Exploring Edge Computing for Multitier Industrial Control

Yehan Ma, Chenyang Lu, *Fellow, IEEE*, Bruno Sinopoli, *Senior Member, IEEE*, and Shen Zeng, *Member, IEEE*

***Abstract*—Industrial automation traditionally relies on local controllers implemented on microcontrollers or programmable logic controllers. With the emergence of edge computing, however, industrial automation evolves into a distributed two-tier computing architecture comprising local controllers and edge servers that communicate over wireless networks. Compared to local controllers, edge servers provide larger computing capacity at the cost of data loss over wireless networks. This article presents switching multitier control (SMC) to exploit edge computing for industrial control. SMC dynamically optimizes control performance by switching between local and edge controllers in response to changing network conditions. SMC employs a data-driven approach to derive switching policies based on classification models trained based on simulations while guaranteeing system stability based on an extended Simplex approach tailored for two-tier platforms. To evaluate the performance of industrial control over edge computing platforms, we have developed *WCPS-EC*, a real-time hybrid simulator that integrates simulated plants, real computing platforms, and real or simulated wireless networks. In a case study of an industrial robotic control system, SMC significantly outperformed both a local controller and an edge controller in face of varying data loss in a wireless network.**

***Index Terms*—Cyber–physical systems, edge computing, machine learning, wireless networked control system (NCS).**

I. Introduction

INDUSTRIAL automation is undergoing a significant transformation driven by the emergence of edge computing and wireless networking technologies. As shown in Fig. 1, the new generation of industrial automation systems features a two-tier computing architecture comprising local and edge computing platforms. Traditionally, industrial automation relies on local controllers running on microcontrollers or programmable logic controllers (PLCs) that are often embedded in control plants with wired connections to sensors and actuators. Edge computing platforms comprise edge servers located on industrial premises. To lower deployment and maintenance costs, edge servers may communicate with control systems through wireless technologies that are increasingly being adopted in industry [1], [2]. Despite their flexibility, wireless networks may suffer from data loss due to environmental factors, such as obstacles, noises, interferences, extreme weather, as well as cyberattacks. The reduced reliability of the wireless network results in degradation of the control performance. Therefore, compared with local controllers, edge servers provide the advantages of computation capacity at the cost of communication reliability and latency.

While edge computing has received significant attention in the industry, research on exploiting edge computing for industrial control has remained limited. In this article, we propose switching multitier control (SMC), a novel approach to optimize control performance at runtime on a two-tier computing platform. SMC dynamically switches control between local and edge platforms in response to changing network reliability. SMC has two salient features. First, in order to overcome theoretical challenges of analyzing control performance of complex systems under network dynamics [3]–[5], it employs *data-driven* approaches to derive control switching policies. A key contribution of this approach is to formulate the platform selection problem as a *data-driven* classification problem, optimal platform classifier (OPC), which can be solved using models extracted from simulations. Second, it extends the Simplex framework [6]–[10] to the distributed two-tier architecture comprising local and edge controllers. The Simplex approach enables SMC to dynamically optimize control performance without sacrificing stability. Furthermore, as a tool to study edge computing for industrial control, it presents wireless cyber–physical simulator-edge computing (WCPS-EC), a real-time hybrid simulator that integrates: 1) real computing platforms; 2) real or simulated wireless networks; and 3) simulated physical plants. The contributions of this article are fivefold.

1) An SMC architecture that dynamically switches control between local and edge controllers in response to changes in network reliability.
2) Data-driven approaches to derive switching policies for multitier control based on the classification models learned from simulations.
3) A controller-switching design that guarantees system stability by extending the Simplex approach to a multitier architecture.

Manuscript received April 17, 2020; revised June 17, 2020; accepted July 6, 2020. Date of publication October 2, 2020; date of current version October 27, 2020. This work was supported in part by NSF under Grant 1646579 (CPS); and in part by the Fullgraf Foundation. This article was presented in part at the International Conference on Embedded Software 2020 and appears as part of the ESWEEK-TCAD special issue. *(Corresponding author: Chenyang Lu.)*

Yehan Ma and Chenyang Lu are with the Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130 USA (e-mail: yehan.ma@wustl.edu; lu@wustl.edu).

Bruno Sinopoli and Shen Zeng are with the Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO 63130 USA (e-mail: bsinopoli@wustl.edu; s.zeng@wustl.edu).

Digital Object Identifier 10.1109/TCAD.2020.3012648

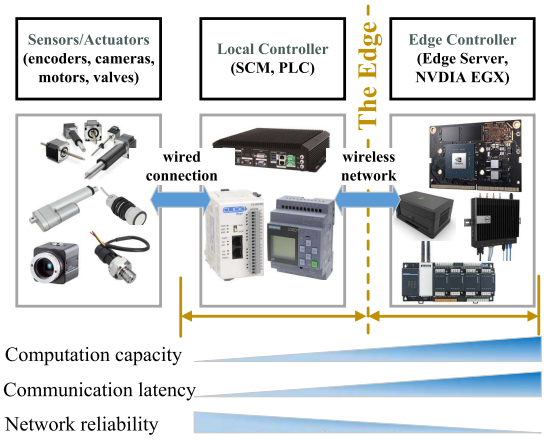

Fig. 1. Multitier control system (local and edge tiers).

4) A real-time hybrid simulator (WCPS-EC) for studying and evaluating control systems based on edge computing.
5) A case study on industrial robotic control demonstrating the advantage of SMC to optimize control performance while maintaining system stability.

The remainder of this article is organized as follows. Section II proposes the SMC architecture and design. Section III introduces a control-theoretic framework for enabling a systematic design of both a safety controller and a stability-ensuring switching policy. Section IV presents the data-driven approaches to optimize control performance based on physical states and network reliability. Sections V and VI describe the case study and performance evaluation of SMC for industrial robotic control. Section VII reviews related works and Section VIII concludes this article.

II. SMC Architecture

In industrial control, the states of physical plants (e.g., the joint position of a robotic arm) are monitored and controlled by either a local controller or an edge controller. The edge controllers communicate with the sensors and actuators of the physical plants through communication networks. However, to date wireless networks and edge computing face open challenges for industrial control with stringent requirements: 1) control performance, which is related to factory revenue and 2) system stability, which is related to the safety of factory operation. The system may face data loss over wireless networks. For example, if a control command is lost, the actuator may not react to changes in the physical states in time. Therefore, our goal is to improve control performance and guarantee stability in the presence of data loss over the wireless network.

This section presents an overview of SMC architecture. The objective of SMC is to: 1) dynamically optimize control performance and 2) guarantee system stability when the reliability of the wireless network changes dynamically. SMC exploits the two-tier platform by switching between the local controller and the edge controller in response to changing network reliability.

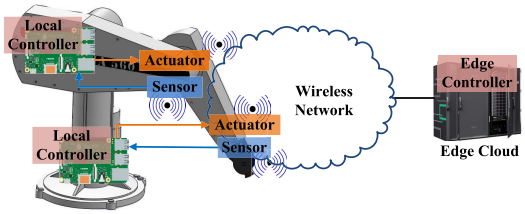

Fig. 2. Two-tier control system of robotic arms.

A wireless network incurs both communication latency and data loss. For digital control systems with periodic sampling, communication latency is acceptable as long as the packet meets its deadline (associated with the sampling period). When a packet misses its deadline, it is treated as a loss [11], [12]. In addition, industrial wireless network standards, such as ISA100 [13] and WirelessHART [14], employ TDMA protocols with predictable latency. In such systems, the impact of wireless networks on control is often manifested through packet losses. Therefore, we focus on addressing data loss in this article. Latency will be addressed in future work.

In this section, we introduce the SMC architecture and provide an overview of its switching logic and mechanisms. The design to guarantee system stability and optimize performance will be detailed in Sections III and IV, respectively.

A. System Model

SMC employs a pair of local/edge controllers for each feedback control loop. Compared to the local controller, the edge controller may run a more sophisticated control algorithm given the larger computational capacity of the edge server. The local controller is connected to sensors and actuators through a wired network with no data loss. The local controller and the edge controller communicate over wireless networks with varying data loss. An example of two-tier robotic control is shown in Fig. 2. At any time, only one controller is active and controls the physical plant at a sampling period T_s. Furthermore, a *switching agent* is co-located with each controller. The switching agent co-located with the active controller checks the trigger of stability status to determine whether to switch to the safety controller and checks the trigger for performance optimization every coordination period T_c to determine whether to switch to the controller with the optimal performance. If a switch is triggered, the control is transferred to the other controller along with the necessary state information using a switching protocol.

We now introduce the variables representing the physical and network states. $\boldsymbol{x}(k)$ denotes the state vector of the controlled system in the kth sampling period; and $\boldsymbol{x}_e(k)$ denotes the state error vector defined as the difference between $\boldsymbol{x}(k)$ and its reference value. In our case study, we use the mean absolute error MAE $:= (1/n+1)\sum_{k=0}^{n}|\boldsymbol{x}_e(k)|$, where n denotes the number of samples in the coordination period, as a suitable metric for control performance of a robotic system tracking a reference trajectory.

The control performance of the edge controller depends on the data loss over the wireless network. We explore two models

to describe the data loss process: 1) independent and identically distributed (i.i.d) Bernoulli random distribution with packet loss ratio ρ and maximum loss bound η defined as the maximum number of consecutive packet loss and 2) two-state Markov chain with the probabilities of transiting from/to the packet loss state to/from the packet reception state as α and β, respectively. While these models are relatively simple, we show in our evaluation that they are effective in selecting the controller when used with data-driven approaches.

B. Switching Logic

The switching logic of SMC integrates two switching rules: 1) the *Stability Switch* for guaranteeing stability and 2) the OPC for selecting the optimal control platform.

The Simplex framework [6] supports high-performance and dependable control systems through the integration of an unverifiable complex controller, a verified safety controller, and a switching logic. When the system is in danger of entering an unrecoverable state, the switching logic makes the system switch to the safety controller. In this way, a control system can employ the complex controller while maintaining the guarantees of the safety controller. In the Simplex framework, the physical states satisfying all operational constraints, such as physical state restrictions and limits of actuation, are defined as *admissible states*. The set of *recoverable states* is a subset of the admissible states. If the given *safety controller* is used from these states, all future states will remain admissible. The Stability Switch in SMC extends the Simplex framework to our two-tier architecture to guarantee stability.

SMC extends the Simplex framework [6]–[10] in three novel ways: 1) while earlier research on Simplex usually assumed that the safety controller and the performance controller are co-located on a same computational platform, SMC extends the approach to the distributed two-tier architecture comprising local and edge controllers; 2) we introduce OPC, a new data-driven classifier to select the optimal controller at run time; and 3) we integrate the Stability Switch and OPC in a coordinated switching logic that optimizes control performance without sacrificing stability.

Stability Switch: In SMC, the local controller serves as the safety controller as it usually employs a simple control law and has a reliable connection to sensors and actuators without data loss. The edge controller serves as the performance controller that can afford a sophisticated control law and suffers time-varying data loss over the wireless network. As illustrated in the left figure in Fig. 3, the physical system may operate in the recovery region (RR) or the performance region (PR) in terms of its physical states. RR is the region of recoverable states that can be stabilized by the local controller. PR is the subset of the RR, where either the local or the edge controller may be active as selected by the OPC at runtime. The derivation of PR is presented in Section III.

OPC: When the system operates in PR, the OPC is responsible for selecting between the local and edge controllers based on the network conditions and physical states. The control performance of the local and edge controllers depends on both the current network conditions and the physical states of the system. To overcome the theoretical challenges of analyzing control performance of complex systems under network dynamics [3]–[5], OPC employs data-driven approaches to select controllers. The data-driven approach allows SMC to be adopted in a wide range of practical systems and networks. The inputs to the OPC include the physical states (e.g., $\boldsymbol{x}_e$) and network conditions. The network conditions include the distribution and burstiness of data loss. e.g., ρ and η in the i.i.d. model, or α and β in the Markov chain model. The OPC may be trained based on simulation or experimental results. The design of OPC is detailed in Section IV.

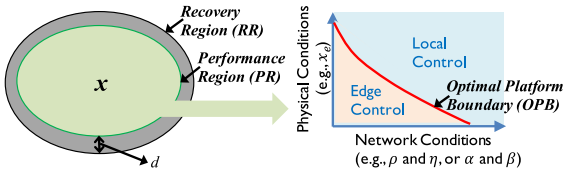

Fig. 3. Regions and boundaries of SMC.

Integration of Stability Switch and OPC: As illustrated in Fig. 3, when $\boldsymbol{x} \in$ PR, the OPC selects the controller that optimizes control performance based on network conditions and physical states. When $\boldsymbol{x} \notin$ PR, control is switched to the local controller to guarantee stability. By designing the local controller and PR with theoretical guarantees and by setting the stability switch with higher priority, SMC can dynamically optimize control performance without sacrificing stability.

C. Switching Protocol

We now present the switching protocol used by SMC to execute the switching between local and edge controllers across the two-tier platform.

In the two-tier architecture, each controller is co-located with a switching agent. Each switching agent integrates the Stability Switch and the OPC. The OPC monitors network and physical conditions every T_c to achieve the platform that optimizes predicted control performance over the next T_c. The monitored network conditions depend on which loss model the OPC is trained on. If the OPC is trained on an i.i.d loss model, it measures the packet loss rate (ρ) and the maximum number of consecutive packet loss (η) over the last time window of T_c. If the OPC is trained on the Markov chain model, it measures the probabilities of transiting from/to the bad state (packet drop) to/from the good state (packet reception), α and β, respectively, over the last time window of T_c.

At any time, only one controller is active along with its switching agent. The active controller operates its control policy in each sampling period T_s to generate the actuation commands. When multiple tasks share the local or edge platform, schedulability analysis can be performed for each platform to guarantee that all the tasks remain schedulable when controllers are activated. As shown in the finite state machine (Fig. 4), the switching protocol works as follows.

1) When the *local controller* is active, the switching agent is invoked every T_c, the prediction horizon of OPC. If a) $\boldsymbol{x} \in$ PR and b) the OPC selects the edge controller as the optimal platform over the next prediction horizon T_c,

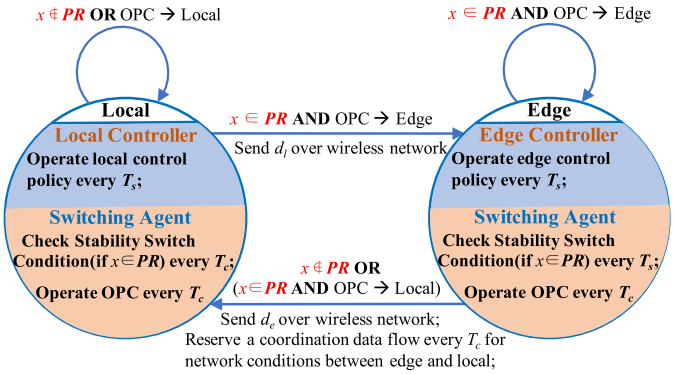

Fig. 4. Finite state machine for SMC. The arrows represent platform switches, and the switching conditions are above the arrows. The main actions taken to switch the computation platform are below the horizontal arrows.

the control switches to the edge controller. The switching agent on the local platform sends a switching command to the switching agent on the edge platform along with any state data, d_l, needed by the edge controller. e.g., depending on control design, d_l may include previous $\boldsymbol{x}$.

2) When the *edge controller* is active, the switching agent checks if $\boldsymbol{x} \in$ PR every T_s and switches to a local controller immediately when $\boldsymbol{x} \notin$ PR. The switching agent also invokes OPC every T_c. If the OPC selects the local controller as the optimal platform over the next T_c, the control switches to the local controller. The switching agent on the edge platform sends a switching command to the switching agent on the local platform along with any state data, d_e, needed by the local controller.

Given the criticality of the switching command, the switching agents rely on retransmissions and acknowledgments to ensure reliable communication. Furthermore, to ensure a higher level of assurance, the local controller may remain active to monitor the physical states and take over control when $\boldsymbol{x} \notin$ PR. The local controller may also monitor the network condition using the control commands from the edge controller as heartbeat messages. When it has not received an actuation command from the edge controller within a timeout threshold, the local controller can take over the control. This approach enhances the dependability of SMC as it can switch to the local controller without the switching commands from the edge switching agent, e.g., when the network between the edge and local is completely jammed. The actuators can be designed to allow override by the local controller if it receives control commands from both controllers.

III. Local Controller and Performance Region

The key problem of a Stability Switch boils down to developing the local controller and the PR. We consider the system dynamics approximated by a linear time-invariant (LTI) model with linear constraints since a wide variety of systems can be represented with satisfactory accuracy by such LTI model

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{1a}$$

$$a_i^T \boldsymbol{x} \le 1, \ \ i = 1, 2, \ldots, q \tag{1b}$$

$$b_j^T \boldsymbol{u} \le 1, \ \ j = 1, 2, \ldots, r \tag{1c}$$

where (1b) are physical restrictions and (1c) are limits of actuators. The local controller is a state feedback controller, $\boldsymbol{u} = K\boldsymbol{x}$. We next formulate a convex optimization problem to establish both the safety feedback gain, K, and RR

$$\mathbf{S} = \left\{\boldsymbol{x} | \boldsymbol{x}^T P \boldsymbol{x} \le 1\right\} \tag{2}$$

by applying the Lyapunov stability theory and linear matrix inequality (LMI) methodologies [7]–[10], where P is a positive-definite matrix. P defines a Lyapunov function, $\boldsymbol{x}^T P \boldsymbol{x}$, which is positive definite with a negative-definite derivative, thus guaranteeing the stability of the linear system.

We establish the local controller (K) and $\mathbf{S}$ ($Q = P^{-1}$) jointly by applying the method presented in [10]. We solve an optimization problem over feasible K and Q, subject to LMI constraints, such that the resulting closed-loop system is asymptotically stable and RR is maximized. Since the volume of RR given by (2) is proportional to $\sqrt{\det(Q^{-1})}$, maximizing RR is equivalent to minimizing the determinant $\det(Q^{-1})$. The LMI problem can be formulated as

$$\underset{Q,Z}{\text{minimize}} \ \ \log\det(Q^{-1}) \tag{3a}$$

$$\text{subject to} \ \ Q > 0 \tag{3b}$$

$$QA^T + AQ + Z^T B^T + BZ < 0 \tag{3c}$$

$$a_i^T Q a_i \le 1, i = 1, 2, \ldots, q \tag{3d}$$

$$\begin{bmatrix} 1 & b_j^T Z \\ Z^T b_j & Q \end{bmatrix} \ge 0, j = 1, 2, \ldots, r. \tag{3e}$$

Applying the change of variable, we obtain $K = ZQ^{-1}$ and $P = Q^{-1}$. Constraints (3b) and (3c) are the stability (Lyapunov equation) constraints, and (3e) is an LMI constraint converted from (1c). Since multitier control systems are implemented on digital platforms, we discretize (1a) with sampling period T_s. We consider the worst case latency of T_s, since in our case (Table II, joint position control) the worst case end-to-end (E2E) latency of local safety controller is 6.19 ms which is shorter than $T_s = 50$ ms, which results in the consideration of

$$\boldsymbol{x}(k+1) = A_d \boldsymbol{x}(k) + B_d \boldsymbol{u}(k-1). \tag{4}$$

Considering the state feedback control $\boldsymbol{u}(k) = K\boldsymbol{x}(k)$ and defining a new state vector $\boldsymbol{z}(k) = \left[\boldsymbol{x}^T(k) \ \boldsymbol{x}^T(k-1)\right]^T$, the closed-loop form of the augmented system of $\boldsymbol{z}(k)$ is

$$\boldsymbol{z}(k+1) = \bar{A}_d \boldsymbol{z}(k) \tag{5}$$

where

$$\bar{A}_d = \begin{bmatrix} A_d & B_d K \\ I_{N\times N} & 0_{N\times N} \end{bmatrix}. \tag{6}$$

We solve $\bar{Q} = \bar{P}^{-1}$ by [7] as

$$\underset{\bar{Q}}{\text{maximize}} \ \ \log\det(\bar{Q}) \tag{7a}$$

$$\text{subject to} \ \ \bar{Q} > 0 \tag{7b}$$

$$\bar{Q}\bar{A}_d^T - \bar{A}_d^{-1}\bar{Q} < 0 \tag{7c}$$

$$\bar{\alpha}_m^T \bar{Q} \bar{\alpha}_m \le 1, m = 1, 2, \ldots, g \tag{7d}$$

where $\bar{\alpha}_m = [\alpha_m^T, 0_{1\times N}]^T$ and $\bar{\alpha}_m = [0_{1\times N}, \alpha_m^T]^T$ are the constraints corresponding to $\boldsymbol{x}(k)$ and $\boldsymbol{x}(k-1)$, respectively.

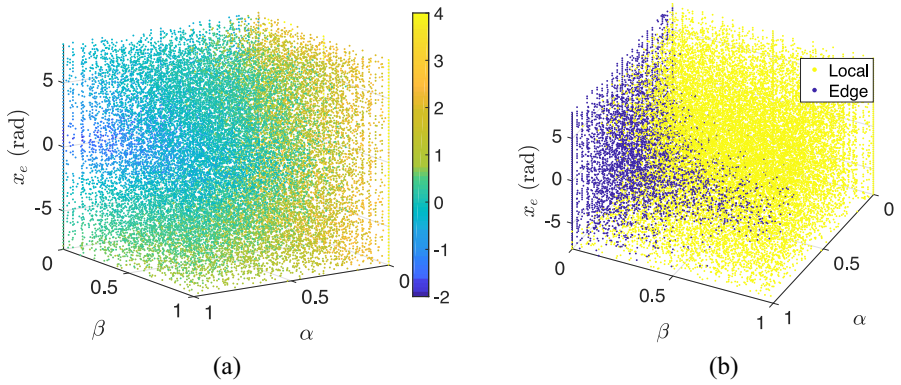

Fig. 5. Data collection, labeling, and classifying (Markov loss when $T_c = 15$ s, the color bar of (a) indicates log-scale MAE), (a) data collected for edge control, (b) optimal platform labeling.

$\alpha_m^T = a_m^T$, for $m = 1, \ldots, q$; $\alpha_m^T = b_j^T K$, for $m = q+1, \ldots, g$, $j = 1, \ldots, r$, and $g = q + r$. The corresponding RR is

$$\mathbf{S} = \left\{ z(k) | z(k)^T \bar{P} z(k) < 1 \right\}. \tag{8}$$

Remark 1: The stability condition of the two-tier control system is that $\boldsymbol{x}$ should stay in RR. Thus, control should be switched to the local controller before $\boldsymbol{x}$ leaves RR. SMC is designed to switch among two platforms in a distributed way. Hence, the switching latency is unavoidable. It may result in $\boldsymbol{x}$ out of RR. In order to guarantee stability, a smaller region of PR should be calculated. A practical solution would be to choose a smaller ellipsoid, e.g., an ellipsoid defined by $\boldsymbol{x}^T P \boldsymbol{x} = 0.7$ [6]. The distance d in Fig. 3 can be derived theoretically through the searching of control policies that drive $\boldsymbol{x}$ within RR to the boundary of RR in the shortest time, which can be formulated as a minimum-time optimal control problem and solved by the bang-bang control. If the resulting shortest time is shorter than the sum of T_s and switching latency, $\boldsymbol{x}$ belongs to PR.

Remark 2: When switching occurs, the actuation commands may change in a nonsmooth (discrete) fashion. The discrete changes in the actuation commands will not affect stability since the stability switch is rigorously derived for the system (4), and switches do not happen frequently (multiples of T_c). In addition, there exist switching system designs that deal with discontinuous design space [15], which are not the focus of this article.

IV. Optimal Platform Classifier

OPC employs data-driven approaches to select controllers trained by simulations. Invoked every T_c, OPC measures the network conditions and physical states and selects the controller that is predicted to optimize control performance over the prediction horizon T_c. Our data-driven approach is inspired by simulation-guided certificate construction [16], [17], but we apply the data-driven approaches for different purposes and scenarios. Unlike certificate construction, the false positive of which is extremely dangerous to a control system, the OPC is designed to improve the control performance within the stabilizable PR, the misclassification of which does not affect system stability in SMC.

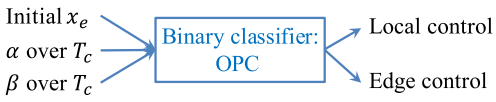

Fig. 6. OPC (with Markov data loss).

A. Cyber–Physical Inputs

There exist different approaches to model data loss in the context of networked control systems (NCSs): 1) modeling loss as stochastic processes [18], e.g., i.i.d Bernoulli random distribution [19], [20] and two-state Markov chain [21]; 2) modeling loss as weakly hard real-time constraints [22], e.g., (m, k) model [11], [23]; and 3) modeling loss with consecutive loss bound [24], [25]. The (m, k) model is commonly used in real-time scheduling for computing tasks and is not suitable for the probabilistic nature of wireless communication. Hence, we explore two stochastic models for data loss, including i.i.d process with consecutive loss bounds and two-state Markov chain. The i.i.d loss model is characterized by: 1) the loss ratio ρ and 2) the maximum number of consecutive packet loss count η. The two-state Markov chain loss (Gilbert-Elliott) model is characterized by: 1) the probability of transiting from the packet loss state to the reception state α and 2) the probability of transiting from the reception state to the loss state β. Our experiments show that the OPC trained by either model works for realistic wireless traces. In addition to the network conditions, OPC also takes the physical state errors, $\boldsymbol{x}_e$, as inputs. OPC, therefore, selects controllers based on both cyber (network) and physical states.

B. Data Collection

To train and test OPC, we conduct simulations of local and edge control by sampling the input values over their feasible ranges. Since that OPC is invoked every T_c at runtime, the simulation time for data collection is T_c such that SMC can optimize the control performance over the prediction horizon of T_c. We will study and discuss the choice of T_c in Section VI. Our simulations are performed for a robotic case study in WCPS-EC, a real-time hybrid simulator (see Sections V-C and VI for details).

We run simulations to collect data for i.i.d loss model and two-state Markov chain loss model, respectively. The

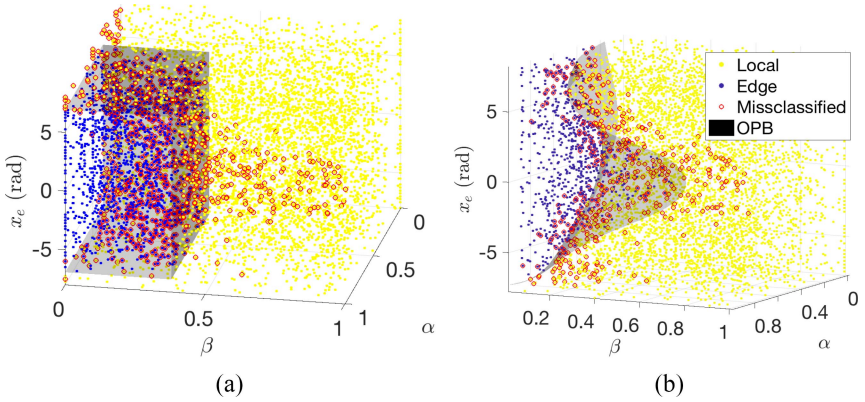

Fig. 7. OPBs derived by threshold-based classifier and SVM, and testing results (Markov loss, $T_c = 15$ s), (a) threshold-based (Grid) classifier, (b) SVM.

simulations incorporate communication latency corresponding to the maximum latency of the local and edge platforms measured experimentally (see Section VI-A). Fig. 5(a) shows data collection and processing examples for cases of two-state Markov chain loss model when $T_c = 15$ s (other experimental settings of this group of experiments are the same as in Section VI-A2). The three axes are OPC inputs, i.e., α, β, and initial $\boldsymbol{x}_e$. Each data point indicates one round of simulation for 15 s. The color bar indicates the log-scale MAE. We can see that when $\boldsymbol{x}_e$ and β are low, and α is high, the edge control has smaller MAE. By comparing MAEs of edge control and local control, we label each data point with the optimal platform that achieves smaller MAE, as shown in Fig. 5(b).

C. Classifier: Training and Testing

We then train classifiers to select the local or edge platform (as shown in Fig. 6). We explore two classification models, threshold-based classification and support vector machines (SVMs). The goal of a classifier is to identify the optimal platform boundary (OPB), i.e., the boundary that separates the regions in which the local and edge platforms achieve better control performance, respectively. While simple to train and efficient at runtime, a threshold-based classifier may not be able to achieve a close approximation of the OPB that is often nonlinear in practical systems. In comparison, SVM is a well established and powerful method to efficiently establish the separation boundary of arbitrary data sets, even when the training data are not linearly separable [26]. We train and compare both classifiers in this article.

For the threshold-based classifier, we use grid search to find the set of thresholds that minimize the classification error on the training datasets. For example, for the two-state Markov chain loss model, we exhaustively sweep all feasible values of α, β, and $\boldsymbol{x}_e$ at certain step sizes to find the set of parameters that leads to the minimum classification error. We tuned the step sizes to balance the search time and classification error. The step sizes used to generate the thresholds are 0.067 for α and β, and 0.53 for $\boldsymbol{x}_e$. For SVM, we use the radial basis kernel function, proper box constraint, and kernel scale to minimize the tenfold cross-validation loss on the training datasets.

We trained and tested the classifiers on our training and testing datasets, respectively. The OPBs established by the threshold-based (Grid) classifier and SVM are shown as the two shaded surfaces in Fig. 7, respectively. The regions to the left and right of the boundaries are classified as edge and local control, respectively. As expected, when $\boldsymbol{x}_e$ and β are low, and α is high, OPC will choose edge control, and local control otherwise. Notably, SVM establishes a nonlinear boundary separating edge control and local control while the threshold-based classifier identifies a cube-shaped boundary.

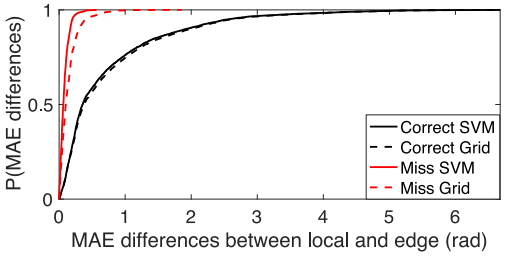

Fig. 8. CDF of MAE difference for local and edge control in the correct classification sets and miss classification sets, respectively.

The testing results of SVM are shown in Fig. 7 (misclassified points are circled in red). We observe that the miss classifications happen near OPB, where local and edge controllers have similar control performance. Fig. 8 shows the empirical cumulative distributions of MAE difference of local and edge control in their correct classification sets and in their miss classification sets, respectively. The results confirm that misclassifications have moderate impacts on control performance for both classifiers. The SVM outperforms threshold-based (Grid) classification in the miss classification set since SVM establishes a more accurate boundary. We train two sets of OPCs based on the i.i.d loss model and two-state Markov chain loss model to show the generality of the data-driven approach. Both OPCs work for realistic wireless traces, which will be discussed in Section VI-B2.

Remark 3: Due to the generality of the data-driven approach, the OPC can be trained for other metrics of control performance. For example, to optimize the settling time of a control system, the OPC can be trained to select the optimal control platform with the shortest settling time. Furthermore, to minimize settling time, the local and edge controllers can be designed to maximize the convergence speed with proper closed-loop eigenvalues [27] or learning-based

approaches [28]. For our robotic case study, as we consider a linear state-space model for the plant, we can design the stabilizing controller to result in a certain convergence rate of the closed-loop system, which is readily given in terms of the eigenvalues of the closed-loop system matrix.

V. Case Study Design for Multitier Control

Fig. 2 shows a case for multitier control, where single or multiple robotic arms are in a workshop of a factory. We will utilize this case study to explore the multitier control system in the rest of this article. The objectives are to control the velocity and position of the independent joints.

In order to evaluate multitier control and SMC, we built WCPS-EC, a real-time hybrid simulator that integrates real multitier computation platforms, real or simulated wire/wireless networks, and physical plants simulated in Simulink.

A. Physical Plants

A robotic arm comprises a chain of joints and rigid links. The motion of the end effector is the composition of the motion of each link, and the links are ultimately moved by forces and torques exerted on the joints.

The most common structure for joint control is the nested control loop. The outer loop is responsible for maintaining position and determines the velocity of the joint that will minimize position error. The inner loop is responsible for maintaining the velocity of the joint as demanded by the outer loop. The motor drive assembly comprises a motor to generate torque, a gearbox to amplify the torque and reduce the effects of the load, and an encoder to provide feedback about position and velocity. We can write the torque balance on the motor shaft as

$$K_m K_a u - B'\omega - \tau_C'(\omega) - \frac{\tau_d(q)}{G} = J'\dot{\omega} \tag{9}$$

where K_m is the motor torque constant, K_a is the transconductance of the amplifier, q is the joint coordinates, w is the joint velocity, and u is the control voltage. B', τ_C', and J' are, respectively, the effective total viscous friction, Coulomb friction, and inertia due to the motor, gearbox, bearings, and the load. $\tau_d(q)$ is the disturbance torque due to the link motion. In order to analyze the dynamics of (9), we linearize it by setting all additive constants to zero

$$J'\dot{\omega} + B'\omega = K_m K_a u \tag{10}$$

which admits the transfer function description

$$\frac{\Omega(s)}{U(s)} = \frac{K_m K_a}{J's + B'}. \tag{11}$$

The outer position loop provides the velocity demand for the inner velocity loop.

We modify the open-source Robotics ToolBox [29] to simulate a PUMA560 robotic arm in a multirate and real-time fashion. We refer readers to [29] for the underlying kinematics and dynamics of the robotic arm and the guide for the toolbox, and to [30] and [31] for the parameters of PUMA560.

TABLE I
System Settings of the Case Studies

Tiers	Local	Edge
Computation	Raspberry Pi 3 (4 ARMv7 CPUs@ 900 MHz, 1G RAM)	Intel Server (4 Intel Core i5-4590 CPUs@ 3.3 GHz, 16 G RAM)
Communication	I/O + Ethernet cable	I/O + Wi-Fi + University network

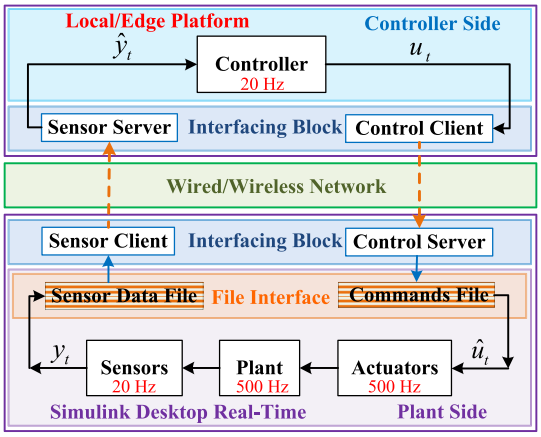

Fig. 9. Architecture of WCPS-EC.

B. System Settings of Local and Edge Platforms

The architecture of our case studies for exploring the multitier control system was shown in Fig. 2. The settings of the computation platforms and communication approaches that we use are described in Table I.

C. Wireless Cyber–Physical Simulator-Edge Computing

It is challenging to conduct experiments on industrial control systems in the field, especially under cyber and physical disturbances. We built a real-time hybrid simulator, WCPS-EC, which integrates: 1) real controllers running on various computation platforms; 2) real Wi-Fi network and Ethernet, or simulated network using TOSSIM; and 3) simulink desktop real time (SLDRT), which simulates the robotic arm in real time.

The architecture of WCPS-EC is shown in Fig. 9. Compared with RT-WCPS [32], WCPS-EC includes immigrated controllers running on various computation platforms instead of controllers running in MATLAB/Simulink. In addition, WCPS-EC can reflect the impacts of real communication network and computation platform during runtime.

We implement the control policies in Python and C [33] so that the policies can run on any platforms that support Python or C. Based on our measurements, while the controller written in C has a slightly shorter execution time than the Python implementation, the difference is not significant as the computation time is dominated by solving the same quadratic programming problem. In RT-WCPS, the worst case E2E latency is below T_s. Hence, the actuation commands are set to have *fixed* latency of T_s. However, in multitier control systems, the E2E latency can be longer than T_s. In addition, we consider a more realistic control system setup [4], [8], [34]

TABLE II
LATENCY MEASUREMENTS OF EACH TIER (IN THE FORMAT OF MEDIAN AND WORST CASE LATENCY PAIR)

Tiers	Control policy	Communication latency (ms)	Computation latency (ms)	E2E latency (ms)
Local	PID	(3.89, 5.28)	(2.00, 2.79)	(5.91, 6.19)
	MPC	———	(51.49, 61.65) (infeasible)	———
Edge	PID	(22.17, 34.40)	(0.04, 0.05)	(23.44, 35.26)
	MPC	(20.80, 34.80)	(3.69, 4.44)	(25.29, 38.08)

with: 1) clock-driven sensors that sample the plant outputs periodically every T_s; 2) an event-driven controller which calculates the actuation commands as soon as the sensor data arrives; and 3) event-driven actuators, which means actuators can respond to updated actuation commands immediately.

As shown in Fig. 9, *Sensors* sample periodically (e.g., 20 Hz), and write the new measurements to a *Sensor Data File*. The *Sensor Client*, located on the host of SLDRT, discovers the sensing update immediately and sends it to the *Sensor Server* via a wired or wireless networks. The event-driven *Controller* starts operation once *Sensor Server* receives the sensing measurements, and sends the control commands to *Control Server*, which writes the control commands to the *Commands File*. The actuator in SLDRT checks the *Commands File* with high frequency (e.g., 500 Hz) and updates the actuation once it discovers an update of control commands. The intermediate *File Interface* is introduced between Simulink and the computation platform. In this way, Simulink and the computation platform are asynchronous, which is essential for the sequential execution of the Simulink loop.

VI. CASE STUDY EVALUATION

We evaluate the multitier control and SMC in the case study introduced in Section V. We explore control systems with static local and edge controllers. Two cases, i.e., multitier control of joint velocity and of joint position, are evaluated from the aspect of control performance. Next, we evaluate the SMC that optimizes performance and guarantees system stability.

A. Evaluation of Static Cases

When the network is in normal condition, Table II summarizes the latencies of different tiers of control with the system settings described in Table I. We choose proportional-integral-derivative (PID) and model predictive control (MPC) as alternative control policies for independent joint velocity and position control. As shown in Table II, since PID is more computationally lightweight than MPC, the computational latency of PID is shorter than that of MPC over both local and edge platforms. Since the edge platform is equipped with more computation capacity than the local platform, it finishes PID and MPC within 0.05 and 4.5 ms, respectively, while the local platform requires 2.8 and 61.7 ms, respectively. On the other hand, communication latency between the local

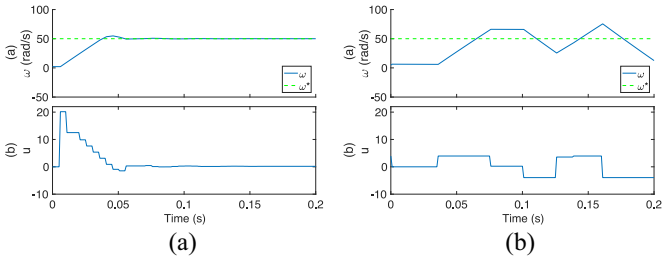

Fig. 10. Response curves of the joint velocity control. (a) PID over local. (b) MPC over edge.

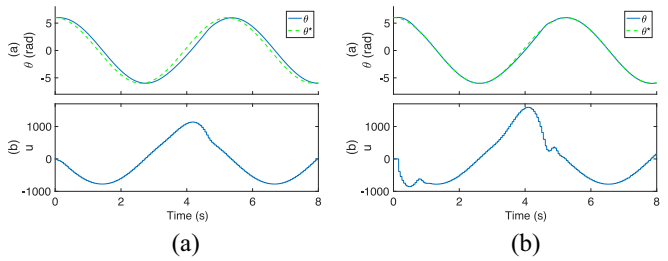

Fig. 11. Response curves of the joint position control. (a) PID over local. (b) MPC over edge.

platform and the plant is around 5 ms, which is much shorter than $\sim$ 30 ms of the edge platform. In summary, edge can run MPC much faster than local due to higher computation capacity at the cost of longer communication latency.

1) Case 1—Joint Velocity Control of PUMA560: The physical plants described by (10) are discretized and simulated at 1000 Hz to mimic a continuous system. The sampling rate of the sensors is 200 Hz, a reasonable rate for the system time constant of joint velocity control. The settings for communication between sensors/actuators and the computation platform are shown in Table I. The communication latency of the case studies is shown in Table II.

The response curves of the joint velocity control are shown in Fig. 10. The reference value of the physical state w is w^*. For joint velocity control that has a short time constant (high sensitivity to latency), the local controller, which has the shortest latency, performs the best. We ran experiments for 30 times. The statistical results for MAE are shown in Fig. 12(a). PID over local has the best control performance, which is consistent with the observation in Fig. 10.

2) Case 2—Joint Position Control of PUMA560: The settings for the joint position control are mostly the same as for the joint velocity control described in Section VI-A1, except that the sampling rate is 20 Hz since the time constant is much longer than that of velocity control. We control the joint position (θ) to track a sinewave signal (θ^*), which is common in position control of robots [35]. The response curves and the statistical results of the joint position control are shown in Figs. 11 and 12(b), respectively. The edge controller performs the best since the gain of MPC on edge overcomes the effects of extra communication latency.

In summary, for joint velocity control, PID over local has the best performance, since it is sensitive to latency. For joint position control, MPC over the edge has the best performance since MPC performs better than PID, which overcomes the

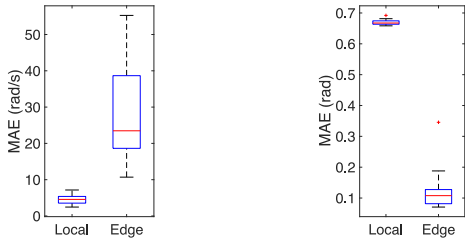

Fig. 12. Performance of the joint velocity and position control. (a) Joint velocity control. (b) Joint position control.

effects of longer communication latency. The control performances of control tiers are determined by the properties of control policies, networks, and physical plants. The advantage of the two-tier computing architecture is enabled by the flexibility in controller placements and the choice of corresponding control policies tailored for physical plants.

Note that the MPC controller over local is infeasible due to the computational resource shortage on edge: the computation latency is longer than the sampling period. For instance, in Table II, the maximum computational latency of MPC over local is 61.7 ms, which makes it infeasible for a sampling period of 50 ms. In this section, the network conditions are normal. The performance of the closed-loop control system may deteriorate due to the unreliable wireless network connection between the edge controller and the plant [4]. In the next section, we will focus on evaluating SMC facing changing data loss.

B. Evaluation of SMC

1) OPC Training and Testing: We run 26 000 simulations to collect around 40-GB data for i.i.d loss model and Markov chain loss model, respectively. We set the worst case end-to-end latency of local control and edge control to 7 and 40 ms, respectively, based on the measurements in Table. II. For each data loss model (i.i.d or Markov), we randomly choose 6500 simulations as the training set, and the other 6500 as a testing set. The accuracies of training and testing are summarized in Table III. SVM consistently achieved higher accuracies than the threshold-based (Grid) classification. The average time taken by one SVM classification is 0.67 and 10.27 ms on the edge and local platforms (over 20 000 testing cases), respectively, which suggests that online classification is practical even on a local platform. The average time it takes to train the SVM classifier is 26.62 s on the edge server (over 100 training runs). It is a reasonable amount of time since training is done offline.

2) Performance Optimization: In order to evaluate SMC, we consider joint position control with the same settings as described in Section VI-A2, except that we simulate data loss based on traces generated by TOSSIM [36], [37], which is a high-fidelity wireless simulator. Received signal strength indicator (RSSI) data have been collected over real wireless testbed. In addition, we use controlled background noise strength to simulate various network conditions between edge and local as shown in Fig. 13(a). Both the RSSI and controlled noise strength are fed into TOSSIM to derive realistic packet

TABLE III
ACCURACY OF THRESHOLD-BASED (GRID) AND SVM CLASSIFIERS. (THE TRAINING ACCURACIES OF GRID AND SVM ARE FROM THE GRID SEARCH AND TENFOLD CROSS-VALIDATION, RESPECTIVELY, ON THE TRAINING DATASETS. THE TESTING ACCURACIES ARE MEASURED ON THE TESTING DATASETS)

Loss Model Approach	Accuracy	$T_c = 5$ s	$T_c = 10$ s	$T_c = 15$ s	$T_c = 20$ s
i.i.d Grid	Training	95.46%	89.60%	88.02%	89.68%
	Testing	94.80%	89.68%	87.74%	89.71%
i.i.d SVM	Training	96.95%	91.89%	91.26%	92.62%
	Testing	96.83%	91.72%	90.25%	92.48%
Markov Grid	Training	96.02%	90.58%	86.91%	85.48%
	Testing	95.31%	90.08%	86.68%	85.69%
Markov SVM	Training	96.82%	92.17%	91.72%	91.63%
	Testing	96.94%	93.11%	90.98%	91.78%

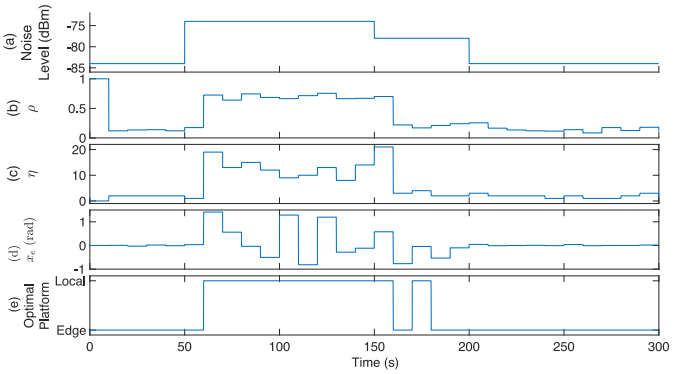

Fig. 13. System dynamics under SMC. ($T_c = 10$ s, i.i.d data loss).

loss traces. The OPC (SVM trained for i.i.d loss model and $T_c = 10$ s) monitors ρ and η during a sliding window of the past T_c (10 s), as shown in Fig. 13(b) and (c). The runtime position error inputs of OPC are shown in Fig. 13(d). The optimal platform derived by OPC is shown in Fig. 13(e). We can see that edge control is the best choice when the noise level is low, and the control should be switched to local when the noise level is high since the gain of the edge control is offset by the effects of data loss.

With a coordination period T_c of 10 s, we observe a 10-s delay of OPC in reacting to the changes of the noise level. Fig. 14(a) compares the control performance of SMC with various T_c, fixed local control, and fixed edge control over 30 rounds of simulations. When T_c is properly chosen, i.e., 10–15 s, our SMC provides over 30% and 40% MAE reduction over fixed local control and edge control, respectively. While the OPC is trained based on the i.i.d loss model, it works for more realistic loss traces generated by TOSSIM as well.

To study the impact of T_c, we run experiments under more frequently changing noise [Fig. 14(b)]. As shown in Fig. 14(c), SMC achieves the best performance with $T_c = 10$ s. When T_c is too short, the OPC is trained based on the data of transient physical states without taking steady states into consideration. On the other hand, when T_c is too long, the OPC cannot react to frequently changing network conditions in time. We will explore adaptive T_c to balance the above factors in the future.

In Fig. 15, we compare the performance of the OPC trained based on the i.i.d loss model and that based on the Markov

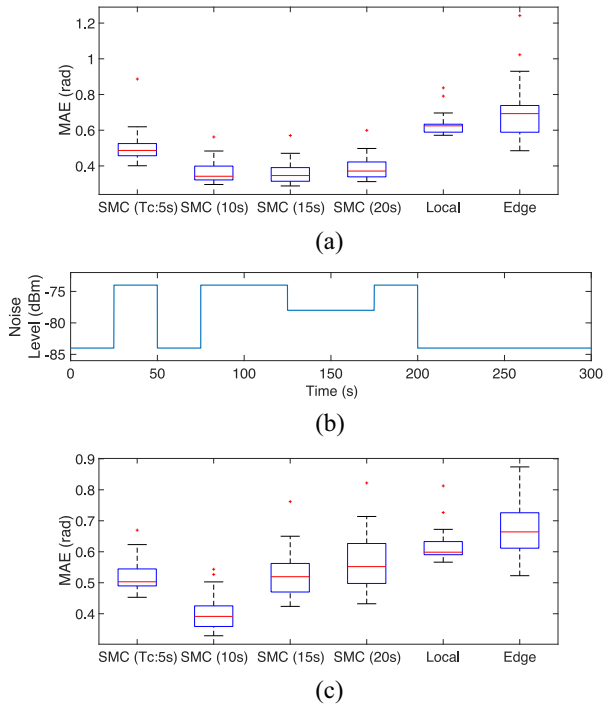

Fig. 14. Control performance under SMC, fixed local control, and fixed edge control. (a) Under changing noise as in Fig. 13(a). (b) Frequently changing noise. (c) Under frequently changing noise as in (b).

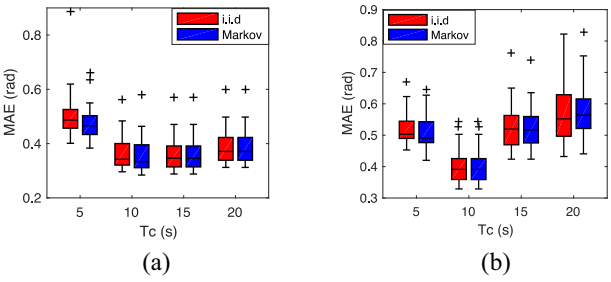

Fig. 15. Performance comparison of OPC based on the i.i.d loss model and the Markov loss model. (a) Changing noise as in Fig. 13(a). (b) Changing noise as in Fig. 14(b).

chain loss model. The OPC performs similarly when trained with different data loss models, which suggests the generality and robustness of the data-driven approach for selecting optimal platforms.

In Fig. 16, we compare the control performance under learning-based (SVM) and threshold-based (Grid) OPC. Thanks to its higher classification accuracy, SVM generally outperforms Grid. Furthermore, SVM is a more general and precise approach that may lead to more significant improvement when the cost of misclassification is higher under other settings, which we will evaluate in future work.

3) Stability Analysis: We model the joint position control by treating the inner velocity loop as constant since the response of the velocity loop is much faster than that of the position loop, as shown in Figs. 10 and 11

$$\dot{\theta}(t) = -K_g K_p \theta(t). \tag{12}$$

We define $\tilde{\theta}(t) = \theta(t) - \theta^*(t)$, and the proportional controller $u(t) = -K_p\tilde{\theta}(t)$. To achieve the equilibrium point at the origin,

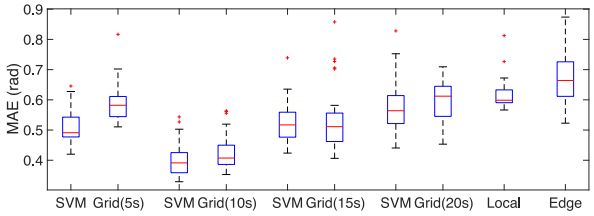

Fig. 16. Performance comparison of learning-based (SVM) and threshold-based (Grid) OPC [with Markov loss model, under noise as in Fig. 14(b)].

we transform the coordinates of (12)

$$\dot{\tilde{\theta}}(t) = -K_g K_p \tilde{\theta}(t) = K_g u(t).$$

Solving (3), among multiple solutions, we choose $K_p = 800$.

Then, we model the discrete system with one sampling period latency (T_s) using the Euler method

$$z(k+1) = \begin{bmatrix} 1 & -T_s K_g K_p \\ 1 & 0 \end{bmatrix} z(k), \text{ where } z(k) = \begin{bmatrix} \tilde{\theta}(k) \\ \tilde{\theta}(k-1) \end{bmatrix}.$$

We solve (7) by using YALMIP and the multiparametric toolbox (MPT) with semidefinite programming solver (SDPT3) solver in MATLAB

$$\Rightarrow \bar{Q} = \begin{bmatrix} 2.837 & 3.382 \\ 3.382 & 7.834 \end{bmatrix} \Rightarrow \bar{P} = \begin{bmatrix} 0.732 & -0.316 \\ -0.316 & 0.264 \end{bmatrix}.$$

The corresponding RR is

$$\mathbf{S} = \left\{ z(k) | z(k)^T \bar{P} z(k) < 1 \right\}.$$

We confirm that during the dynamic switch process, all physical states are within this RR. Therefore, stability is guaranteed, and no Stability Switch is needed.

VII. Related Work

The cloud computation platforms proposed and commercialized in last two decades for large-scale data centers are famous for their high computing ability, voluminous data storage, and low cost [38]–[40]. The cloud-based IIOT provides a centralized solution for statistical data analysis of increased amounts of tasks and data. Many mainstream cloud computing vendors, such as Amazon, Google, and IBM, have offered services in various domains, such as data management and analysis, and intelligent transportation systems. However, for industrial control systems, a cloud-based computation platform suffers from long and fluctuating latency to end devices because it is up to hundreds of miles away from the end devices [41].

In the recent decade, the increase of IoT devices at the edge of the network and the evolution of hardware with increased computational capability have spawned the edge computation platform. Compared with the cloud-based platform, the edge platform benefits from the localization of computation resources [42]. The ability to provide local data aggregation and processing not only reduces the bandwidth demand on network links to the cloud [43] but also meets the latency requirements of applications [44]. Therefore, although only in its early stage, edge computing is already widely deployed among applications with low-latency requirements, such as vehicles [45], mobile gaming [46], and health monitoring [47]; applications dealing with large amounts of data, such as

video [48] and large-scale sensor networks [49]; and applications with geographical distribution [50] and mobility [51], such as mining, smart grid, transportation, waste management, and agriculture [52]. However, few edge computing technologies have been applied to industrial control systems thus far.

In a conventional industrial control system, the control algorithm runs on a local controller, which is usually deployed physically near the sensors and actuators and implemented with an SCM or PLC [53] with a reliable wired connection but limited computational ability. In recent years, the physical plants tend to be controlled by a remote controller for the advantages of: 1) more conductive environments for hardware implementation of control algorithms and 2) centralized control and coordination of multiple plants. Due to the network connection between the remote controller and the physical plant, remote control systems are regarded as NCSs [1], [2].

We note that there are remote controller designs that guarantee stability in a mean square sense in the presence of data loss. The design is based on stochastic Lyapunov functions [54], [55], as well as certain assumptions about the communication network. However, the unpredictable wireless conditions mean that the assumptions may not be guaranteed, leading to unsafe physical plant operations with deteriorated performance.

Lin and Antsaklis [56] and Liberzon and Morse [57] summarized switching control approaches that maintain stability. Dai *et al.* [58] established an optimization framework for switching sampling periods that maximizes the control performance and CPU resource efficiency. However, those work do not consider control over wireless networks and multitier computing platforms. The Simplex framework ensures the safe use of an unverifiable complex controller by using a verified safety controller and a switching logic. In the conventional Simplex framework [6], the complex controller and safety controller operate in parallel. ORTEGA [7] enhances the efficiency and flexibility of Simplex by eliminating the redundant execution of controllers. NetSimplex [8] extends the Simplex from a single node control system to an NCS. Bak *et al.* [9] derived the switching logic by combining offline LMI results and online reachability computation, which significantly reduces conservatism. In all the above cases, both the complex controller and safety controller are running on the same computation platform. Whereas most previous works consider single machine cases, we extend the Simplex framework by considering: 1) distributed two-tier architecture comprising local and edge controllers; 2) new data-driven OPC for control performance optimization; and 3) a coordinated switching logic which integrates stability switch and OPC.

Ma *et al.* [5] designed the stability switch between local and edge controllers under data loss from another perspective, based on the co-design of edge and local controllers that are designed via a joint Lyapunov function. The co-design approach is nontrivial and specific to the control policy. In comparison, by extending the Simplex framework to the distributed two-tier architecture, our SMC approach can be applied to any control policies on edge without sacrificing stability.

Data-driven approaches have been applied to control design [59] and safety verification of control systems [17] to overcome the restrictions of analytical modeling. We develop our data-driven approach for a different purpose and system architecture. The data-driven OPC is designed to improve the control performance by selecting the optimal control platform in a distributed two-tier computing architecture.

Skarin *et al.* [60] presented a 5G-based edge cloud computing testbed for a multitier control system and evaluated the system while operating a mission-critical control application of an MPC controlled ball and beam process. However, systematic studies, such as how to co-design proper computation platforms and control policies, and how to adjust to uncertainties and maintain closed-loop stability, remain to be performed. Ma *et al.* [32], [61] proposed the concept of holistic control that co-joins network management and physical control at run time, where the focus is the remote control. In this article, we focus on a multitier industrial control system with a comprehensive view of the properties of the edge/cloud controller, network conditions, and physical plants. We also propose a controller dynamic switch among multitier computation platforms which not only optimizes the control performance at runtime but also guarantees system stability under various network conditions.

Simulation tools are of vital importance to study multitier control. NCS simulators [36], [62], [63] are MATLAB/Simulink-based tools, which enable simulations of CPU scheduling, communication, and control by integrating wireless simulators. Given simulators cannot always capture the real-world wireless network dynamics, network-in-the-loop simulations have recently been developed [32], [64]. Skarin *et al.* [60] presented a 5G-based edge cloud computing testbed for a multitier control system. However, the real physical plants used in its experiments are limited to lab settings. We build WCPS-EC, which integrates real multitier computation platforms and wireless networks and leverages simulation support for various physical plants.

VIII. Conclusion

With the emergence of edge computing and wireless network technologies, controllers located in various computational tiers have different computational capacities and communication reliability, which influence the performance and stability of industrial control systems. We presented an SMC architecture for industrial control systems. SMC dynamically switches between local and edge control based on plant states and network reliability. It employs a switching agent that integrates: 1) a data-driven optimal performance classifier for selecting the controller platform with optimal performance and 2) a Stability Switch for guaranteeing system stability. SMC effectively reaps the benefits of switching among different computation tiers. In hybrid simulations on WCPS-EC, SMC achieved over 30% and 40% reductions in mean absolute errors when compared with fixed local and edge control, respectively, while maintaining stability guarantees under changing network reliability.

REFERENCES

[1] C. Lu *et al.*, "Real-time wireless sensor-actuator networks for industrial cyber-physical systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1013–1024, May 2016.

[2] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 978–1013, 2nd Quart., 2018.

[3] K. Gatsis, H. Hassani, and G. J. Pappas, "Latency-reliability tradeoffs for state estimation," 2018. [Online]. Available: arXiv:1810.11831.

[4] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Syst.*, vol. 21, no. 1, pp. 84–99, Feb. 2001.

[5] Y. Ma *et al.*, "A smart actuation architecture for wireless networked control systems," in *Proc. IEEE Conf. Decis. Control*, Miami Beach, FL, USA, 2018, pp. 1231–1237.

[6] L. Sha, "Using simplicity to control complexity," *IEEE Softw.*, vol. 18, no. 4, pp. 20–28, Jul./Aug. 2001.

[7] X. Liu *et al.*, "ORTEGA: An efficient and flexible online fault tolerance architecture for real-time control systems," *IEEE Trans. Ind. Informat.*, vol. 4, no. 4, pp. 213–224, Nov. 2008.

[8] J. Yao, X. Liu, G. Zhu, and L. Sha, "NetSimplex: Controller fault tolerance architecture in networked control systems," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 346–356, Feb. 2013.

[9] S. Bak, T. T. Johnson, M. Caccamo, and L. Sha, "Real-time reachability for verified simplex design," in *Proc. IEEE Real Time Syst. Symp.*, Rome, Italy, 2014, pp. 138–148.

[10] D. Seto and L. Sha, "An engineering method for safety region development," Dept. Softw. Eng. Inst., Carnegie-Mellon Univ., Pittsburgh PA, USA, Rep. CMU/SEI-99-TR-018, 1999.

[11] D. Soudbakhsh, L. T. Phan, O. Sokolsky, I. Lee, and A. Annaswamy, "Co-design of control and platform with dropped signals," in *Proc. ACM/IEEE Int. Conf. Cyber Phys. Syst.*, Philadelphia, PA, USA, 2013, pp. 129–140.

[12] B. Demirel, Z. Zou, P. Soldati, and M. Johansson, "Modular design of jointly optimal controllers and forwarding policies for wireless control," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3252–3265, Dec. 2014.

[13] *Wireless Systems for Automation*, ISA100, Research Triangle Park, NC, USA, 2020. [Online]. Available: http://www.isa100wci.org

[14] *WirelessHART Specification*, Fieldcomm, Austin, TX, USA, 2020. [Online]. Available: https://fieldcommgroup.org/technologies/hart/hart-technology

[15] X. Zhao, P. Shi, Y. Yin, and S. K. Nguang, "New results on stability of slowly switched systems: A multiple discontinuous lyapunov function approach," *IEEE Trans. Autom. Control*, vol. 62, no. 7, pp. 3502–3509, Jul. 2017.

[16] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Arechiga, "Simulation-guided Lyapunov analysis for hybrid dynamical systems," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2014, pp. 133–142.

[17] J. F. Quindlen, U. Topcu, G. Chowdhary, and J. P. How, "Active sampling-based binary verification of dynamical systems," in *Proc. AIAA Guid. Navig. Control Conf.*, 2018, p. 1107.

[18] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry, "Foundations of control and estimation over lossy networks," *Proc. IEEE*, vol. 95, no. 1, pp. 163–187, Jan. 2007.

[19] M. Kögel, R. Blind, F. Allgöwer, and R. Findeisen, "Optimal and optimal-linear control over lossy, distributed networks," *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 13239–13244, 2011.

[20] E. Garone, B. Sinopoli, and A. Casavola, "LQG control over lossy TCP-like networks with probabilistic packet acknowledgements," in *Proc. 47th IEEE Conf. Decis. Control*, Cancun, Mexico, 2008, pp. 2686–2691.

[21] Y. Mo, E. Garone, and B. Sinopoli, "LQG control with Markovian packet loss," in *Proc. IEEE Eur. Control Conf.*, Zurich, Switzerland, 2013, pp. 2380–2385.

[22] R. Blind and F. Allgöwer, "Towards networked control systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process," in *Proc. 54th IEEE Conf. Decis. Control*, Osaka, Japan 2015, pp. 7510–7515.

[23] P. Ramanathan, "Overload management in real-time control applications using (m, k)-firm guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 6, pp. 549–559, Jun. 1999.

[24] M. Lješnjanin, D. E. Quevedo, and D. Nešić, "Packetized MPC with dynamic scheduling constraints and bounded packet dropouts," *Automatica*, vol. 50, no. 3, pp. 784–797, 2014.

[25] H. S. Chwa, K. G. Shin, and J. Lee, "Closing the gap between stability and schedulability: A new task model for cyber-physical systems," in *Proc. IEEE Real Time Embedded Technol. Appl. Symp.*, Porto, Portugal, 2018, pp. 327–337.

[26] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.

[27] T. Hu, Z. Lin, and Y. Shamash, "On maximizing the convergence rate for linear systems with input saturation," *IEEE Trans. Autom. Control*, vol. 48, no. 7, pp. 1249–1253, Jul. 2003.

[28] J.-X. Xu and Y. Tan, "Robust optimal design and convergence properties analysis of iterative learning control approaches," *Automatica*, vol. 38, no. 11, pp. 1867–1880, 2002.

[29] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB Second, Completely Revised*. Cham, Switzerland: Springer, 2017, vol. 118.

[30] P. I. Corke and B. Armstrong-Hélouvry, "A meta-study of PUMA 560 dynamics: A critical appraisal of literature data," *Robotica*, vol. 13, no. 3, pp. 253–258, 1995.

[31] P. I. Corke and B. Armstrong-Helouvry, "A search for consensus among model parameters reported for the PUMA 560 robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Diego, CA, USA, 1994, pp. 1608–1613.

[32] Y. Ma and C. Lu, "Efficient holistic control over industrial wireless sensor-actuator networks," in *Proc. IEEE Int. Conf. Ind. Internet*, Seattle, WA, USA, 2018, pp. 89–98.

[33] P. Zometa, M. Kögel, and R. Findeisen, "μAO-MPC: A free code generation tool for embedded real-time linear model predictive control," in *Proc. IEEE Amer. Control Conf.*, Washington, DC, USA, 2013, pp. 5320–5325.

[34] F.-L. Lian, J. Moyne, and D. Tilbury, "Modelling and optimal controller design of networked control systems with multiple delays," *Int. J. Control*, vol. 76, no. 6, pp. 591–606, 2003.

[35] M. M. Olsen and H. G. Petersen, "A new method for estimating parameters of a dynamic robot model," *IEEE Trans. Robot. Autom.*, vol. 17, no. 1, pp. 95–100, Feb. 2001.

[36] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu, "Wireless routing and control: A cyber-physical case study," in *Proc. ACM/IEEE 7th Int. Conf. Cyber Phys. Syst.*, Vienna, Austria, 2016, p. 32.

[37] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *Proc. IEEE 6th Int. Conf. Inf. Process. Sens. Netw.*, Cambridge, MA, USA, 2007, pp. 21–30.

[38] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[39] A. Zou *et al.*, "Voltage-stacked GPUs: A control theory driven cross-layer solution for practical voltage stacking in GPUs," in *Proc. 15th IEEE/ACM Int. Symp. Microarchit.*, 2018, pp. 390–402.

[40] A. Zou, J. Leng, X. He, Y. Zu, V. J. Reddi, and X. Zhang, "Efficient and reliable power delivery in voltage-stacked manycore system with hybrid charge-recycling regulators," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf.*, San Francisco, CA, USA, 2018, pp. 1–6.

[41] V. K. Reddy, B. T. Rao, and L. Reddy, "Research issues in cloud computing," *Global J. Comput. Sci. Technol.*, vol. 11, no. 11, pp. 70–76, 2011.

[42] H. El-Sayed *et al.*, "Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2017.

[43] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[44] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, Jan.–Mar. 2018.

[45] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, 2015.

[46] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[47] M. Chen, W. Li, Y. Hao, Y. Qian, and I. Humar, "Edge cognitive computing based smart healthcare system," *Future Gener. Comput. Syst.*, vol. 86, pp. 403–411, Sep. 2018.

[48] I. Burago, M. Levorato, and A. Chowdhery, "Bandwidth-aware data filtering in edge-assisted wireless sensor systems," in *Proc. 14th Annu. IEEE Int. Conf. Sens. Commun. Netw.*, San Diego, CA, USA, 2017, pp. 1–9.

[49] V. Mihai, C. Dragana, G. Stamatescu, D. Popescu, and L. Ichim, "Wireless sensor network architecture based on fog computing," in *Proc. IEEE Int. Conf. Control Decis. Inf. Technol.*, Thessaloniki, Greece, 2018, pp. 743–747.

[50] G. Ananthanarayanan *et al.*, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[51] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial Internet of Things and industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4674–4682, Oct. 2018.

[52] A. Yousefpour *et al.*, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.

[53] M. G. Ioannides, "Design and implementation of PLC-based monitoring control system for induction motor," *IEEE Trans. Energy Convers.*, vol. 19, no. 3, pp. 469–476, Sep. 2004.

[54] F. Mager, D. Baumann, R. Jacob, L. Thiele, S. Trimpe, and M. Zimmerling, "Feedback control goes wireless: Guaranteed stability over low-power multi-hop networks," in *Proc. Int. Conf. Cyber Phys. Syst.*, 2019, pp. 97–108.

[55] J. Wu and T. Chen, "Design of networked control systems with packet dropouts," *IEEE Trans. Autom. Control*, vol. 52, no. 7, pp. 1314–1319, Jul. 2007.

[56] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: A survey of recent results," *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 308–322, Feb. 2009.

[57] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Syst. Mag.*, vol. 19, no. 5, pp. 59–70, Oct. 1999.

[58] X. Dai, W. Chang, S. Zhao, and A. Burns, "A dual-mode strategy for performance-maximisation and resource-efficient CPS design," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5s, p. 85, 2019.

[59] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, 3rd Quart., 2009.

[60] P. Skarin, W. Tärneberg, K.-E. Årzen, and M. Kihl, "Towards mission-critical control at the edge and over 5G," in *Proc. IEEE Int. Conf. Edge Comput.*, San Francisco, CA, USA, 2018, pp. 50–57.

[61] Y. Ma, D. Gunatilaka, B. Li, H. Gonzalez, and C. Lu, "Holistic cyber-physical management for dependable wireless control systems," *ACM Trans. Cyber Phys. Syst.*, vol. 3, no. 1, p. 3, 2018.

[62] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Arzen, "How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime," *IEEE Control Syst. Mag.*, vol. 23, no. 3, pp. 16–30, Jun. 2003.

[63] E. Eyisi *et al.*, "NCSWT: An integrated modeling and simulation tool for networked control systems," *Simulat. Model. Pract. Theory*, vol. 27, pp. 90–111, Sep. 2012.

[64] J. Araújo, M. Mazo, A. Anta, P. Tabuada, and K. H. Johansson, "System architectures, protocols and algorithms for aperiodic wireless control systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 175–184, Feb. 2014.