward $R(x, a, x')$ of each state transition will be used to update the Q-factors at each time step. For example, if the reward is positive (i.e., good state transition) after one action is selected at a given state, the reinforcement learning scheme will increase the Q-factor for the state-action pair. Otherwise, the reinforcement learning scheme will decrease the Q-factor to reduce the chance of selecting that action at the given

state. This Q-factor updating process will continue for a predetermined number of time steps. Finally, at the end of the learning phase, the Q-learning scheme will produce a set of actions with the highest valued Q-factor as the optimal actions at corresponding states. In the frozen phase, the Q-learning scheme will no longer update the Q-factors. The set of action and state pairs will be considered as the optimal policy. By adopting the optimal policy, we can compute the average reward in the frozen phase to estimate the performance of reinforcement learning.

D. Learning Policies

To achieve fast convergence of the learning process as required for control systems, we design the following three new policies to regulate and improve reinforcement learning performance for the IIoT system. Particularly, we design Policy I to reduce the action space, Policy II to reduce the explorations, and Policy III to stabilize the system quickly, as follows.

Policy I. Reducing Action Space: We assign actions (e.g., sampling rate and modulation type) into pairs and rank the actions following IEEE 802.15.4 bit error rate (BER) performance. We can directly map the BER to packet loss due to the low latency and high reliability requirements of the IIoT setting. Intuitively, the higher sampling rate increases resource contention, which increases the BER and packet loss. Thus, the action sets, such as a higher sampling rate with a low noise resistant modulation type (e.g., quadrature phase shift keying (QPSK)) will be eliminated from the action sets, preventing the potential for bad control performance. Since the data rate of some modulation types cannot satisfy the delay requirement of the high sampling rate, we can reduce the action space by removing unmatched action pairs.

Policy II. Reducing Explorations: We assign the exploration thresholds and MAE threshold to ensure that reinforcement learning can minimize the exploration rate when the stable state action has been identified or a stable state action has not been found after a number of iterations. This is because, once the current system state is stable, the control system should prefer to stay in a stable state rather than explore other potentially unstable states. We also assign a low exploration rate to ensure convergence when the system state is unstable after a number of iterations.

Specifically, we choose MAE threshold and exploration thresholds (determining a low exploration or a high exploration) based on the system stability and the number of iterations. For example, we can set the MAE threshold to be 1, indicating that a system state measured by MAE that is smaller than 1 is considered is a stable state, and a system state greater than 1 is considered as an unstable state. In addition, based on the number of iterations, we set a low exploration threshold to be 0.4926 (i.e., 30 iterations) and a high exploration threshold to be 0.4854 (i.e., 60 iterations). Note that the exploration rate decreases gradually during iterations.

Then, when the system is in a stable state (i.e., system state satisfies the MAE threshold) and the exploration is insufficient (i.e., exploration rate satisfies low exploration threshold) or the system is in an unstable state and the exploration is sufficient (i.e., exploration rate satisfies high exploration threshold), the system selects the action with the maximum Q-value. Otherwise, the system selects another action. We demonstrate the implementation in the next section.

Policy III. Stabilizing System Quickly: Due to the fact that the physical system's stability is not restricted to a single point in the state space, we set blurry boundaries on the states around setpoints. In more detail, the physical system can be considered stable when the system state fluctuates around the setpoint within a certain threshold. To do so, we first define system states with certain values. Then, we add a condition (i.e., deviation) to further make decision on the system states. In this way, when the current system state satisfies the threshold for stable states, the reinforcement learning will consider the current system as stable.

V. Algorithm Design and Analysis

To implement our scheme illustrated in Section IV, we detail the algorithm design and analysis based on the new policies designed for reinforcement learning-enabled IIoT systems.

A. Algorithms

We implement Algorithm 1 for the learning phase and Algorithm 2 for the frozen phase of the Q-learning algorithm, as described in Section IV-C. In Algorithm 1, we introduce the procedures for updating the Q-factors. The first step is the initialization (line 1), which initializes some key setting parameters for the reinforcement learning module, including the maximum iterations $ItLe_{MAX}$, learning rate α^k, a scaling constant η close to 1, exploration rate p^k determined by $G1$ and $G2$ and state threshold Th_{STAB}, low exploration threshold Th_{Lexp}, and high exploration threshold Th_{Hexp}. Here, $G1$ and $G2$ can be positive values, such as 1000 and 2000. Then, as the iteration progresses, k increases, and the exploration rate p^k decreases gradually. The output of Algorithm 1 is the optimal policy $Q(x, a)$.

Before the maximum number of iterations is reached, when the current state is stable ($MAE < Th_{STAB}$) and the exploration rate satisfies the threshold for the low exploration rate ($p^k < Th_{Lexp}$), the reinforcement learning will assign an action with the highest Q-value. Otherwise, when the current state is stable, but the exploration is not sufficient, reinforcement learning will assign an action based on a ranking distribution.

In addition, when the current state is unstable and the exploration rate satisfies the threshold for the high exploration rate ($p^k < Th_{Hexp}$) (i.e., sufficient exploration), the reinforcement learning will assign an action with the highest Q-value. Otherwise, when the current state is unstable, but the exploration is not sufficient, the reinforcement learning scheme will assign an action based on a ranking distribution.

Thus, we conclude that our reinforcement learning scheme assigns the highest Q-valued action when the system is stable after only a few iterations, as well as when the system is unstable after a number of iterations. Otherwise, the reinforcement learning scheme explores other actions. Generally speaking, the time complexity is $O((|A| - c_1)(|X| - c_2)^2)$ for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2020.2980909, IEEE Journal on Selected Areas in Communications

each iteration, where $|A| - c_1$ is the size of the reduced action sets after the designed policies are applied and $|X| - c_2$ is the size of the reduced states after the policies are applied.

Algorithm 1: Q-learning for Reinforcement Learning-enabled WCPS

Result: $Q(x, a)$
initialization: $Q(x, a) \leftarrow 0$, $k = 1$, α^k, η, p^k, $G1$, $G2$, $ItLe_{MAX}$, Th_{STAB}, Th_{Lexp}, Th_{Hexp};
while $k < ItLe_{MAX}$ **do**
  $p^k = G1/(G2 + k)$;
  if $MAE < Th_{STAB}$ **then**
    if $p^k < Th_{Lexp}$ **then**
      select $\arg \max_{a \in A(x)} (Q(x, a))$;
    else
      select action a based on ranking distribution;
    end
  else
    if $p^k < Th_{Hexp}$ **then**
      select $\arg \max_{a \in A(x)} (Q(x, a))$;
    else
      select action a based on ranking distribution;
    end
  end
  update $Q(x, a)$: $Q(x, a) \leftarrow (1 - \alpha^k)Q(x, a) + \alpha^k[R(x, a, x') + \eta \max_{a' \in A(x')} (Q(x', a'))]$;
  $k = k + 1$;
end
STOP

Algorithm 2: Average Reward Ω^k of the Optimal Policy

Result: $\Omega^k = \frac{REW_{TOT}}{TIM_{TOT}}$
initialization: π, k, REW_{TOT}, TIM_{TOT}, $ItFr_{MAX}$;
while $k < ItFr_{MAX}$ **do**
  select $\pi(x(k))$;
  $REW_{TOT} \leftarrow REW_{TOT} + r(t) * R(x, \pi(x(k)), x')$;
  $TIM_{TOT} \leftarrow TIM_{TOT} + t(x, \pi(x(k)), x')$;
  $k = k + 1$;
  set $x \leftarrow x'$;
end

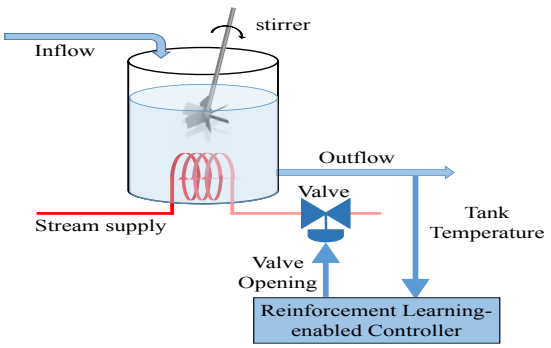

Fig. 2. CSTR System Overview

In Algorithm 2, we show the procedure of the frozen phase, which estimates the average reward of the optimal policy calculated as Ω^k. In the initialization step, reinforcement learning initializes the learned optimal policy $Q(x, a)$, k, REW_{TOT}, TIM_{TOT}, and maximum iterations $ItFr_{MAX}$. Then, the algorithm selects actions based on the optimal policy and updates the transition reward denoted by $R(x, \pi(x(k)), x')$ in Equation (1) and transition time denoted by $t(x, \pi(x(k)), x')$. The transition time is computed by $t(x, \pi(x(k)), x') = \frac{Cycle\ Time}{Sampling\ Rate}$. Here, *Cycle Time* is a constant value that refers to the time of closed-loop sensing and actuation process per sampling cycle. Finally, it returns the average reward Ω^k. The time complexity for Algorithm 2 is $O(n)$.

B. Algorithm Analysis

Without Policy I, there are $|X| \times |A|$ number of Q-factors, and the reinforcement learning exploration space is $|X| \times |A|$. In each iteration, the time complexity of the reinforcement learning model is $O(|A \times X|^2)$, where $|A|$ is the number of action sets and $|X|$ is number of states. In contrast, when Policy I is used, the number of Q-factors is reduced to $|X| \times (|A| - c_1)$, where c_1 is the number of eliminated action sets. The exploration space is then reduced to $|X| \times (|A| - c_1)$. The time complexity of each iteration is reduced to $O((|A| - c_1)|X|^2)$.

With Policy II in place, the number of iterations to achieve the optimal policy is reduced. This is because, once the system is in stable state and the exploration rate is minimized, the reinforcement learning does not need to further iterate for better actions. Thus, the number of iterations is reduced after Policy II is used. As demonstrated by the experimental results in Fig. 13, the total reduction on the number of iterations is maximally 37.5 % when our policies are applied.

Without Policy III, the time complexity of each iteration is $O((|A| - c_1)|X|^2)$. Nonetheless, when Policy III is used, in each iteration, the time complexity is reduced to $O((|A| - c_1)(|X| - c_2)^2)$, where c_2 indicates the eliminated number of states. This is because the total iterations needed to obtain the optimal result is a polynomial of the number of states. Thus, the number of iterations is further reduced with Policy III applied.

VI. Performance Evaluation

We have conducted a thorough performance evaluation to validate the feasibility of our approach in IIoT systems. We carry out extensive experiments using Matlab/Simulink with implementation of reinforcement learning module, physical plant, observer and controller, and wireless network, as shown in Fig. 3. In the following, we first present our experimental methodology and system implementation, and then detail the experimental results.

A. Methodology

We carried out our experiments in the following three phases: *Phase I: System Observation Phase* observes the interactions of control and networking systems, *Phase II: Performance Validation Phase* validates the effectiveness of integrating the reinforcement learning scheme into IIoT system by collecting state observer results as inputs to the reinforcement learning module and providing control and networking parameters as inputs to the IIoT system, and *Phase III: Performance Improvement Phase* optimizes reinforcement learning

performance in terms of reducing state space and improving system stability by incorporating the new policies introduced in Section IV-D.

We use the MAE to measure the control results of our reinforcement learning-enabled IIoT system. The MAE measures the absolute error between two continuous variables, such as the absolute error between step response under sufficient network resources (e.g., wired network) and insufficient network resources that leads to packet loss. We use three noise level (-80 dBm, -78 dBm, and -76 dBm) to simulate the light, moderate, and heavy channel noise, which leads to link failures of 15 % to 98 % [13] and affects the packet delivery rate (PDR). Note that 1 dB is relative to 1 mW (dBm). A high PDR indicates that a high ratio of packets has been successfully delivered.

We use the number of iterations to measure the performance of the reinforcement learning module. The number of iterations required to achieve a stable state indicates the time used by the reinforcement learning to converge. Our goal is to reduce the number of iterations, as a smaller number of iterations indicates faster convergence. We also use error bars to denote the 95 % confidence estimates of our reinforcement learning model. We run the reinforcement learning ten times and compute the average iterations, as well as the maximum and the minimum iterations under different noise levels (i.e., light, moderate, and heavy) individually.

B. System Implementation

Recall that, our IIoT testbed consists of four key elements, including physical plant, networking system, control system, and reinforcement learning module.

Physical System: In our study, we consider a temperature control system, which is a representative example for numerous industrial process systems (oil refineries, chemical plants, etc.) [13], [38]. For many chemical plants, the fluid temperature is a single setpoint, because a particular temperature is required for the desired chemical reactions. We use the CSTR as our physical system [39] to demonstrate our design. As shown in Fig. 2, CSTR has one main tank for chemical reactions, one inflow for product input, and one outflow for effluent output. The stream supply provides heat exchange. A stirrer is used to fully mix the products. The controller sends control commands to the stream valve to control the stream rate so that the tank temperature can be adjusted. To maintain the required temperature in the main tank, an MPC controller can transmit control signals to the valve to adjust the stream rate based on the sensed temperature (system state) from the outflow. The key parameters used in CSTR are shown in Table III.

To model the CSTR, we use the reactor mass balance and reactor energy balance as the foundation of the state space function. The reactor mass balance is used to describe the change rate of material mass while the reactor energy balance is used to describe the change rate of energy (i.e., heat). Then, we linearize the nonlinear balance equations into a state space function for our control system.

According to the model of CSTR [39], we first present the reactor mass balance as $V\frac{dC_A}{dt} = q(C_{Ai} - C_A) - Vr_A$, where V is the volume of the CSTR main tank, q is the inflow rate, C_{Ai} is the product concentration of inflow, C_A is the product concentration in the main tank, and r_A is the reaction rate per unit volume. We have $r_A = k_0 C_A \exp(\frac{-E}{RT})$, where k_0, E, and R are constant values for reaction rate, activation energy and ideal gas constant, and T denotes the reactor temperature. Then, the reactor energy balance is denoted as $V\rho C_p \frac{dT}{dt} = q\rho C_p(T_i - T) - (-\delta H)Vr_A + \rho C_{ps} q_s \left[1 - \exp\left(\frac{-hA}{q_s \rho_s C_{ps}}\right)\right](T_{is} - T)$, where $-\delta H$, hA, T_i, T_{is}, and ρ are constant values for reaction heat, heat transfer coefficient, inflow temperature, stream supply temperature, and product density, respectively. Also, C_p and C_{ps} denote the specific heats of the inflow and stream fluids, and q_s and ρ_s are the stream flow rate and stream density, respectively [39].

Then, we linearize above Equations and cast them into the state space function as Equations (2) and (3), listed below.

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}, \tag{2}$$

where $\tilde{u}$ is vector of the control signals for the valve, $\tilde{x}$ is the vector of state variables, and A and B are the *Jacobian* matrices for the nominal value of $\tilde{u}$ and $\tilde{x}$. The output is related to the state vector by

$$\tilde{y} = C\tilde{x}, \tag{3}$$

where C and $\tilde{y}$ are the output matrix and system output (i.e., tank temperature).

Finally, the state space function in Equation (2) can be written as Equations (4) [39].

Particularly, $K_s = k_0 \exp\left(\frac{-E}{RT_{Ss}}\right)$, where k_0 is the constant value of reaction rate [39]. Also, $K_s{}' = k_0 \exp\left(\frac{-E}{RT_{Ss}}\right)\left(\frac{-E}{RT_{Ss}{}^2}\right)$, and $\tilde{y} = \begin{bmatrix} C_A - C_{ASs} \\ T - T_{Ss} \end{bmatrix}$.

EKF State Observer and MPC Controller: The EKF state observer for sensing can robustly estimate the system state under packet loss [13], [14], predicting the current system state based on the previous system state and recursively updating the prediction accuracy with the actual system output. In addition, it receives the control command computed by the MPC controller to improve prediction accuracy. The state observation results from EKF will be used as inputs to the reinforcement module.

We use a standard MPC controller that solves a fundamental finite horizon linear-quadratic regulator (LQR) optimal control problem. We leverage the Gurobi Optimizer to solve the LQR optimal control problem in Matlab/Simulink [40]. The MPC controller computes finite horizon predictions by linearizing the system around the prior control signal $u(k-1)$.

Buffer for Actuation: The buffer for actuation stores a sequence of actuation commands per time step. When new actuation commands are not received, the system directly uses the stored actuation commands from the buffer.

Wireless Network: We adopt the topology setting, deployment, and noise traces from the open source WCPS testbed [9], as it integrates a mature wireless network environment following the WirelessHART network protocol. The wireless network simulates a WSAN in WCPS with a 16-node topology and an average 4-hop distance between sensor, controller, and actuators. We additionally use TOSSIM, a standard simulator

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2020.2980909, IEEE Journal on Selected Areas in Communications

$$\dot{\tilde{x}} = \begin{bmatrix} -\frac{q}{V} - K_s & -C_{ASs}K'_s \\ -(\frac{\delta H}{\rho C_p})K_s & (-\frac{q}{V}) + (\frac{(-\delta H)C_{ASs}}{\rho C_p})K'_s + (\frac{-\rho C_{ps}}{\rho C_p V})q_s + (\frac{\rho_s C_{ps}}{\rho C_p V})q_s \exp(\frac{-hA}{q_s \rho C_p}) \end{bmatrix} \begin{bmatrix} C_A - C_{ASs} \\ T - T_{Ss} \end{bmatrix} + \begin{bmatrix} (\frac{C_{Ai} - C_{ASs}}{V}) & 0 \\ \frac{T_i - T_{Ss}}{V} & \frac{\rho_s C_{ps}}{\rho C_p V}(T_{is} - T_{Ss})\{q_s[(-\exp(\frac{-hA}{q_s \rho C_{ps}}))((\frac{hA}{{q_s}^2 \rho C_{ps}}))] + [1 - \exp(\frac{-hA}{q_s \rho C_{ps}})]\} \end{bmatrix} \begin{bmatrix} q - q_{Ss} \\ q_s - q_{sSs} \end{bmatrix} \quad (4)$$

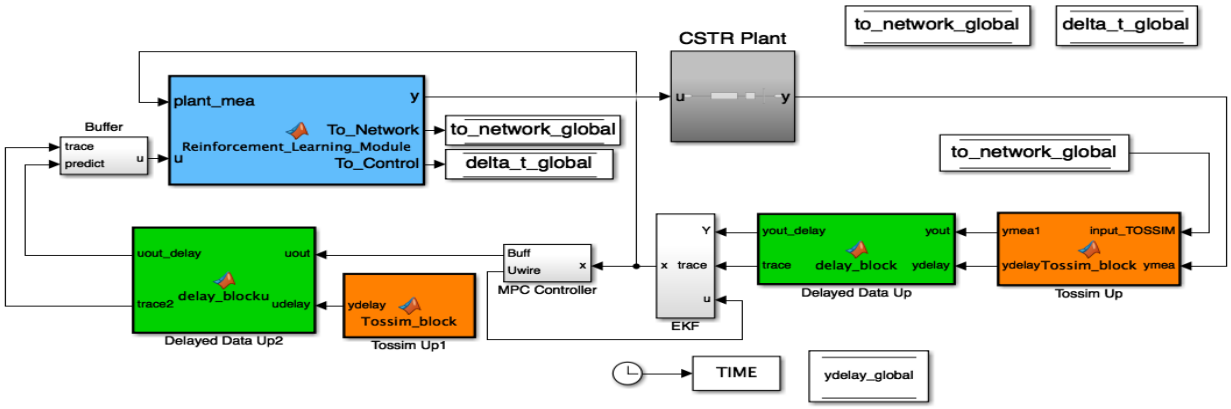

Fig. 3. System Implementation in Matlab/Simulink

Symbols	Descriptions
$\tilde{x}$	Vector of state variables
$\tilde{u}$	Vector of control signals for the valve
$\tilde{y}$	System output (i.e., tank temperature)
A and B	Jacobian matrices for the nominal value of $\tilde{u}$ and $\tilde{x}$
C	Output matrix
V	Volume of the CSTR main tank (e.g., 100 L)
C_A	Product concentration in the main tank (e.g., 0.0882 mol/L)
C_{Ai}	Product concentration of inflow (e.g., 1 mol/L)
C_p, C_{ps}	Specific heat of inflow and stream fluids (e.g., 1 cal/(g K))
q	Inflow rate (e.g., 100 L/min)
q_s	Stream flow rate (e.g., 100 L/min)
ρ, ρ_s	Inflow and Stream density (e.g., 1×10^3 g/L)
T	Reactor temperature (e.g., 300.15 K (27 C))
T_i	Inflow temperature (e.g., 273.15 K)
T_{is}	Stream supply temperature (e.g., 350 K)
hA	Heat transfer coefficient (e.g., 7×105 cal/(min K))
δH	Reaction heat (e.g., 2×10^5 cal/mol)
k_0	Reaction rate (e.g., 7.2×10^{10} min^{-1})
$\frac{E}{R}$	Activation energy term (e.g., 1×10^4 K)
q_{Ss}, q_{sSs}, T_{Ss}, C_{ASs}	Stable state values of effluent concentration

TABLE III. IIoT System Parameters

for TinyOS [41], to simulate the WSAN with received signal strength, realistic channel noise, and link model. In addition, the noise level can be adjusted by varying the noise offset for light, moderate, and heavy noise levels (i.e., -80 dBm, -78 dBm, and -76 dBm), which leads to link failures of 15 % to 98 % [13]. The output of the reinforcement learning module will be given to the wireless network so that the network performance can be adapted to different noise conditions by applying proper modulation types.

Testbed Implementation: We implemented our reinforcement learning-enabled IIoT system in Matlab/Simulink, as shown in Fig. 3. This figure contains modules that we implemented ourselves, along with modules from the open source WCPS. To be specific, we implemented the CSTR system in Simulink as illustrated in Section VI-B with key parameters described in Table III. We also implemented and integrated our reinforcement learning module (i.e., Reinforcement_Learning_Module) with the open source WCPS. Our reinforcement learning module takes a system state from the EKF as input and computes a control parameter (i.e., delta_t_global) and a networking parameter (i.e., to_network_global) at each iteration. Note that the reinforcement learning module injects the parameters to the data store memory as global variables to be used by the other Simulink blocks (i.e., CSTR plant and Tossim_block).

C. Evaluation Results

Results in System Observation Phase: In the system observation phase, we can see from Fig. 4 that the high sampling rate will achieve the best control performance in the network with sufficient resources (left). At the same time, in the wireless network with insufficient resources (right), the high sampling rate will lead to the poor control performance, with considerable fluctuations around 30 C, which is 10 % above the desired temperature, as it produces strong network resource contention (e.g., severe packet loss).

Fig. 5 shows the control performance with respect to different sampling rates, which yields different levels of corresponding resource contention. As expected, both the highest sampling rate and the lowest sampling rate lead to poor control performance, because the highest sampling rate causes strong resource contention and the lowest sampling rate takes too long to stabilize the system. Thus, we chose a sampling rate (e.g., 7 Hz) that achieves the best control performance under a reasonable level of network resource contention. Then, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2020.2980909, IEEE Journal on Selected Areas in Communications

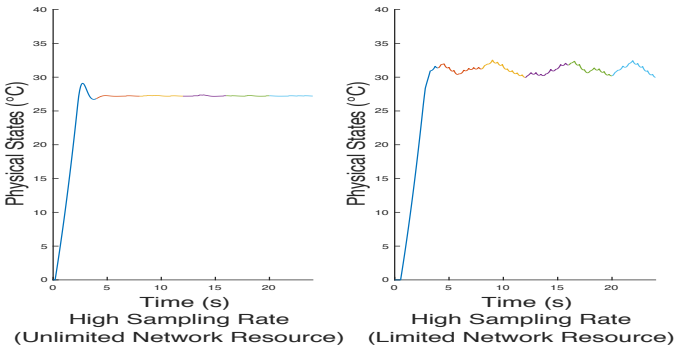

Fig. 4. Control Performance vs. Network Resource

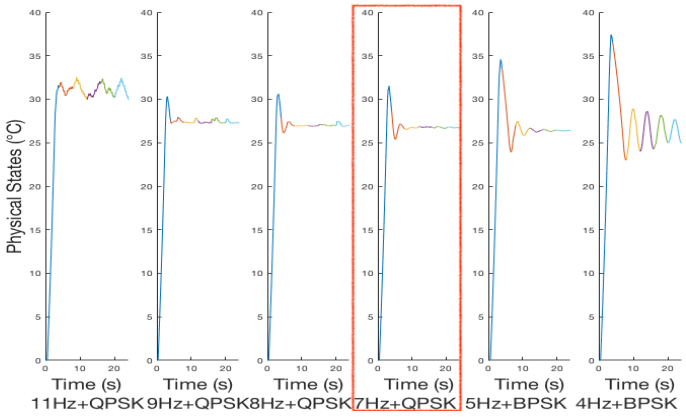

Fig. 5. Control Performance vs. Sampling Rate & Modulation Type

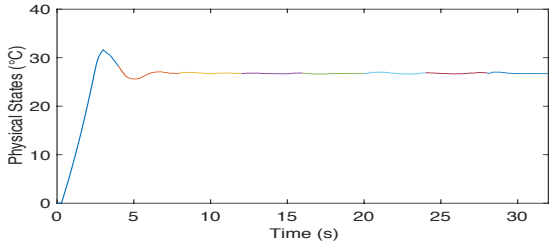

Fig. 6. Control Performance of Benchmark Control Parameter (i.e., 7Hz)

use this sampling rate (i.e., 7 Hz) as our benchmark control parameter. We apply the benchmark control parameter to the system and observe the control results from Fig. 6.

We can conclude that the benchmark outperforms the others in a constrained resource wireless network, as shown in Fig. 6. Nonetheless, the performance of benchmark changes under a dynamic communication environment (i.e., different communication noise levels), as shown in Figs. 7, 8, and 9. Particularly, as seen in Fig. 7, the sampling rate 7 Hz (red box) can stabilize the control system and remain stable quickly in comparison to other sampling rates under light communication noise. Nonetheless, as shown in Fig. 8, the sampling rate 5 Hz (red box) outperforms the other rates under moderate communication noise. The reason for this is that a lower sampling rate can reduce the resource contention, improving the packet (e.g., control command) delivery. Thus, a lower sampling rate leads to a better control performance. When the communication noise is high (severe packet loss), an even

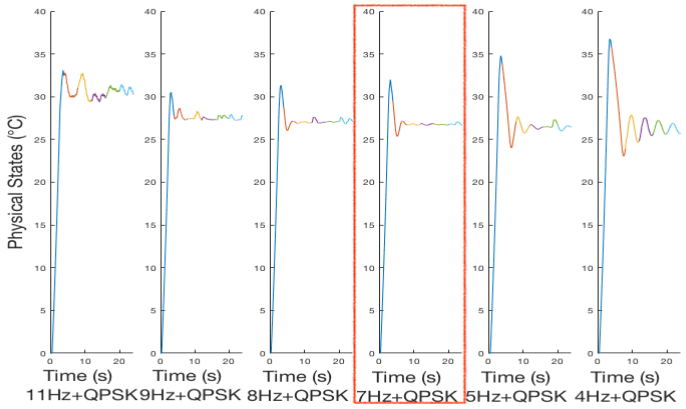

Fig. 7. Control Performance vs. Sampling Rate & Modulation Type under Light Communication Noise

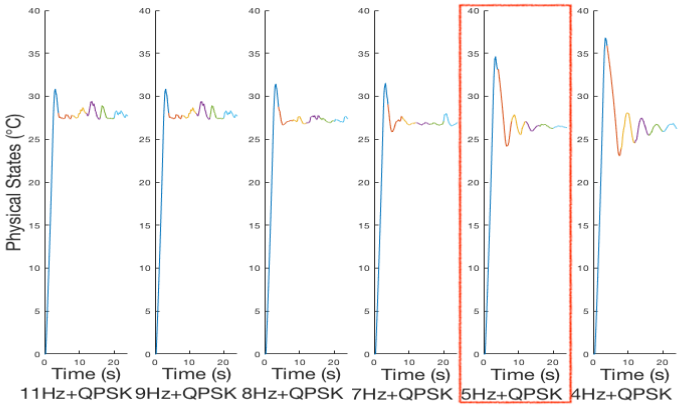

Fig. 8. Control Performance vs. Sampling Rate & Modulation Type under Moderate Communication Noise

lower sampling rate 4 Hz (red box) outperforms the others, as shown in Fig. 9. This is because the low sampling rate results in a higher packet delivery rate under high communication noise due to reduced resource contention.

From the experimental results above, we can conclude when the communication resources are insufficient, we should reduce the sampling rate to avoid severe packet losses. In addition, in a varying communication environment (i.e., variable noise level), we should dynamically select the sampling rate to achieve the best communication performance. Further, when the communication noise level varies, we should assign different modulation types to the network, which improves the packet delivery rate of a severely noisy communication channel. For example, when the control performance decreases due to a noisy communication channel, we should assign a more noise-resistant modulation type, such as Binary Phase Shift Keying (BPSK) instead of Quadrature Phase Shift Keying (QPSK).

Results in Performance Validation Phase: In the performance validation phase, Figs. 10, 11, and 12 illustrate the performance of the reinforcement learning under different noise levels. In Fig. 10, we can observe that the reinforcement learning-based control performance improves with increasing rounds of iteration (left to right and top to bottom). Note that we reset the system state to repeatedly learn the environment.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2020.2980909, IEEE Journal on Selected Areas in Communications

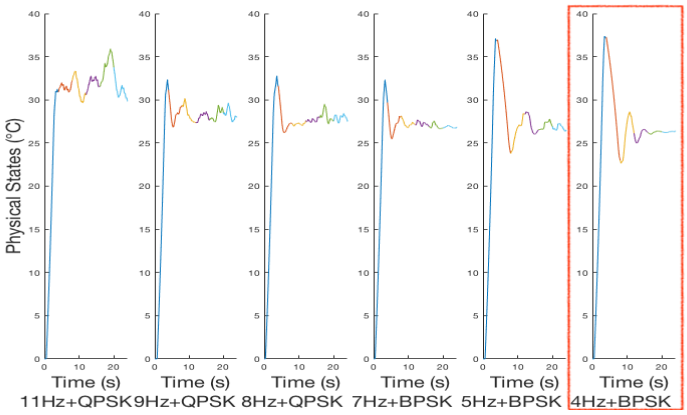

Fig. 9. Control Performance vs. Sampling Rate & Modulation Type under Heavy Communication Noise

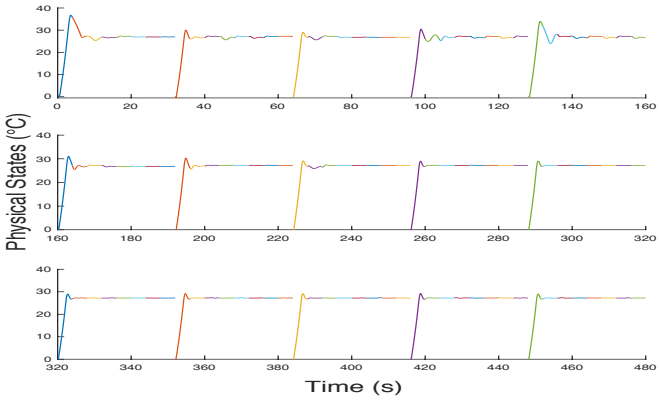

Fig. 10. Reinforcement Learning Performance under Light Communication Noise, each curve shows performance after one additional iteration of the learning algorithm

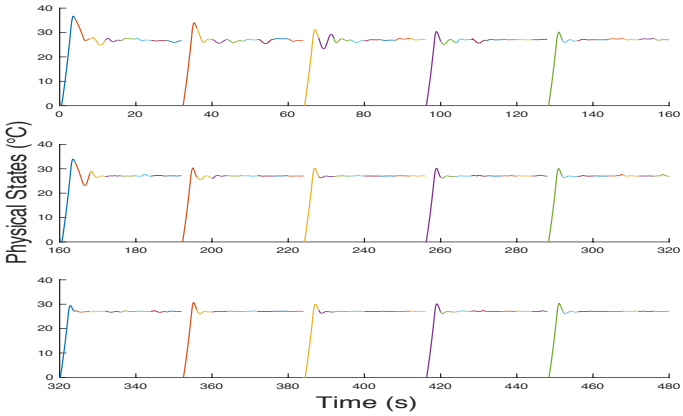

Fig. 11. Reinforcement Learning Performance under Moderate Communication Noise, each curve shows performance after one additional iteration of the learning algorithm

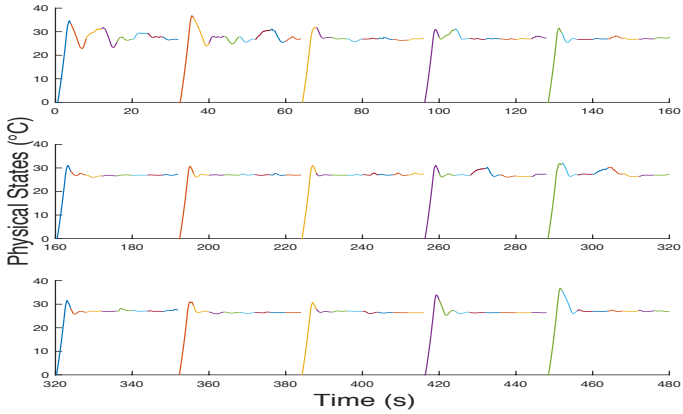

Fig. 12. Reinforcement Learning Performance under Heavy Communication Noise, each curve shows performance after one additional iteration of the learning algorithm

This is because once the system is stable, reinforcement learning can no longer explore the environment sufficiently. From all three figures, we observe that the control system is unstable in the first five rounds (top row) of iteration, but gradually becomes stable between rounds six and ten (middle row). Once the system becomes stable, the reinforcement learning no longer explores new actions and continues with the optimal actions. Note also that the reinforcement learning converges more quickly in Fig. 10 in comparison to Figs. 11 and 12. These results show that the policies designed in Section IV-D are more effective at reducing unnecessary explorations when the communication noise is light, as less randomness is introduced to the reinforcement learning process. Note that each round contains 8 learning iterations.

In addition, as shown in Figs. 11 and 12, the reinforcement learning quickly adapts to the different communication conditions and computes optimal actions after only a few rounds of learning. As shown in Fig. 11, the control system is unstable in the first seven rounds (second peak of the middle row), becoming stable after few more rounds. Similarly, as shown in Fig. 12, the reinforcement learning stabilizes the control system after about ten rounds.

Results in Performance Improvement Phase: In the performance improvement phase, Fig. 13 illustrates the confidence estimates of our reinforcement learning module in the evaluated IIoT system. From the figure, the left two bars (blue) show the performance of the reinforcement learning under light communication noise.

Note that the left-most bar of each pair shows the number of iterations for reinforcement learning without the policies we designed in Section IV-D, which are always higher. The number of iterations is calculated by the number of rounds multiplied by the number of iterations per round. Thus, in the right bar of each pair, we can see that the number of iterations declines when our designed policies are used. As we can see from the figure, our designed policies are capable of reducing the number of iterations in every setting. In addition, we observe that the error is smaller when our policies were used (right bar) in the light communication noise scenario, as the light noise marginally affects the performance of reinforcement learning compared to moderate and heavy communication noises. The middle two bars (yellow) show the performance of the reinforcement learning under moderate communication noise. Without the policies applied (yellow, left), the number of iterations is higher than the right bar (yellow, right) in which the policies are in place. Similarly, the right two bars (red) show the performance of the reinforcement

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2020.2980909, IEEE Journal on Selected Areas in Communications

learning under heavy communication noise.

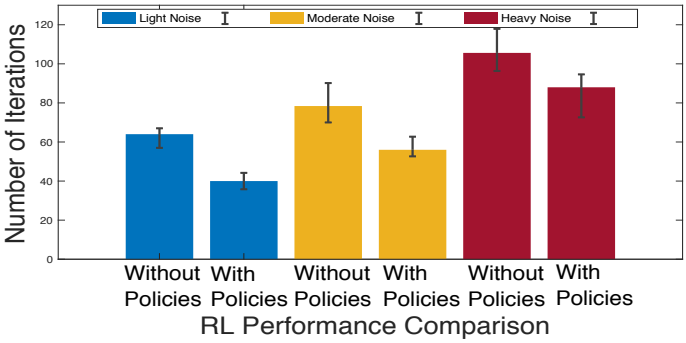

Fig. 13. Effectiveness of Policies Applied under Different Noise Levels. Left bars: Section IV-E policies not implemented, right bars: Section IV-E policies in use. 95 % confidence intervals are shown

As the result, our policies improve the reinforcement learning performance at most by 46.57 % (light communication noise) and 22.99 % on average. In addition, our reinforcement learning (with policies) can obtain the optimal policy in a little as 40 iterations and in 61 iterations on average. In comparison, reinforcement learning without the policies achieves a minimum of 64 iterations and 82.3 iterations on average.

From the figures, we can conclude that the performance improvement (iteration reduction) of the policies designed is the highest under light communication noise (left), followed by moderate communication noise, and lastly heavy communication noise. This is because the heavy noise affects the performance of reinforcement learning the most, as it raises additional randomness in the system. Under the light communication noise, the policies can be the most effective, as the system has less uncertainty from the communication errors. Thus, the experience from systematic trial and error can be utilized, which contributes to the best performance.

VII. Discussion

In the following, we outline several future research directions.

Applying Co-design Methodology to Other IIoT System Parameters: Recall that in this study, our co-design approach enables a learning module to interact with both networking and control components. We generalize the co-design methodology approach via four steps that can be adopted to conduct configurations in other IIoT system parameters: (i) identifying the tight interactions between IIoT system parameters, (ii) specifying the key parameters that describe the interactions, (iii) adopting appropriate techniques to conduct configurations on the key parameters, and (iv) leveraging realistic IIoT testbeds to test and verify the effectiveness of the approach.

For example, we can leverage the co-design approach in IIoT control and networking systems, such as sampling and network resource scheduling co-design. Specifically, we first identify the tight interactions between sampling and network resources (e.g., a higher sampling rate requires more network resources to transmit data). We then specify the key parameters such as sampling rate, superframe duration, and scheduling in CSMA/CA (a MAC layer parameter). We further adopt the reinforcement learning techniques to conduct configurations on the sampling rate, superframe duration, and scheduling. Note that a longer superframe duration and adaptive scheduling can improve network throughput, which accommodates the network resource demands raised by a higher sampling rate. Finally, we implement the reinforcement learning algorithm that automatically configures the sampling rate, superframe duration, and scheduler to test and verify the performance of WCPS.

Applying Reinforcement Learning to Other IoT Applications: As reinforcement learning is effective in making decisions under unknown environments via systematic trial and error, we shall apply reinforcement learning to other IoT applications, including smart transportation, smart home, and connected health, among others [42]. Reinforcement learning can help smart-world systems to conduct decision-making to adapt to dynamic environments.

Nonetheless, diverse requirements are raised from the distinct characteristics inherent to smart-world systems (e.g., ultra-high reliability and ultra-low latency) and must be taken into consideration when applying reinforcement learning into the system. We shall improve the performance of reinforcement learning algorithms (e.g., convergence) to meet the strict requirements via exploring the tradeoff between exploration and exploitation. It is known that the sufficient exploration of the algorithm usually computes an optimal solution (e.g., policy) with the cost of slow convergence. At the same time, lower exploration and higher exploitation (e.g., greedy algorithm) may converge to a sub-optimal solution quickly. Thus, to adopt the reinforcement learning, we shall study the tradeoff between exploration and exploitation and conduct fine-tuning on the hyperparameters via sufficient experimentation.

VIII. Final Remark

In this paper, we have proposed the control and networking co-design approach to improve the system performance for a highly coupled IIoT system. In order to achieve the co-design, we have proposed a reinforcement learning module to reconfigure both the control and networking systems dynamically and automatically. We have also implemented the reinforcement learning-based co-design approach into a simulated IIoT system with realistic wireless cyber-physical components. In our reinforcement learning-enabled IIoT system, the reinforcement learning module can automatically configure the networking and control parameters under a dynamic environment.

To further improve the performance of reinforcement learning for IIoT, we have designed three new policies with regard to the characteristics of industrial systems, including the assigning of action pairs, the minimization of exploration under stable states, and the design of blurry state boundaries. Finally, we have conducted extensive experiments to demonstrate that our designed approach can stabilize the highly-coupled IIoT system quickly under a dynamic industrial environment. Last, but not least, with the designed policies applied, our reinforcement learning approach significantly outperforms standard

reinforcement learning all levels of dynamic disruption in industrial environments.

REFERENCES

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial Internet of things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.

[2] J. Gläscher, N. Daw, P. Dayan, and J. P. O'Doherty, "States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning," *Neuron*, vol. 66, no. 4, pp. 585–595, 2010.

[3] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 703–711.

[4] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[5] 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.

[6] G. Zhao, M. A. Imran, Z. Pang, Z. Chen, and L. Li, "Toward real-time control in future wireless networks: communication-control co-design," *IEEE Communications Magazine*, vol. 57, no. 2, pp. 138–144, 2019.

[7] H. Xu and L. R. G. Carrillo, "Near optimal control and network co-design for uncertain networked control system with constraints," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 2339–2344.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] "Wcps: Wireless cyber-physical simulator," http://wsn.cse.wustl.edu/index.php/WCPS:_Wireless_Cyber-Physical_Simulator, accessed: 2019-04-13.

[10] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[11] S. Jeschke, C. Brecher, H. Song, and D. B. Rawat, *Industrial Internet of Things: Cybermanufacturing Systems*. Springer, 2017.

[12] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.

[13] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu, "Wireless routing and control: A cyber-physical case study," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–10.

[14] Y. Ma, D. Gunatilaka, B. Li, H. Gonzalez, and C. Lu, "Holistic cyber-physical management for dependable wireless control systems," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 1, p. 3, 2018.

[15] 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, 2017.

[16] S. S. Oyewobi, G. P. Hancke, A. M. Abu-Mahfouz, and A. J. Onumanyi, "An effective spectrum handoff based on reinforcement learning for target channel selection in the industrial internet of things," *Sensors*, vol. 19, no. 6, p. 1395, 2019.

[17] H. Yang, A. Alphones, W.-D. Zhong, C. Chen, and X. Xie, "Learning-based energy-efficient resource management by heterogeneous RF/VLC for ultra-reliable low-latency industrial iot networks," *IEEE Transactions on Industrial Informatics*, 2019.

[18] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Reinforcement learning-based computing and transmission scheduling for LTE-U-Enabled IoT," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[19] G. Dartmann, H. Song, and A. Schmeink, *Big Data Analytics for Cyber-Physical Systems: Machine Learning for the Internet of Things*. Elsevier, 2019.

[20] H. Dakdouk, E. Tarazona, R. Alami, R. Féraud, G. Z. Papadopoulos, and P. Maillé, "Reinforcement learning techniques for optimized channel hopping in ieee 802.15. 4-TSCH networks," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 2018, pp. 99–107.

[21] A. Kusiak, "Smart manufacturing," *International Journal of Production Research*, vol. 56, no. 1-2, pp. 508–517, 2018.

[22] B.-h. Li, B.-c. Hou, W.-t. Yu, X.-b. Lu, and C.-w. Yang, "Applications of artificial intelligence in intelligent manufacturing: a review," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 1, pp. 86–96, 2017.

[23] F. Li, K.-Y. Lam, Z. Sheng, X. Zhang, K. Zhao, and L. Wang, "Q-learning-based dynamic spectrum access in cognitive industrial internet of things," *Mobile Networks and Applications*, vol. 23, no. 6, pp. 1636–1644, 2018.

[24] J. Wang, C. Jiang, K. Zhang, X. Hou, Y. Ren, and Y. Qian, "Distributed Q-learning aided heterogeneous network association for energy-efficient IIoT," *IEEE Transactions on Industrial Informatics*, 2019.

[25] C. Arvind and J. Senthilnath, "Autonomous RL: Autonomous vehicle obstacle avoidance in a dynamic environment using MLP-SARSA reinforcement learning," in *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*. IEEE, 2019, pp. 120–124.

[26] J. X. Zou, L. Li, T. Zhang *et al.*, "Time-optimal path tracking for industrial robot: A model-free reinforcement approach," *arXiv preprint arXiv:1907.01348*, 2019.

[27] L. Zeng, E. Li, Z. Zhou, and X. Chen, "Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the industrial internet of things," *IEEE Network*, vol. 33, no. 5, pp. 96–103, 2019.

[28] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8050–8062, 2019.

[29] C. H. Liu, Q. Lin, and S. Wen, "Blockchain-enabled data collection and sharing for industrial IoT with deep reinforcement learning," *IEEE Transactions on Industrial Informatics*, 2018.

[30] V. Léchappé, E. Moulay, F. Plestan, A. Glumineau, and A. Chriette, "New predictive scheme for the control of lti systems with input delay and unknown disturbances," *Automatica*, vol. 52, pp. 179–184, 2015.

[31] J. Holaza, M. Klaučo, J. Drgoňa, J. Oravec, M. Kvasnica, and M. Fikar, "MPC-based reference governor control of a continuous stirred-tank reactor," *Computers & Chemical Engineering*, vol. 108, pp. 289–299, 2018.

[32] W. Yu, D. An, D. Griffith, Q. Yang, and G. Xu, "Towards statistical modeling and machine learning based energy usage forecasting in smart grid," *ACM SIGAPP Applied Computing Review*, vol. 15, no. 1, pp. 6–16, 2015.

[33] W. G. Hatcher and W. Yu, "A survey of deep learning: platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[34] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: deep learning for the Internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[35] Q. Zhang, L. T. Yang, Z. Chen, P. Li, and F. Bu, "An adaptive droupout deep computation model for industrial IoT big data learning with crowdsourcing to cloud computing," *IEEE Transactions on Industrial Informatics*, 2018.

[36] M. Lavassani, S. Forsström, U. Jennehag, and T. Zhang, "Combining fog computing with sensor mote machine learning for industrial IoT," *Sensors*, vol. 18, no. 5, p. 1532, 2018.

[37] N. Gronau, A. Ullrich, and M. Teichmann, "Development of the industrial IoT competences in the areas of organization, process, and interaction based on the learning factory concept," *Procedia Manufacturing*, vol. 9, pp. 254–261, 2017.

[38] J.-S. Ko, J.-H. Huh, and J.-C. Kim, "Improvement of temperature control performance of thermoelectric dehumidifier used industry 4.0 by the SF-PI controller," *Processes*, vol. 7, no. 2, p. 98, 2019.

[39] N. Kamala, "Studies in modeling and design of controllers for a nonideal continuous stirred tank reactor," 2013.

[40] "gurobi optimization," https://www.gurobi.com, accessed: 2019-04-13.

[41] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 126–137. [Online]. Available: http://doi.acm.org/10.1145/958491.958506

[42] H. Song, D. Rawat, S. Jeschke, and C. Brecher, *Cyber-Physical Systems: Foundations, Principles and Applications*. MA: Academic Press, 2016.