Smart Actuation for End-Edge Industrial Control Systems

Ma, Yehan; Wang, Yebin; Di Cairano, Stefano; Koike-Akino, Toshiaki; Guo, Jianlin; Orlik, Philip V.; Guan, Xinping ;Lu, Chenyang

TR2022-138 November 01, 2022

Abstract

Along with the fourth industrial revolution, industrial automation systems are evolving into a multi-tier end-edge computing architecture. Edge controllers, which are equipped with a larger computing capacity compared to local controllers, can communicate with local plants over mainstream wireless networks such as WirelessHART, Wi-Fi, and cellular networks. Well-known challenges induced by networks, such as uncertain time delays and packet drops, have been intensively investigated from various perspectives: control synthesis, network design, or control and network co-design. The status quo is that the industry remains hesitant to close the loop between the edge controller and the actuation side due to safety concerns. This work offers an alternative perspective to address the safety concern, by exploiting the design freedom of an end-edge computing architecture. Specifically, we present a smart actuation framework, which deploys (1) an edge controller, which communicates with physical plant via wireless network, accounting for optimality, adaptation, and constraints by conducting computationally expensive operations; (2) a smart actuator, which is co-located with the physical plant on the end tier and executes a local control policy, accounting for system safety in the view of network imperfections, (3) the end-edge control co-design strategies and cooperation logic for both performance and stability. For certain classes of plants, semi-globally asymptotic stability of the resulting end-edge control systems is established when the edge controller is the model predictive control (MPC), or policy iteration-based learning control. We also provide an adaptation strategy for the end-edge control systems facing model parameter mismatches when the edge controller employs reinforcement learning. Extensive simulations demonstrate the advantages of the proposed end-edge co-design and cooperation procedures.

IEEE Transactions on Automation Science and Engineering 2022

Smart Actuation for End-Edge Industrial Control Systems

Yehan Ma, *Member, IEEE,* Yebin Wang, *Senior Member, IEEE,* Stefano Di Cairano, *Senior Member, IEEE,* Toshiaki Koike-Akino, *Senior Member, IEEE,* Jianlin Guo, *Senior Member, IEEE,* Philip Orlik, *Senior Member, IEEE,* Xinping Guan, *Fellow, IEEE*, and Chenyang Lu, *Fellow, IEEE*

***Abstract*—Along with the fourth industrial revolution, industrial automation systems are evolving into a multi-tier end-edge computing architecture. Edge controllers, which are equipped with a larger computing capacity compared to local controllers, can communicate with local plants over mainstream wireless networks such as WirelessHART, Wi-Fi, and cellular networks. Well-known challenges induced by networks, such as uncertain time delays and packet drops, have been intensively investigated from various perspectives: control synthesis, network design, or control and network co-design. The status quo is that the industry remains hesitant to close the loop between the edge controller and the actuation side due to safety concerns. This work offers an alternative perspective to address the safety concern, by exploiting the design freedom of an end-edge computing architecture. Specifically, we present a *smart actuation* framework, which deploys (1) an *edge controller*, which communicates with physical plant via wireless network, accounting for optimality, adaptation, and constraints by conducting computationally expensive operations; (2) a *smart actuator*, which is co-located with the physical plant on the end tier and executes a local control policy, accounting for system safety in the view of network imperfections, (3) the end-edge control co-design strategies and cooperation logic for both performance and stability. For certain classes of plants, semi-globally asymptotic stability of the resulting end-edge control systems is established when the edge controller is the model predictive control (MPC), or policy iteration-based learning control. We also provide an adaptation strategy for the end-edge control systems facing model parameter mismatches when the edge controller employs reinforcement learning. Extensive simulations demonstrate the advantages of the proposed end-edge co-design and cooperation procedures.**

***Notes to Practitioners*— Edge computing is gaining momentum in areas that require low latency and high efficiency, i.e., mobile computing, video analytics, and autonomous driving. Industrial automation systems are also evolving into a multi-tier end-edge computing architecture. It pays obvious dividends to leverage the cooperation between end and edge, benefiting from fast and reliable communication on the end side, and powerful computation capacity on the edge side. The current end-edge cooperation focuses on how to partition tasks and offload computation resources in order to minimize delay and energy consumption, as well as how to balance the tradeoff between them. However, the impacts of end-edge cooperation on the safety, optimality, and cost of industrial automation have not been systematically studied. This paper aims to tailor end-edge cooperation in a *smart actuation framework*, for industrial automation to reconcile the above aspects by leveraging co-design of end and edge controllers and their switching logic. Extensive pure and semi-physical simulations demonstrate the advantages in performance and system stability of the proposed end-edge co-design and cooperation procedures.**

***Index Terms*—Edge computing, end-edge cooperation, industrial automation, wireless networked control systems, model predictive control, reinforcement learning, stability**

I. Introduction

INDUSTRIAL automation is powered by the internet of things (IoT) technologies, such as edge computing and wireless networking. Traditionally, industrial automation relies on local controllers running on microcontrollers or programmable logic controllers (PLC), co-located with the actuators of the plants in the end tier. Limited by their sizes and costs, microcontrollers have limited computation capacity and can accommodate only simple control algorithms. The deployment of edge computing platforms in industrial control systems opens the door for improving control performance with advanced algorithms, as well as lowering installation and maintenance costs. The integration of edge computing and traditional end devices turns industrial control systems into an end-edge computing architecture, as shown in Fig. 1. Edge computing platforms comprise powerful edge servers, which communicate with physical plants through networks. Edge control over wireless networks results in wireless networked control systems (WNCSs), which find applications over industrial automation, unmanned aerial vehicles, and teleoperated robots, autonomous warehouses, and smart factories [1]. It pays obvious dividends to leverage the cooperation between end and edge in industrial automation applications, benefiting from fast and reliable communication on the end side, and powerful computation capacity on the edge side.

Since the environment of the industrial field could be harsh, and the wireless network may be exposed to metal obstacles, high temperature, high pressure, and dusty environment, WNCSs are prone to communication problems, such as poor connections and devices damages. The operation of fans, pumps, and other equipment may cause interference and reduce the reliability of network transmissions as well [2]. Therefore, the networks may suffer significant packet loss under rare but possible circumstances above or even jamming

Y. Ma is with John Hopcroft Center, Shanghai Jiao Tong University, Shanghai, 200240 China. e-mail: yehanma@sjtu.edu.cn

Y. Wang, S. Di Cairano, T. Koike-Akino, J. Guo and P. Orlik are with Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA. email: {yebinwang,dicairano,koike,guo,orlik}@merl.com

X. Guan is with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China. email: xpguan@sjtu.edu.cn

C. Lu is with Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA. email: lu@wustl.edu

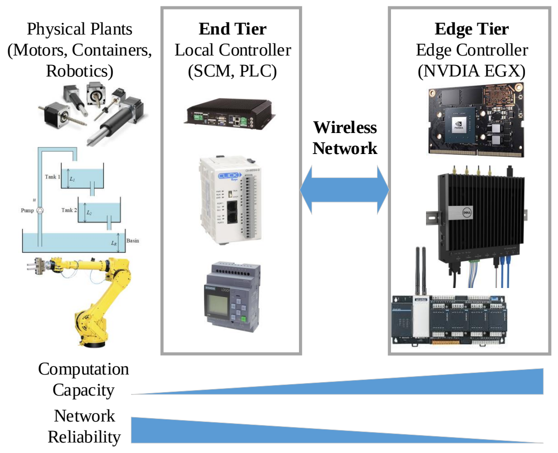


Fig. 1. End-edge computing architecture for industrial automation

attacks, even though they are usually reliable. For the sake of plant safety, it is important to ensure stability even under these challenging conditions.

This pivotal demand has spawned a variety of works to ensure system safety and performance. Most of works assume that the WNCS admits a *direct* architecture, striving to obtain guaranteed stability with conservative designs or stochastic stability, which is usually unsatisfactory to industrial practitioners. Alternatively, a *hierarchical* end-edge architecture is widely adopted in industrial automation, where local controllers fulfill stabilization under the supervision of an edge controller. The stability of the entire closed-loop system leans on the local controller, which is relatively straightforward to establish, at the expense of extra cost and insufficient authority to shape the closed-loop control system performance. This paper offers an alternative perspective to address safety and performance concerns, by exploiting the design freedom of a *smart actuation* (end-edge cooperation) architecture.

Recently, end-edge cooperation technologies have attracted pervasive attentions in areas that require low latency and high efficiency, such as mobile computing and video analysis. However, industrial control systems must meet the unique requirements to improve control performance while guaranteeing system stability, which has not been thoroughly studied yet. This paper aims at tailoring end-edge cooperation strategies for industrial control to reconcile the safety, optimality, and cost, from the perspective of co-designing controllers and their switching logic over end and edge computing tiers. As the control systems are relatively resilient to the sensing packet loss [3], we focus on the packet loss from the edge controller to actuators. The main contributions of this paper are five-fold:

1) propose a *smart actuation* framework that combines features of direct and hierarchical architectures, which is enabled by the cooperation of local and edge controllers and the switching logic between them;
2) present procedures of co-designing and coordinating edge and local control policies facing information loss, for the cases that the edge controller implements model predictive control (MPC) and policy iteration (PI) based reinforcement learning control, respectively;
3) provide an adaptation strategy facing model parameter mismatches when the edge controller implements reinforcement learning, which further illustrates the advantages and resiliency of the smart actuation framework;
4) establish semi-globally asymptotic stability (SGAS) of the resultant end-edge control systems;
5) demonstrate the effectiveness of the *smart actuation* framework in both reliable and unreliable networks through extensive pure and semi-physical simulations. The semi-physical simulation integrates real edge and local controllers, cooperation logic, Wi-Fi communication, and simulated physical plants.

The rest of the paper is organized as follows. We present related work in Section II. Section III introduces existing architectures of networked control systems and proposes the *smart actuation* architecture. Section IV presents a procedure to co-design edge and local controllers and establishes closed-loop stability when the edge controller is MPC. Section V describes an end-edge cooperation strategy to address model parameter mismatches and establishes SGAS when the edge controller employs PI-based learning. Section VI provides evaluation results of the proposed architecture and co-design procedure. Conclusions and future work are stated in Section VII.

II. Related Work

Edge control over wireless networks in the context of industrial automation faces serious challenges due to the inherent nondeterminism and limited throughput of wireless networks, which have spawned a variety of research directions in both network and control communities to ensure system safety and performance.

On the network side, standard wireless networks have been rapidly tailored for industrial automation, e.g., ISA100 [4] and WirelessHART [5]. Other approaches to address nondeterminism include, among others, deployment [6], coding [7], retransmissions and channel selection [8], and routing [9], etc. Another important research direction to improve the systems' resiliency to network imperfections is rooted in control theory. A plethora of control designs have been developed based on the plant models as well as on network parameters. To name a few, Sinopoli et al. [10] discuss Kalman filtering with intermittent measurement; Zhang et al. [11] investigate robust output tracking control subject to time-varing sampling and time delay; Truong et al. [12], [13] model packet loss as a Bernoulli or Markov-type process and establish stochastic stability of the resulting WNCS. More recently, network and control co-design has been explored to jointly determine the control and network policies to attenuate the effects of uncertainties and limited throughput, for example, [14] on network QoS-aware control, [15] on sampling periods and [16] on self-triggered control for balancing network energy consumption and control cost, [17] on control and network channel allocation, [18] on control and network scheduling policy, and [19] on control and network power policy, etc. Interested readers are referred to [20], [21] and references therein for more details.

Most of the aforementioned works assume that the WNCS admits a *direct* architecture [22]. With sensing and actuation signals being transmitted over an unreliable network, most existing works either strive to obtain guaranteed stability (albeit with conservative designs) or stochastic stability, which is usually unsatisfactory to industrial practitioners. Alternatively, a *hierarchical* end-edge architecture is widely adopted in industrial automation, where local controllers fulfill stabilization and tracking of local control loops, and an edge controller supervises local controllers over an unreliable network. It is noteworthy that the edge controller typically runs much slower than local control loops, and provides references to local counterparts for optimal process operation [22], [23]. By decoupling the unreliable network from local control loops, the stability of the entire closed-loop system is relatively straightforward to establish, at the expense of extra cost. Another shortcoming of the hierarchical architecture is that the edge controller has insufficient authority to shape the transient of the closed-loop control system, and thus might incur performance loss.

In recent years, end-edge cooperation technologies have attracted pervasive studies in areas that require low latency and high efficiency, i.e., mobile computing [24], video analytics [25], autonomous driving, and smart city [26]. The state-of-the-art end-edge cooperation focuses on how to partition tasks and offload computation resources in order to minimize delay and energy consumption, as well as to balance the tradeoff between them [27]. In contrast to those earlier applications of edge computing, industrial control systems have unique concerns [21] of overall closed-loop control performance, such as mean absolute tracking error, and system stability with complex dependencies on cyber and physical states. Edge computing has been considered for control systems recently. Vreman et al. [28] describe the challenges and their future research in the field of computer security for multilayer distributed control systems over 5G networks. Skarin et al. design and implement an industrial control system on local/edge/cloud platforms [29], and compare the impacts of the different platforms on MPC [30]. Their work however does not address coordination of computation tiers and adaptation to varying cyber-physical conditions. Ma et al. propose a switching multi-tier control (SMC) approach to exploit edge computing in control systems [31], where edge and local platforms take turns to select between the local and edge according to varying cyber-physical states. In this work, we design the stability switch between local and edge controllers under data loss from another perspective, based on co-design of edge and local control policies, which can be proved to be resilient to arbitrary information loss. The smart actuator, where the switch decision is neatly made, is the specialty of smart actuation architecture.

III. System Architectures

This section briefly describes two prevailing architectures, and proposes the *smart actuation* architecture. Interested readers are referred to [22], [32] for a comprehensive discussion of existing architectures.

A. Existing Architectures

Fig. 2 illustrates a typical schematic of *direct* architecture comprising plants, sensors, network, edge controllers, and actuators. The sensors transmit the measurements $y(k)$ to the edge controller, and the edge controller transmits control inputs $u(k)$ to the actuators, via wireless networks. Since there is only one active controller over an unreliable network, this architecture requires sophisticated control and network design to ensure the closed-loop system stability.

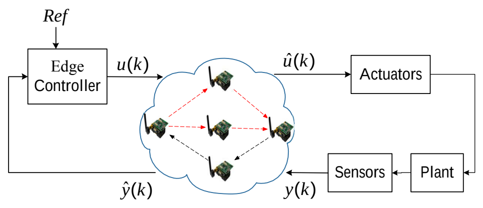


Fig. 2. Direct architecture

Fig. 3 outlines a *hierarchical* architecture. The edge controller supervises a local control loop by specifying its references, denoted by $y^*(k)$ for simplicity. The local controller generates control inputs $u(k')$ to actuators. Here we use k and k' to suggest that edge and local controllers have distinct sample rates. Since the edge controller typically runs at a much slower pace than the local one, the time scale separation principle can be applied. Thus the local controller mostly provides the closed-loop system stability. Thanks to its scalability and reliability [22], the hierarchical architecture has been found in many industrial applications such as distributed control systems for process automation, mobile robots control systems [33], and smart power grids [34]. Compared with the direct architecture, the hierarchical architecture enhances reliability but increases the system cost, through extra installation, maintenance, and cabling. In addition, the edge controller has insufficient authority to shape the transient of the closed-loop system, which limits the capacity of the edge platform to directly improve the control performance.

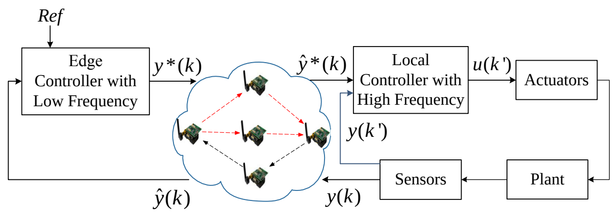


Fig. 3. Hierarchical architecture

B. Smart Actuation Architecture

The *smart actuation* architecture is shown in Fig. 4. At time step k, the edge controller determines control input $u_e(k)$ based on the estimated state $x_e(k)$ and references, and sends these signals to the local controller; the local controller

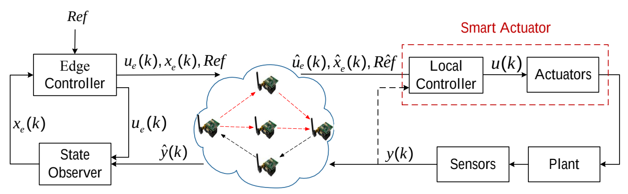


Fig. 4. Smart actuation architecture

generates $u_l(k)$ by adopting a local control policy $u_l(k) = h_l(x_l(k))$. The local controller might pass on either $\hat{u}_e(k)$ or $u_l(k)$ to actuators, depending on, for example, whether the actuation packet is delivered:

$$u(k) = \begin{cases} \hat{u}_e(k), & \text{packet delivered on time,} \\ u_l(k), & \text{otherwise,} \end{cases} \tag{1}$$

where $u(k)$ is the command to the actuators. The edge controller adopts policies, e.g., MPC and reinforcement learning, to tackle optimality, uncertainties, and constraints. The local controller implements a computationally lightweight policy as a backup in case of network performance degradation.

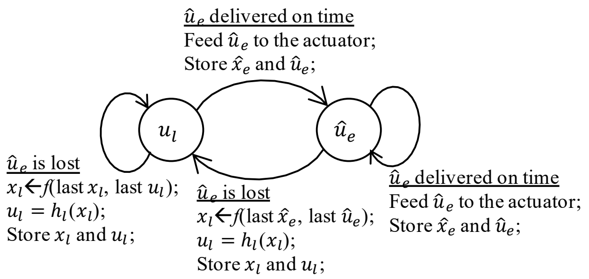
Consider the nonlinear discrete-time system,

$$x(k+1) = f(x(k), u(k)),$$

where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ the control. The local controller implements a control policy as follows:

$$\begin{aligned} x_l(k+1) &= \begin{cases} f\big(\hat{x}_e(k), u(k)\big), & \text{packet delivered,} \\ f\big(x_l(k), u(k)\big), & \text{otherwise,} \end{cases} \\ u_l(k+1) &= h_l\big(x_l(k+1)\big). \end{aligned} \tag{2}$$

As shown in the finite state machine (FSM) (Fig. 5), the smart actuation based on the switching rule of Eq. (1) works as follows. If the actuation packet from the edge, $\hat{u}_e(k)$, arrives on time, $u(k) = \hat{u}_e(k)$ according to (1), and $\hat{u}_e(k)$ and $\hat{x}_e(k)$ are stored; otherwise, $u(k) = u_l(k)$ is calculated according to (2), and $x_l(k)$ is the predicted state based on states and actuation commands at time step $k-1$.



$\hat{u}_e$ delivered on time
Feed $\hat{u}_e$ to the actuator;
Store $\hat{x}_e$ and $\hat{u}_e$;
u_l
$\hat{u}_e$
$\hat{u}_e$ is lost
$x_l \leftarrow f$(last x_l, last u_l);
$u_l = h_l(x_l)$;
Store x_l and u_l;
$\hat{u}_e$ is lost
$x_l \leftarrow f$(last $\hat{x}_e$, last $\hat{u}_e$);
$u_l = h_l(x_l)$;
Store x_l and u_l;
$\hat{u}_e$ delivered on time
Feed $\hat{u}_e$ to the actuator;
Store $\hat{x}_e$ and $\hat{u}_e$;

Fig. 5. Finite state machine for the switching rule (1)

The switching rule (1), along with edge and local control policies, should be carefully devised to secure system safety. For instance, with MPC as the edge controller, a tailored switching mechanism is described in detail in Sec. IV-B. As another instance, with reinforcement learning as the edge controller, a different switching mechanism designed for adaptability is described in Sec. V-A.

By providing control input to actuators directly, the edge controller has sufficient authority to shape the closed-loop system performance. However, this treatment may lead to the dilemma encountered in the direct architecture: unsatisfactory stability and sophisticated control design. The local controller is introduced to limit these concerns. This idea is consistent with results in [35], where the optimal locations of controllers are investigated, and which concludes that controllers should be collocated with the actuator when the size of packets is allowed to be infinitely long.

Remark 3.1: The signal flow directly connecting sensors and the local controller, represented by the dashed line in Fig. 4, is optional. With this signal flow, the *smart actuation* architecture is similar to the hierarchical one, except that the edge controllers in two architectures play different roles, and its local and edge controllers adopt the same time scale. Without directly connected sensors and the local controller, it is similar to direct architecture, except that *smart actuation* architecture has local controllers in place. □

Remark 3.2: Closing the real-time control loop over insecure networks renders WNCS vulnerable to various cyber-attacks, such as denial of service and jamming, eavesdropping, replay, zero-dynamics attacks, etc. How to mitigate cyber-attacks forms an important research thrust which has led to a fruitful of contributions, e.g. [36]–[38] and references therein. While not explicitly devoted to security aspect, this work positively enhances the resilience to cyber-attacks in two ways. First, it addresses the system stability against the loss of remote control packets, or information loss [39], which characterizes an important and common cyber-attack on quality of service (QoS). Secondly, the smart actuator can act as a watch-dog to detect cyber-attacks and respond accordingly. □

IV. Smart Actuation for Constraints and Performance

The closed-loop system corresponding to the *smart actuation* strategies over end-edge computing architecture is hybrid. Specifically, it can be regarded as a switched system arbitrarily triggered by the event designating edge-end network (actuation) packet loss. Stability analysis tools for hybrid systems can be found in [40] and references therein. This section presents stability analysis and control design based on a well-received result: if there exists a common Lyapunov function for all subsystems, then the stability of the switched system is guaranteed under arbitrary switching. Particularly we show that the construction of such a common Lyapunov function entails co-design of controllers on end and edge tiers.

A. Simplification and Assumptions

We tackle a specific problem by considering the edge control as MPC. Buffered actuation based on MPC is an efficient way to mitigate the negative effects of packet dropouts [3], [41]. Over edge, a sequence of control inputs over a finite horizon is obtained according to MPC strategy and is sent to the smart actuator. Depending on transmission outcomes and via appropriate cooperation at the smart actuator, some predicted control values can be applied to the plant. This problem is restrictive but meaningful because MPC, by taking constraints and optimality into account, provides the desired features of the edge controller. This focus implies the following assumptions.

Assumption 4.1: The closed-loop control system in the *smart actuation* framework is such that:

(i) there is no packet loss from sensors to edge controller;
(ii) delays of computations and communications within a single sampling period are ignored. Delays longer than the sampling period are regarded as packet losses.

Regarding (i) in Assumption 4.1, existing state estimation works provides robust and theoretically sound protection against loss of sensing information [10], [42], [43]. Besides, control system performance can be more sensitive to downstream (controllers to actuators) packet loss in certain WNCS [3]. Although edge has been readily utilized in sensing and monitoring for smart manufacturing [44], its application in actuation in order to close the control loop is still challenging. Therefore, in this work, we focus on the smart actuation design and stability analysis dealing with downstream while assuming upstream communication occurs over an ideal channel, which is seldomly studied and is important to close the loop over wireless networks for control and actuation. This assumption is lifted in Sec. VI. As we stated in the first paragraph of Sec. VI, the evaluation is conducted where random packet drops of both sensing and actuation sides are simulated. Regarding (ii), stability analysis under network latency of below one sampling period is well studied. We refer interested readers to [45], [46]. Furthermore, the end-to-end delay bounds of TDMA (time division multiple access) network protocol, such as WirelessHART, ISA100, Detnet, are deterministic, which can be scheduled to less than one sampling period [20]. In addition, the effects of real end-to-end communication and computation latency are reflected in the semi-physical simulation in Sec. VI-E.

Regarding the plant, we have the following assumptions to facilitate stability analysis and control design.

Assumption 4.2: The open-loop plant model is exactly known, and its states are measured. Furthermore, it is stabilizable by either the MPC or a state feedback control policy $u_l(k) = h_l\big(x(k)\big)$, when there is no packet loss.

B. Nonlinear Input-affine Sytems

Consider a nonlinear control-affine discrete-time system

$$\begin{aligned} x(k+1) &= f\big(x(k)\big) + g\big(x(k)\big)u(k) \\ y(k) &= x(k), \end{aligned} \tag{3}$$

where $x \in \mathbb{X} \subset \mathbb{R}^n$ is the state, f and g smooth vector fields, $u \in \mathbb{U} \subset \mathbb{R}^m$ the control input, and y the output. Both $\mathbb{X}$ and $\mathbb{U}$ are convex and compact, with each set containing the origin in its interior. The control objective is to steer states to the origin while minimizing a certain cost function.

Next, we illustrate the co-design of the edge MPC and local control policies for system (3) to ensure that the resulting closed-loop system is SGAS.

1) Local Controller Design: Feedback stabilizing or tracking control design for a nonlinear system (3) is one of the fundamental problems in control theory, which is however not the focus of this work. As specified in Assumption 4.2, there exists a smooth state feedback law $u_l(k) = h_l\big(x(k)\big)$ such that the resulting closed-loop system is SGAS, i.e., a continuously differentiable function $V_l : \mathbb{R}^n \to \mathbb{R}$ satisfies [47, Thm 4.2].

Remark 4.3: Several control designs lead to $u_l(k) \in \mathbb{U}, \forall x \in \mathbb{X}$, which renders $\mathbb{X}$, (more likely its subset represented as a level set of V_l), an invariant set of the closed-loop system. This work assumes that $u_l(k)$ always lies in $\mathbb{U}$ for all $x \in \mathbb{X}$, and makes $\mathbb{X}$ an invariant set. □

2) Edge Controller Design: For a variable ρ, given $i \geq 0$, $\rho(i|k)$ denotes the prediction of $\rho(k+i)$, based on information available at time k; and $\rho(k) = \rho(0|k)$. For system (3) at time k, MPC minimizes the cost function [48],

$$V\big(x(k), \mathbf{u}(k)\big) = F\big(x(N|k)\big) + \sum_{i=0}^{N-1} l\big(x(i|k), u(i|k)\big), \tag{4}$$

where $\mathbf{u}(k) = \mathbf{u}_e(k) = \{u_e(k), u_e(1|k), ..., u_e(N-1|k)\}$, $x(i|k)$ for $1 \leq i \leq N-1$ are the predicted states corresponding to $\mathbf{u}_e(k)$, and N is the prediction horizon. The positive definite functions $l(x,u)$ and $F(x)$ represent the stage cost and the terminal cost, respectively. At time k, the MPC solves an optimization problem by minimizing (4), subject to state/control constraints along the state trajectory and a terminal constraint $x(N|k) \in X_f \subset \mathbb{X}$. Assumption 4.2 implies that the optimization problem has an optimal solution $\mathbf{u}_e^*(k) = \{u_e^*(k), u_e^*(1|k), ..., u_e^*(N-1|k)\}$ at time k, and the associated cost function is given by $V_e^*\big(x(k)\big) = V\big(x(k), \mathbf{u}_e^*(k)\big)$.

Assumption 4.2 indicates that there exist functions $F(\cdot), l(\cdot,\cdot), \mathcal{K}_f(\cdot)$ satisfying A1 to A4 [48, Sec. 3.3]:

A1: $X_f \subset \mathbb{X}$, where X_f is closed and $0 \in X_f$.
A2: There exists a local controller $\mathcal{K}_f(x) \in \mathbb{U}$, $\forall x \in X_f$.
A3: $\big(f(x) + g(x)\mathcal{K}_f(x)\big) \in X_f$, $\forall x \in X_f$.
A4: $F\big(f(x)+g(x)\mathcal{K}_f(x)\big) + l\big(x, \mathcal{K}_f(x)\big) \leq F(x), \forall x \in X_f$.

To establish the stability of the hybrid system resulting from the edge MPC and local controller, it is sufficient to prove that (4) is a common Lyapunov function for subsystems associated with the local control policy and the MPC.

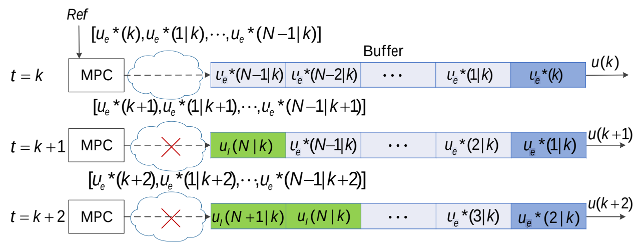


Fig. 6. End-edge switch mechanism when edge controller employs MPC

Remark 4.4: Motivated by [3], [41], where the actuation buffer is used to address time delays and packet loss, the *end-edge switch mechanism* illustrated by Fig. 6 is alternatively executed by the smart actuation strategy in the stability analysis below. Assume that the smart actuator on the end side has a buffer of size $L = N$, and the buffer stores a control sequence $\mathbf{u}(k) = \{u_e^*(k), u_e^*(1|k), \ldots, u_e^*(N-1|k)\}$. If the actuation packet is delivered at time $k+1$, the buffer is refreshed by the MPC control sequence $\mathbf{u}_e^*(k+1)$; otherwise, the local control policy $u_l(N|k)$ is pushed into the buffer. Consequently, the control sequence in the buffer turns into $\mathbf{u}(k) = \{u_e^*(1|k), \ldots, u_e^*(N-1|k), u_l(N|k)\}$. Previous study [41] discussed whether the last input of buffer should be set to 0 or the last value when packet drops. We have developed a third option by setting it as local control input, which is favorable to stability. □

Proposition 4.5: Assume that a local control policy $u_l(k) = h_l\big(x(k)\big)$ renders system (3) globally asymptotically stable, and that for a certain positive definition function $J(x)$ and $\alpha \geq 1$, the following condition holds, for $0 \leq k < \infty, \forall x \in \mathbb{X}$,

$$\alpha \cdot l\big(x(k), h_l(x(k))\big) + J\big(x(k+1)\big) - J\big(x(k)\big) \leq 0. \quad (5)$$

Then the cost function (4), with $F(x) = J(x)$ and the stage cost $l(x,u)$, is a common Lyapunov function of the closed-loop system, where control input switches between MPC and local policy $u_l(k) = h_l\big(x(k)\big)$, according to Remark 4.4.

Proof: We need to show the following facts

(i) the subsystem resulting from the edge MPC policy has a Lyapunov function given by $V_e^*\big(x(k)\big)$;
(ii) with $F(x) = J(x)$, (4) is a Lyapunov function of the subsystem resulting from local control policy $u_l(k)$;
(iii) in the case of switching from the edge MPC policy to the local control policy $u_l(k)$, the Lyapunov function decreases;
(iv) in the case of switching from the local control policy $u_l(k)$ to the edge MPC policy $u_e(k)$, the Lyapunov function deceases.

Proof of (i): With conditions A1–A4, the cost function (4) is a Lyapunov function for the subsystem corresponding to the edge MPC [48]. Proof is omitted.

Proof of (ii): Given $x(k)$, local policy $u_l(k) = h_l\big(x(k)\big)$, and system (3), we have $u_l(n|k) = u_l(k+n)$ and $x(n+1|k) = x(k+n+1)$, where $0 \leq n \leq N$. Therefore, we have $\{u_l(k), ..., u_l(k+N)\}$, $\{x(k+1), \ldots, x(k+N+1)\}$, and

$$\begin{aligned} V\big(x(k+1)\big) &= J\big(x(k+N+1)\big) + \sum_{i=1}^{N} l\big(x(k+i), u_l(k+i)\big) \\ V\big(x(k+1)\big) - V\big(x(k)\big) &= J\big(x(k+N+1)\big) - J\big(x(k+N)\big) \\ &\quad - l\big(x(k), u_l(k)\big) + l\big(x(k+N), u_l(k+N)\big). \end{aligned}$$

Substituting (5) into the above equation gives

$$\begin{aligned} V\big(x(k+1)\big) - V\big(x(k)\big) \leq &- (\alpha-1) l\big(x(k+N), u_l(k+N)\big) \\ &- l\big(x(k), u_l(k)\big), \end{aligned}$$

which is negative definite and implies (ii).

Proof of (iii): The induction principle is used here. Assume that the edge MPC policy $\mathbf{u}_e^*(k) = \{u_e^*(k). \ldots, u_e^*(N-1|k)\}$ is applied at time k. Lyapunov function is $V_e^*(k)$. With the packet drop at time $k+1$, the control sequence is $\mathbf{u}(k+1) = \{u_e^*(1|k). \ldots, u_e^*(N-1|k), u_l(N|k)\}$, where $u_l(N|k)$ is the local control policy. We have

$$\begin{aligned} V(k+1) - V_e^*(k) = &J\big(x(N+1|k)\big) + \sum_{i=1}^{N-1} l\big(x(i|k), u_e^*(i|k)\big) \\ &+ l\big(x(N|k), u_l(N|k)\big) - J\big(x(N|k)\big) - \sum_{i=0}^{N-1} l\big(x(i|k), u_e^*(i|k)\big), \end{aligned} \quad (6)$$

where $x(N+1|k)$ is obtained from applying $u_l\big(x(N|k)\big)$ to (3). Applying (5) to (6), one verifies $V(k+1) - V_e^*(k) < -l\big(x(k), u_e^*(k)\big)$. By induction, one can repeat the aforementioned derivation to establish the decrease of the Lyapunov function, if the packet loss continues beyond time $k+1$. This completes the proof of (iii).

Proof of (iv): $V_e^*(k+1) \leq V_e^*(k)$ because of (i). At time k, since the control sequence in the buffer is a feasible solution with a cost $V(k)$, the MPC policy $\mathbf{u}_e^*(k)$, obtained by solving the optimization problem, necessarily yields $V_e^*(k) \leq V(k)$. Therefore $V_e^*(k+1) - V(k) \leq 0$. ■

Remark 4.6: Proposition 4.5 establishes that the resulting closed-loop system is SGAS over $\mathbb{X}$, by imposing a restrictive condition (5) over $\mathbb{X}$. This restriction can be lifted in many cases by relaxing Assumption 4.2 to hold over X_f. The relaxation, together with A3 for $u_l = h_l(x) = \mathcal{K}_f(x)$ and assuming the successful delivery of the actuation packet at $k = 0$, also ensures the SGAS over $\mathbb{X}$. □

Remark 4.7: Condition (5) is sufficient but not necessary. It is for verification but not for control synthesis. Given a feedback control $u_l = h_l(x)$ satisfying Theorem [47, Thm 4.2], the stage cost $l(x,u)$, and $J(x)$, it is straightforward to verify whether condition (5) holds or not. However, it is not trivial to construct the function $J(x)$ and $u_l = h_l(x)$ from condition (5) for a given $l(x,u)$. This is because $u_l = h_l(x)$ is associated with a Lyapunov function V_l, and leaves V_l decay at a certain rate which is irrelevant to $l(x,u)$. □

3) Policy Evaluation-based Co-Design Procedure: We employ the following *policy evaluation-based procedure* to bridge the gap from $u_l = h_l(x)$ to the construction of $J(x)$.

1) Design a stabilizing local controller $u_{l0} = h_0(x)$.
2) Given $u_{l0} = h_{l0}(x)$, $l(x,u)$, and $\gamma = 1$, perform one step of the policy evaluation as in (21) to evaluate the cost $V_1(x)$ corresponding to the control $u_{l0} = h_{l0}(x)$, and set $J(x) = V_1(x)$.
3) Solve the MPC policy by minimizing the cost function (4) with $F(x) = J(x)$.

Given a local control law $u_l = u_{l0}$, the *policy evaluation-based co-design procedure* eventually outputs a terminal cost $J(x) = V_1(x)$. We have $J\big(x(k+1)\big) - J(x) = -l\big(x(k), u_l(k)\big)$ according to (21), which satisfies condition (5). This implies the stability of the resulting closed-loop system, according to *Proposition* 4.5.

Remark 4.8: As a system of first-order nonlinear difference equations, the closed-form solution of (5) is difficult to obtain. An approximate solution is usually of practical interest. Given $u_l = h_l(x)$ and parameterizations of $J(x)$, (5) is reduced to algebraic equations, and thus, the approximate solution of $J(x)$ can be computed. See Appendix A for details. □

The control policy $u_l = h_l(x)$ is designed to make V_l decay along the system trajectory. This implies that u_{l0} may be far from optimal with respect to (4). The *policy evaluation-based procedure* may give $J(x)$, which is inconsistent with the stage cost. A remedy to this issue is to introduce *policy iteration-based co-design procedure* as follows:

1) Design a stabilizing controller $u_{l0} = h_{l0}(x)$ and let $j = 0$.
2) Given $u_{lj} = h_{lj}(x)$ and $l(x, u)$, perform policy evaluation (see (21) in Appendix) to evaluate the cost $V_{j+1}(x)$ corresponding to the control u_{lj}.
3) Given $V_{j+1}(x)$, perform policy improvement (see (22) in Appendix).
4) $j = j + 1$; repeat the policy evaluation and policy improvement steps until $j = M$, and then let $J(x) = V_{M+1}(x)$. M denotes the allowed number of iterations.
5) Solve the MPC policy by minimizing the cost function (4) with $F(x) = J(x)$.

Given the stabilizing control policy u_{l0}, the control policy updated in the policy improvement step also stabilizes the system (3). Hence, the *policy iteration-based co-design procedure* will eventually produce a local control policy $u_l = u_{lM} = h_{lM}(x)$ and terminal cost $J(x) = V_{M+1}(x)$ in (4) of edge control, which satisfy condition (5). This implies the stability of the resulting closed-loop system.

Remark 4.9: Policy evaluation and iteration-based co-design procedures can be extended to take control constraints into account. See [49] for details. □

C. LTI System Case

Consider a linear time-invariant system

$$x(k+1) = Ax(k) + Bu(k), \quad y(k) = x(k). \tag{7}$$

1) Controller Design: We take $l(x, u) = x^T Qx + u^T Ru$, and $F(x) = x^T Sx$ in (4) of edge control, where Q, R, S are positive definite. The cost function becomes

$$\begin{aligned} V\big(x(k)\big) = &\sum_{i=k}^{k+N-1} \big(x^T(i)Qx(i) + u^T(i)Ru(i)\big) \\ &+ x^T(k+N)Sx(k+N). \end{aligned} \tag{8}$$

One can validate that (8) is a common Lyapunov function of the resulting hybrid system, and obtain the following result.

Proposition 4.10: Consider system (7), the cost function (8), and the switching policy in Remark 4.4, let S be the solution of the discrete-time algebraic Riccati equation (DARE)

$$A^T SA - S + Q - A^T SB(B^T SB + R)^{-1}B^T SA = 0, \tag{9}$$

and let the local control policy be given by $u(k) = -(B^T SB + R)^{-1}B^T SAx(k)$. The resulting hybrid system is asymptotically stable.

V. Adaptation to Parameter Mismatches

Previously, both edge and local controllers assume exact knowledge of the plant. This is not always the case in practice. In order to maintain system stability and performance, either local or edge controller shall incorporate adaptation mechanisms to address parametric model uncertainties. In case that both controllers have access to the same information, adaptation might reside in the local controller to avoid adversary effects of network imperfection. However, for an automation system comprising multiple coupled local machines, the edge controller can access more information and computing power, and thus conduct adaptation more effectively. Thus here we implement adaptation on the edge, as an illustration.

A. Policy Iteration-based Adaptation

Let the plant model be given by

$$\dot{x} = f(x, \theta) + g(x)u, \quad y = x, \tag{10}$$

where f, g are smooth vector fields, and $\theta \in \Omega_\theta \subset \mathbb{R}^{n_\theta}$ is a vector of unknown parameters with Ω_θ being a compact set. One way to enable adaptation is to replace the baseline MPC with adaptive variants, e.g. [50]. Reinforcement learning-based control bridges the gap between traditional optimal control and adaptive control algorithm by enabling adaptive control in an optimal manner. [51] Thus, this work adopts the online reinforcement learning approach [52] in virtue of its guaranteed convergence and performance improvement. Particularly, the edge controller not only performs parameter estimation, but also iteratively synthesizes control policies for local controllers to implement in real-time. Fig. 7 illustrates the architecture where the PI algorithm is implemented on the edge to learn control policies, which are then pushed towards local controllers. Here PI determines a control policy based on parameter estimation, and usually has a much slower time scale than the parameter estimator. Roughly speaking, the edge controller samples a parameter estimate trajectory $\hat{\theta}(t)$ sparsely over time and obtains a sequence: $\{\hat{\theta}_i\}$. Given an estimate $\hat{\theta}_i$, it performs PI to develop an optimal control policy $u_i^*(x)$ and the corresponding value function $V_i^*(x)$.

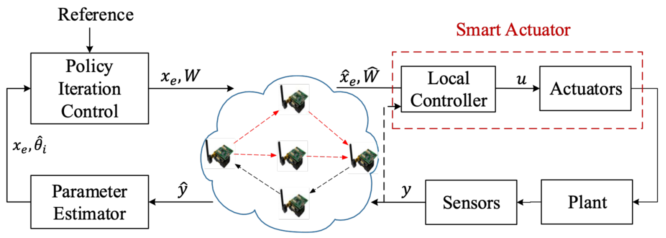


Fig. 7. Smart actuator diagram when edge controller employs policy iteration

In PI, the control policy and value function are approximated as $V^*(x) = W^T\Phi(x)$, and $u^*(x) = \Gamma^T\Psi(x)$, respectively, where W and Γ are vectors of parameters to be updated as described in Appendix A. Readers are referred to [52] for more details. The implementation of smart actuator depends on whether it receives updated control policy packet from edge and receives y from sensors directly or not. It updates control policy once the control policy packet carrying W_i arrives. Let the smart actuator implement control policy $u_i^*(x) = -\frac{1}{2}R^{-1}g^T(x)\frac{\partial\Phi^T(x)}{\partial x}W_i$ (according to (24)) with W_i

being synthesized from PI based on $\hat{\theta}_i$. With y from sensors, it computes $u_i^*(\hat{x}_e)$; otherwise, it calculates $u_i^*(x_l)$. That is:

$$u_l(k) = \begin{cases} u_i^*(\hat{x}_e(k)), & \hat{x}_e \text{ packet delivered} \\ u_i^*(x_l(k)), & \text{otherwise.} \end{cases} \tag{11}$$

B. Stability Analysis

Once the local controller deploying control policy u_i^*, the resultant closed-loop system is obtained as

$$\dot{x} = f(x,\theta) + g(x)u_i^*(x) \equiv f_i(x,\theta,\tilde{\theta}_i) \tag{12}$$

where $u_i^*(x)$ implicitly depends on the parameter estimate $\hat{\theta}_i$. System (12) is perturbed by $\tilde{\theta}_i = \theta - \hat{\theta}_i$. The fact that the local controller deploys control policies $u_i^*(x), 0 \le i < \infty$, results in a switched system. The analysis below casts insight on how a new control policy should be deployed to ensure safety. The following assumptions are made to facilitate the analysis.

Assumption 5.1: Consider the plant (10).

(i) The vector field f is Lipschitz in $\theta \in \Omega_\theta$, uniformly in a compact set Ω_x, i.e., there exists $L > 0$ such that

$$\left\| f(x,\theta) - f(x,\hat{\theta}) \right\| \le L \left\| \tilde{\theta} \right\|, \ \forall \theta, \hat{\theta} \in \Omega_\theta, \forall x \in \Omega_x.$$

(ii) There exists an initial control policy $u_0(x)$ such that the closed-loop system state trajectory stays inside Ω_x for any $x(0) \in \Omega_x$. Particularly, the zero solution of (12) is asymptotically stable with $\tilde{\theta}_0 = 0$.

(iii) There exists a parameter estimator achieving convergent estimation of θ, i.e., $\lim_{t\to\infty} \left\| \tilde{\theta}(t) \right\| = 0$.

(iv) PI solves for a Lyapunov function satisfying

$$\frac{\partial V}{\partial x}\big(f(x,\hat{\theta}) + g(x)h(x,\hat{\theta})\big) = -(x^\top Q x + u^\top R u)$$
$$\left\| \frac{\partial V}{\partial x} \right\| \le c_3 \|x\|.$$

Remark 5.2: Conditions (i)-(ii), imposed on the plants, hold for most of the engineering systems, e.g. servomotors [53], Lithium-ion batteries [54], etc. Given the continuity of f and the boundedness of Ω_x, (i) always holds; and (ii) implies that the plant can be stabilized by static state feedback control. Conditions (iii)-(iv) are related to tools employed for control design and stability analysis. Particularly, (iii) facilitates indirect adaptive control design and input-to-state stability (ISS) analysis [55], whereas (iv) always holds as long as (ii) is true and $x \in \Omega_x$. Condition (iii) can be removed or relaxed, for example, when the PI algorithm synthesizes the control policy without using the knowledge of $f, \hat{\theta}_i$ [52]. □

From (i)-(iii) in Assumption 5.1, Lemma 4.7 in [47] can be applied to show that the closed-loop system, with $\tilde{\theta}$ being a decaying unknown disturbance, is asymptotically stable for any $x_0 \in \Omega_x$. The stability holds for control policy $u(x,\hat{\theta})$ if $\hat{\theta}$ is updated continuously. Given a particular policy $u_i^*(x)$, its deployment to local controller will not warrant asymptotic stability, owing to its implicit dependence on $\hat{\theta}_i$ and the non-zero error $\tilde{\theta}_i$. Instead, we have the following result.

Proposition 5.3: With Assumption 5.1, the resulting closed-loop system (12) is uniformly ultimately bounded over Ω_x.

Proof: Given a parameter estimate $\hat{\theta}_i$, the PI procedure yields a control policy $u_i^*(x)$ and the value function $V_i^*(x)$, based on the nominal plant dynamics

$$\dot{x} = f(x,\hat{\theta}_i) + g(x)u. \tag{13}$$

From (ii) of Assumption 5.1 and the property of PI, we know $u_i^*(x)$ asymptotically stabilizes (13). When u_i^* is applied to (10), the time derivative of V_i^* along the state trajectory is

$$\begin{aligned}\dot{V}_i^* &= \frac{\partial V_i^*}{\partial x}(f(x,\theta) + g(x)u_i^*(x) + f(x,\hat{\theta}_i) - f(x,\hat{\theta}_i)) \\ &\le -(c_4 - c_3 L\gamma^2)\|x\|^2 + \frac{c_3 L}{4\gamma^2}\left\|\tilde{\theta}_i\right\|^2,\end{aligned}$$

where both $\tilde{\theta}_i$ and $\gamma > 0$ are constant. One can see that $\dot{V}_i^* \le -c_5 \|x\|^2, \forall \|x\| \ge \sqrt{\frac{c_3 L}{4\gamma^2(c_4 - c_3 L\gamma^2 - c_5)}} \left\|\tilde{\theta}_i\right\|$. From [47, Lem. 4.6], we conclude that the closed-loop system with u_i^* is ISS, i.e., there exists a class $\mathcal{KL}$ function β and a class $\mathcal{K}$ function Γ

$$\|x(t)\| \le \beta(\|x(0)\|, t) + \Gamma(\sup_{0\le\tau\le t} \left\|\tilde{\theta}_0\right\|).$$

With $\Gamma(\tilde{\theta}_0)$ being constant, we show $x(t)$ is bounded. ■

Having received a sequence of control policies $u_i^*(x)$, local controller needs mechanisms to ensure that the switch from $u_i^*(x)$ to $u_{i+1}^*(x)$ is safe. We have the following result.

Proposition 5.4: Consider plant (10) where Assumption 5.1 holds. Denote $\{\hat{\theta}_0, \dots, \hat{\theta}_i, \dots\}$ the sequence of parameter estimates that the edge controller samples at intermittent time instants. The edge controller performs PI for each $\hat{\theta}_i$ and obtains control policy $u_i^*(x)$ and value function $V_i^*(x)$, which are delivered successfully to the local controller. There exists a sub-sequence of control policies to be deployed such that the induced closed-loop system is SGAS over Ω_x.

Proof: For the closed-loop system resulting from u_i^*, we choose Lyapunov function candidate $V_i(x,\tilde{\theta}) = V_i^*$. Its time derivative is $\dot{V}_i \le -c_{i,1}\|x\|^2 + c_{i,2}\left\|\tilde{\theta}_i\right\|^2, \forall x \in \Omega_x$, where $c_{i,1}, c_{i,2}$ are positive constant. This means the state $x(t)$ will eventually be confined in the ball $\mathcal{B}_i \triangleq \{x| \, \|x\| \le \sqrt{\frac{2c_{i,2}}{c_{i,1}}}\left\|\tilde{\theta}_i\right\|\}$ as long as the policy u_i holds long enough. For the policy u_{i+1}^* we have $\dot{V}_{i+1} \le -c_{i+1,1}\|x\|^2 + c_{i+1,2}\left\|\tilde{\theta}_{i+1}\right\|^2$. If u_{i+1}^* is applied in $[t_{i+1}, t_{i+2}]$, as long as the duration $t_{i+2} - t_{i+1}$ is long enough, the state trajectory of the resulting closed-loop system will enter and could not escape from the ball $\mathcal{B}_{i+1} \triangleq \{x| \, \|x\| \le \sqrt{\frac{2c_{i+1,2}}{c_{i+1,1}}}\left\|\tilde{\theta}_{i+1}\right\|\}$. Since $\tilde{\theta}$ is convergent, we can always pick t_{i+2} such that $\mathcal{B}_{i+1} \subset \mathcal{B}_i$. Given the contractive property of $\mathcal{B}_i, 0 \le i < \infty$, one can verify that the closed-loop system satisfies the definition of SGAS over Ω_x. ■

VI. Evaluation

This section presents three examples to illustrate and verify the co-design procedure of local and edge controllers as described in Sec. IV, and a case study of adopting edge adaptive controllers as discussed in Sec. V. Performances of control systems with different architectures are compared to corroborate the effectiveness of the smart actuation architecture. Simulations are conducted in MATLAB/Simulink®, where random packet drops of both sensing and actuation sides are simulated. Sensing packet losses are handled by the EKF with intermittent observations [10]. Actuation packet losses are compensated according to the different architectures, e.g. edge controller only (buffered MPC actuation and reinforcement learning), local controller only, and smart actuation (adopting the switch policy (1)) architectures.

Detailed simulation results are given in Figs. 9 - 19. Each boxplot figure shows the results of five different cases by taking edge MPC controller as an example:

1) *MPC_I*: edge MPC controller and smart actuation strategy in an ideal network. Since there is no packet loss, the edge controller is always active.
2) *Local_I*: local controller in an ideal network.
3) *Smart*: smart actuation strategy with both local and edge controllers, but an unreliable network. The hybrid system adopts the switching policy given by (1).
4) *Local*: local controller and a lossy network. In this case, the EKF transmits estimated states $x_e(k)$ to the local controller via an unreliable network. If the packet arrives, the local controller generates control inputs based on $x_e(k)$; otherwise, it generates control inputs based on $x_l(k)$.
5) *Edge*: direct architecture with the buffered MPC scheme [41] over an unreliable network, with buffer size 5.

Each boxplot is generated from 50 rounds of simulations.

A. Example 1: Linear System

Consider the load-positioning system in [56], [57], which positions the load (L) using a motor. The motor is attached rigidly to a movable base platform (B). The load positioning is a 4-state nonlinear system [57]. When the system is operated at the low frequencies common in industrial settings, the system can be simplified to a 4-state linear system [56]. The state vector is defined as $x = [x_L\ \dot{x}_L\ x_B\ \dot{x}_B]$, where x_L is the displacement of the load relative to the base platform, x_B is the absolute displacement of the base platform, and $\dot{x}_L$ and $\dot{x}_B$ are the speeds of the relative and absolute movements accordingly. Thus the load positioning system model is

$$\dot{x} = A_c x + B_c u, \quad y = x,$$

where system parameters are given in Table I and

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -d_L(\frac{1}{m_L} + \frac{1}{m_B}) & \frac{k_B}{m_B} & \frac{d_B}{m_B} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{d_L}{m_B} & -\frac{k_B}{m_B} & -\frac{d_B}{m_B} \end{bmatrix},$$

$$B_c = \begin{bmatrix} 0 \\ \frac{1}{m_L} + \frac{1}{m_B} \\ 0 \\ -\frac{1}{m_B} \end{bmatrix}.$$

For simplicity, we discretize the continuous-time model using Eluer method, and have the discrete-time model denoted by $x(k+1) = A_d x(k) + B_d(k)$.

TABLE I
SYSTEM PARAMETERS

variable	value	variable	value	variable	value
m_L	10	m_B	20	d_L	15
d_B	0.5	a_1	0.03	b_1	0.7
b	2	c	1.6	d	1.6
B	0.05	C	10^{-3}	D	164.99
α	10	K_l	1	k_B	0.1
a	0.7	A	0.0333	E	123.74

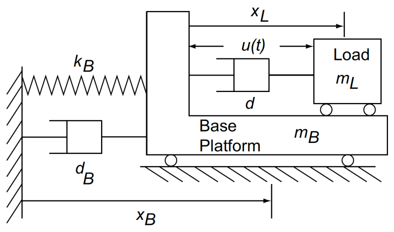


Fig. 8. Diagram of load-positioning system [56]

1) Controller Design: We would like to stabilize the load positioning system to the origin while minimizing the cost function (8) with $Q = I_4$ and $R = 1$. For simplicity, we consider the unconstrained control case. The controller design is to determine the local control policy and the matrix S. Solving the DARE (9), we have

$$S = \begin{bmatrix} 102.52 & 189.35 & 33.07 & 447.29 \\ 189.35 & 5611.85 & 277.79 & 16591.79 \\ 33.07 & 277.79 & 172.05 & 859.10 \\ 447.29 & 16591.79 & 859.10 & 49233.75 \end{bmatrix}.$$

According to *Proposition* 4.10, we have the local control policy $u_l(k) = -(B_d^T S B_d + R)^{-1} B_d^T S A_d x(k)$.

2) Simulation Results: Simulation results are given in Fig. 9, where the simulation time of each round is 600s, and the sampling frequency is 6Hz. Fig. 9 (a) shows the costs when the unreliable network is subject to 80% random packet loss. The *Local* case gives the lowest cost in both ideal and lossy networks. The cost of the *Smart* case is a little higher than the *Local_I* case, but lower than the *Edge* case. This is because the local control $u_l(k)$ is the optimal solution of the LQR problem, while the edge MPC, although yielding the exact LQR solution in a receding horizon manner, is compromised by network. Fig. 9 (b) indicates that, when the edge-end network loses connection from 25 s to 125 s, both the *Smart* and *Local* cases still work properly. However, the performance of the *Edge* case deteriorates drastically because the limited buffer size fails to mitigate network failure lasting for 100s.

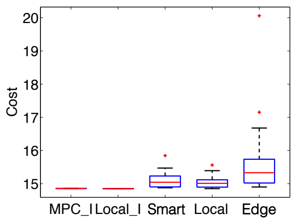
B. Example 2: First Order Nonlinear System

Consider a general first order nonlinear plant case.

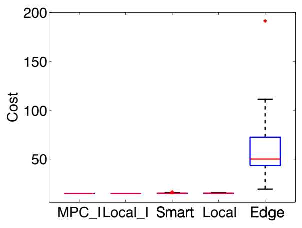
$$\dot{x} = a_1 x^2 + b_1 u,$$

where model parameters are provided in Table I.

1) Controller Design: We would like to regulate the state to the origin while minimizing the cost (4), where $l(x,u) = x^T Q x + u^T R u$, the terminal cost $F(x) = S(x)$. The local control law is $u_l = \frac{1}{b_1}(-x - a_1 x^2)$. To design terminal cost $S(x)$



(a) 80% packet loss



(b) 80% packet loss, and the edge-end network loses connection from 25s to 125s

Fig. 9. Costs of the linear system case

for the edge MPC policy, we take $Q = 1, R = 1, \alpha = 1.2$, and choose $S(x) = x^T W x$, with W being a positive definite matrix. Based on the policy evaluation-based co-design method given in Sec. IV-B3, we obtain $S(x) = 10.16x^2$. As long as $S\big(x(k+1)\big) - S\big(x(k)\big) \le x^T(k)Qx(k) + u_l^T(k)Ru_l(k)$ holds for the entire feasible sets of state and control inputs, (5) is satisfied. The results of policy evaluation are given by Fig. 10, where the curve of $S\big(x(k+1)\big) - S\big(x(k)\big)$ is always below that of $x^T(k)Qx(k) + u_l^T(k)Ru_l(k)$ in the feasible set of x.

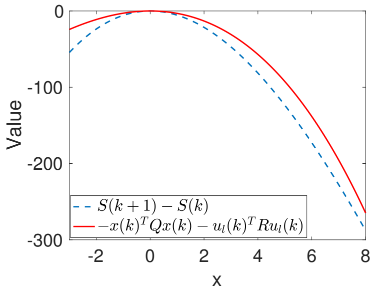


Fig. 10. Policy evaluation results for the first order nonlinear system

2) Simulation Results: Fig. 11 presents the costs of five different cases for the first order nonlinear system. The simulation time of each round is 60 s, and the sampling frequency is 3Hz. *MPC_I* outperforms *Local_I* since the former takes optimality into account by solving a finite-time optimal control problem versus the latter merely solves the stabilizing problem. When there is 20% packet loss, the *Smart* case outperforms the *Local* and *Edge* cases in the sense that the cost distribution of the *Smart* has a lower mean value than the *Local* case, and has a smaller variance than the *Edge* case.

C. Example 3: Second Order Nonlinear System

Consider the following second order nonlinear plant

$$\dot{x}_1 = a x_1^2 - b x_1^3 + c x_2$$
$$\dot{x}_2 = u,$$

where system parameters are provided in Table I.

1) Controllers Design: Again, we would like to regulate the state to the origin while minimizing the cost, which has same format with the cost function in Sec. VI-B and $Q = 3I_2, R = 1$. The local control law is given by:

$$z = x_2 + \frac{a}{c}x_1^2 + \frac{d}{c}x_1$$
$$u_l = -z - x_1 - \Big(\frac{2a}{c}x_1 + \frac{d}{c}\Big)(-bx_1^3 - dx_1 + z).$$

Following the policy evaluation-based design method in Sec. IV-B3, we choose $\alpha = 3$, and get $S(x) = 17.39x_1^2 + 23.16x_2^2$. Fig. 12 presents the results of policy evaluation. One can verify that in the feasible set of x, $S\big(x(k+1)\big) - S\big(x(k)\big) \le x^T(k)Qx(k) + u_l^T(k)Ru_l(k)$. Then (5) is satisfied.

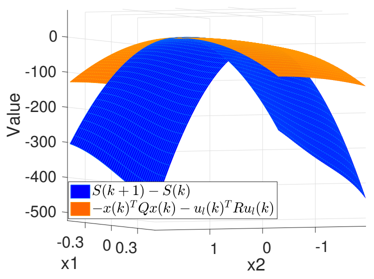
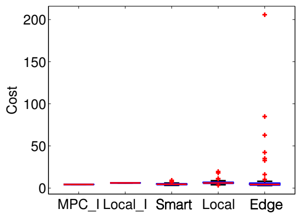
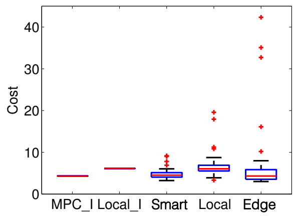


Fig. 12. Policy evaluation for the second order nonlinear system

2) Simulation Results: Fig. 13 (a) and (b) show the costs of five different cases, under 0%, 30%, and 70% of packet loss. Simulation time of each round is 100s, and the sampling frequency is 10Hz. Again, with the ideal network (0% scenario), the cost of *MPC_I* is smaller than that of *Local_I*. With 30% packet loss (Fig. 13 (a)), the *Local* case has a higher cost than both the *Smart* and the *Edge* cases. Particularly, the *Smart* case outperforms the *Edge* case in the sense that both the mean value and variance of the cost distribution for the former are smaller. When the network suffers from more

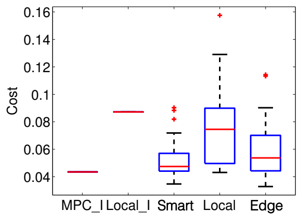


(a) Costs when 20% packet loss

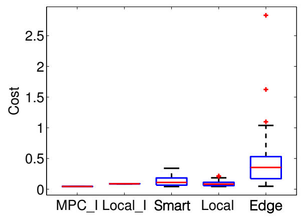


(b) Costs when 20% packet loss (zoom in)

Fig. 11. Costs of the first order nonlinear system under 20% of packet loss



(a) 30% packet loss



(b) 70% packet loss

Fig. 13. Costs of 2-state nonlinear system

severe packet loss (Fig. 13 (b)), both the *Smart* and *Local* cases yield lower costs than the *Edge* case because (1) the linearized MPC prediction is not accurate away from k; (2) the buffer size is not enough under severe packet loss.

Fig. 14 presents the costs of five cases under catastrophic wireless interferences (the edge-end network connection is disrupted from 5s to 50s). Similar to Fig. 9 (b) and Fig. 13 (b), the *Edge* case cannot handle a long period of packet drops since the buffer size is not large enough, while the *Smart* case mainly relies on the local controller.

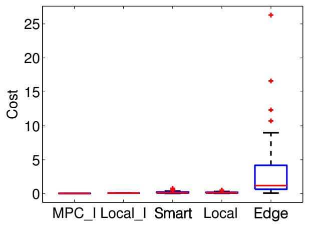


Fig. 14. 30% packet loss, and network connection is lost from 5s to 50s

D. Case Study of Reinforcement Learning as Edge Controller

We present a case study of Sec. V-A. Consider a double water-tank system, as shown in Fig. 15, containing one pump, two water tanks, and one basin. The system dynamics are

$$\begin{aligned}\dot{L}_1 &= A(\alpha u - D\sqrt{L_1})\\ \dot{L}_2 &= B(D\sqrt{L_1} - \theta E\sqrt{L_2})\\ \dot{L}_R &= C(\theta E\sqrt{L_2} - \alpha u)\end{aligned} \tag{14}$$

where control variable u represents the flow through the pump, and system parameters are given in Table I. The parameter of water tank may change over time, and specifically: $\theta = 0.95$ from 60 s to 180 s, and $\theta = 1.05$ from 180 s to 300 s. The control goal is to regulate L_2 at $L_{2ref} = 15\ m$.

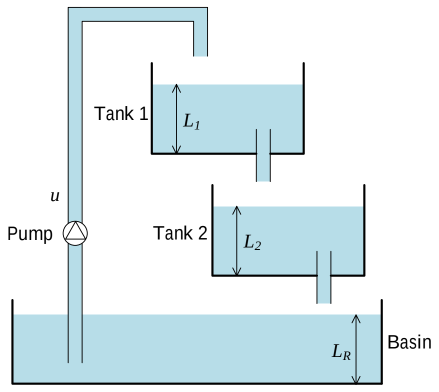


Fig. 15. Diagram of double water-tank system

1) Backstepping Parameter (θ) Estimator Design: Let $x = L_2 - L_{2ref}$, $\tilde{u} = BD\sqrt{L_1}$, $\phi(x) = -BE\sqrt{x + L_{2ref}}$. Therefore, we have:

$$\dot{x} = \theta\phi(x) + \tilde{u}. \tag{15}$$

To achieve regulation of $x(t)$, we employ adaptation. If θ was known, the control $\tilde{u} = -\theta\phi(x) - c_1 x, c_1 > 0$ would render the derivative of $V_0(x) = \frac{1}{2}x^2$ negative definite: $\dot{V}_0 = -c_1 x^2$. However, the control law $\tilde{u}$ cannot be applied since θ is unknown. Instead, θ can be replaced by its estimate θ_e,

$$\tilde{u} = -\theta_e\phi(x) - c_1 x. \tag{16}$$

Substituting (16) into (15), we obtain

$$\dot{x} = -c_1 x + \tilde{\theta}\phi(x). \tag{17}$$

where $\tilde{\theta}$ is the parameter estimation error: $\tilde{\theta} = \theta - \theta_e$. The derivative of $V_0(x) = \frac{1}{2}x^2$ becomes

$$\dot{V}_0 = -c_1 x^2 + \tilde{\theta}x\phi(x).$$

Since the second term is indefinite and contains the unknown parameter error $\tilde{\theta}$, we cannot prove the stability. We make the controller dynamic with an update law for θ_e, by augmenting V_0 with a quadratic term in the parameter error $\tilde{\theta}$, i.e.,

$$V_1(x, \tilde{\theta}) = \frac{1}{2}x^2 + \frac{1}{2\gamma}\tilde{\theta}^2,$$

where $\gamma > 0$ is the adaptation gain. The time derivative is

$$\dot{V}_1 = x\dot{x} + \frac{1}{\gamma}\tilde{\theta}\dot{\tilde{\theta}} = -c_1 x^2 + \tilde{\theta}\big(x\phi(x) + \frac{1}{\gamma}\dot{\tilde{\theta}}\big).$$

The second term is still indefinite but can be canceled by an appropriate choice of $\dot{\theta}_e$, specifically,

$$\begin{aligned}\dot{\theta}_e &= -\dot{\tilde{\theta}} = \gamma x\phi(x) \qquad (18)\\ &= -\gamma BE\sqrt{L_2}(L_2 - L_{2ref}). \qquad (19)\end{aligned}$$

which yields $\dot{V}_1 = -c_1 x^2 \leq 0$.

As shown in Fig. 16, in an ideal network, the parameter estimation law (19) ($\gamma = 0.0004$) performs well in estimating the unknown variable θ in (14).

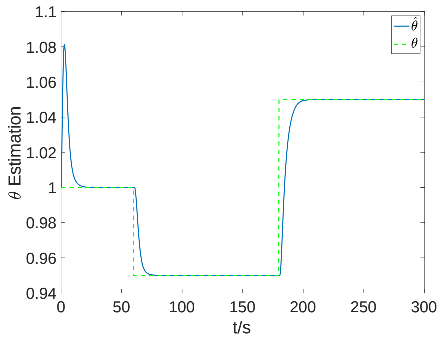


Fig. 16. Backstepping parameter (θ) estimation under ideal network

2) Reinforcement Learning Control Design: The stage cost is $l(x, u) = x^T Q x + u^T R u$, where $Q = 0.5I_3$ and $R = 1$. PI is done every 10 s online with 10 iterations. The basis vector stored in both local and edge controllers is $\Phi(x) = [x_1^2\ x_2^2\ x_1x_2\ x_1^4\ x_2^4\ x_1^2x_2^2]$. The control cost $V = \int_0^\infty\{x^T(t)Qx(t) + u^T(t)Ru(t)\}\mathrm{d}t$ decreases as the number of iterations increases and converges at around the sixth iteration, as shown in Fig. 17. Accordingly, W converges at $[11.02\ 48.85\ 12.72\ 0.01\ -0.01\ 0]^T$ from 0 s to 60 s, $[10.90\ 47.95\ 13.50\ 0.01\ -0.01\ 0]^T$ from 60 s to 180 s, $[11.08\ 49.27\ 11.98\ 0.01\ -0.01\ 0]^T$ from 180 s to 300 s.

3) *Simulation Results:* As shown in Fig. 18 (b), in an ideal network, with edge controller performing online adaptation via PI, L_2 could remain at 15 m most of the time. However, if there is only a local controller that does not support adaptation, L_2 cannot remain at the set point, as shown in Fig. 18 (a).

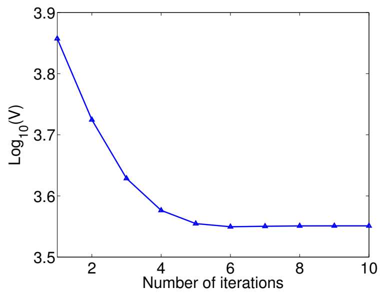
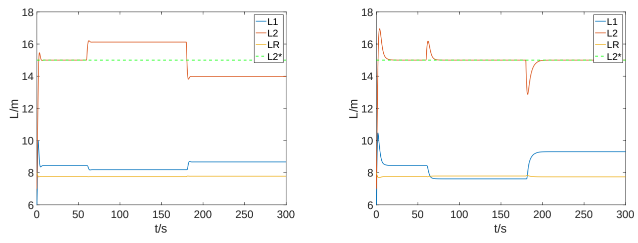


Fig. 17. Convergence of control cost as policy iteration increases

Boxplots are used to present the statistical results of control costs using different control strategies under an ideal network and lossy network. The simulation time of each round is 300 s, frequency is 4 Hz. Five groups of experiments are included in each boxplot figure, in the order from left to right:

1) *PI_I* is the control cost of *hybrid* system with edge controller and local controller in *ideal network*. Since there is no packet loss, only PI is active.
2) *Local_I* concludes system with *local controller only* in *ideal network*.
3) *Smart* concludes our smart actuation framework (*hybrid system*) in an unreliable network. The hybrid system adopts the switching policy given by (1).
4) *Local* is the control cost of the system with only *local controller* over an unreliable network. We assume that there is a signal flow from sensors to the local controller, represented by the dashed line in Fig. 7.
5) *Edge* is the control cost of the system with only edge *PI* controller over an unreliable network. Therefore, the last control input is adopted by the actuator when the actuation packet is lost.



(a) Local controller only (b) Smart actuation

Fig. 18. Separate local and edge controllers under ideal network

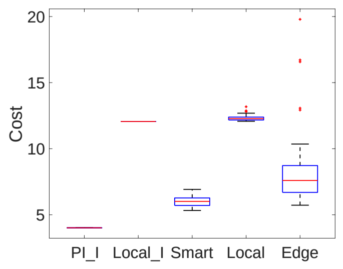


Fig. 19. Costs of double-tank system when there is 80% packet loss,$E = 0.95 * E_0$ from 60 s to 180 s, $E = 1.05 * E_0$ from 180 s to 300 s

Fig. 19 shows the statistical results with model parameter changes, where on-line PI as edge controller is essential. Smart actuation works better than edge controller when there is packet loss since the last control input is adopted by the actuator when the actuation packet is lost in edge controller only system. Local controller has the worst control performance since it does not have enough computational resources to do online policy iteration by itself.

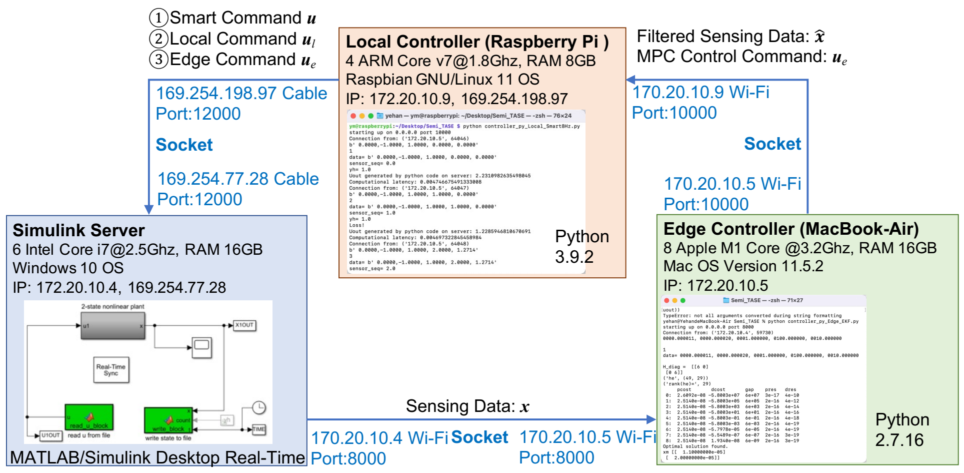
E. Semi-physical Simulation

It is extremely challenging to conduct real experiments in the industrial field, especially under cyber and physical disturbances. We built a semi-physical simulation platform, which integrates (1) real edge/local controllers running on various computation platforms; (2) real Wi-Fi network and Ethernet cable; and (3) MATLAB/Simulink Desktop Real-time (SLDRT), which simulates physical plants.

1) *System Design and Implementation:* The architecture of the semi-physical simulator is illustrated in Fig. 20. The system comprises three computers and two local area networks, Wi-Fi and Ethernet cable. The physical plants, sensors, and actuators are simulated in MATLAB/Simulink Desktop Real-time (SLDRT) running on a computer called the *Simulink Server*. The edge and local controller/smart actuator are implemented in Python on a MacBook and a Raspberry Pi, respectively. Socket interfaces are utilized to coordinate the wired and wireless communication among machines. The settings of the computation platforms and communication approaches that we use are described in Fig. 20.

The Simulink server operates the second-order nonlinear plant model described Sec. VI-C in SLDRT, which generates full state measurements x and then forwards them to the edge controller over real Wi-Fi network via Socket. The edge controller receives x and operates EKF and MPC, which generate state estimation $\hat{x}$ and control command u_e and then forwards them to Local controller over real Wi-Fi network via Socket. The local controller receives $\hat{x}$ and u_e and operates one of the three local control policies. We have implemented three local control policies for comparison: (1) smart actuation, which generates u; (2) local control only, which operates the control law in Sec. VI-C1, and generates u_l based on $\hat{x}$ without considering u_e; (3) pass control command from edge u_e directly. Then, the local control commands are transferred to Simulink server over back-to-back Ethernet cable via Socket. We implement the estimation and control policies in Python such that the policies can run on any platforms that support Python.

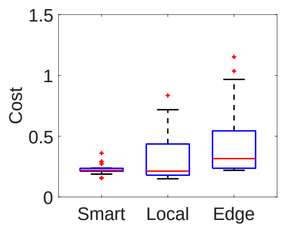
2) *Experimental Results:* For the rest of the experiments, the sampling rates are 5 Hz, 8 Hz, and 10 Hz, in order to explore the effects of sampling rates on smart actuation framework. The duration of each round of semi-physical simulation


①Smart Command $\boldsymbol{u}$
②Local Command $\boldsymbol{u}_l$
③Edge Command $\boldsymbol{u}_e$
169.254.198.97 Cable
Port:12000
Socket
169.254.77.28 Cable
Port:12000
Local Controller (Raspberry Pi)
4 ARM Core v7@1.8Ghz, RAM 8GB
Raspbian GNU/Linux 11 OS
IP: 172.20.10.9, 169.254.198.97
Python 3.9.2
Filtered Sensing Data: $\hat{\boldsymbol{x}}$
MPC Control Command: $\boldsymbol{u}_e$
170.20.10.9 Wi-Fi
Port:10000
Socket
170.20.10.5 Wi-Fi
Port:10000
Simulink Server
6 Intel Core i7@2.5Ghz, RAM 16GB
Windows 10 OS
IP: 172.20.10.4, 169.254.77.28
2-state nonlinear plant
Real-Time Sync
X1OUT
U1OUT
read u from file
write state to file
TIME
MATLAB/Simulink Desktop Real-Time
Sensing Data: $\boldsymbol{x}$
170.20.10.4 Wi-Fi
Port:8000
Socket
170.20.10.5 Wi-Fi
Port:8000
Edge Controller (MacBook-Air)
8 Apple M1 Core @3.2Ghz, RAM 16GB
Mac OS Version 11.5.2
IP: 172.20.10.5
Python 2.7.16

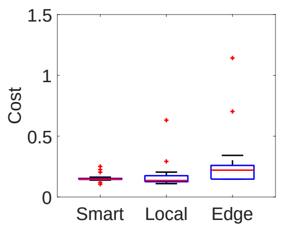
```
yehan — ym@raspberrypi: ~/Desktop/Semi_TASE — -zsh — 76×24
ym@raspberrypi:~/Desktop/Semi_TASE $ python controller_py_Local_Smart8Hz.py
starting up on 0.0.0.0 port 10000
Connection from: ('172.20.10.5', 64046)
b' 0.0000,-1.0000, 1.0000, 0.0000, 0.0000'
1
data= b' 0.0000,-1.0000, 1.0000, 0.0000, 0.0000'
sensor_seq= 0.0
yh= 1.0
Uout generated by python code on server: 2.2310982635498045
Computational latency: 0.004746675491333008
Connection from: ('172.20.10.5', 64047)
b' 0.0000,-1.0000, 1.0000, 1.0000, 0.0000'
2
data= b' 0.0000,-1.0000, 1.0000, 1.0000, 0.0000'
sensor_seq= 1.0
yh= 1.0
Loss!
Uout generated by python code on server: 1.2285946810670691
Computational latency: 0.004697322845458984
Connection from: ('172.20.10.5', 64048)
b' 0.0000,-1.0000, 1.0000, 2.0000, 1.2714'
3
data= b' 0.0000,-1.0000, 1.0000, 2.0000, 1.2714'
sensor_seq= 2.0
```

```
Semi_TASE — -zsh — 71×27
uout))
TypeError: not all arguments converted during string formatting
yehan@YehandeMacBook-Air Semi_TASE % python controller_py_Edge_EKF.py
starting up on 0.0.0.0 port 8000
Connection from: ('172.20.10.4', 59730)
0000.000011, 0000.000020, 0001.000000, 0100.000000, 0010.000000

1
data= 0000.000011, 0000.000020, 0001.000000, 0100.000000, 0010.000000

H_diag =  [[6 0]
 [0 6]]
('he', (49, 29))
('rank(he)=', 29)
     pcost       dcost       gap    pres   dres
 0:  2.6092e-08 -5.8003e+07  6e+07  3e-17  4e-10
 1:  2.5140e-08 -5.8003e+05  6e+05  2e-16  4e-12
 2:  2.5140e-08 -5.8003e+03  6e+03  2e-16  4e-14
 3:  2.5140e-08 -5.8003e+01  6e+01  2e-16  4e-16
 4:  2.5140e-08 -5.8003e-01  6e-01  2e-16  4e-18
 5:  2.5140e-08 -5.8003e-03  6e-03  2e-16  4e-19
 6:  2.5140e-08 -5.7978e-05  6e-05  2e-16  4e-19
 7:  2.5140e-08 -5.5489e-07  6e-07  2e-16  3e-19
 8:  2.5140e-08  1.9340e-08  6e-09  2e-16  3e-19
Optimal solution found.
xm [[  1.10000000e-05]
 [  2.00000000e-05]]
```

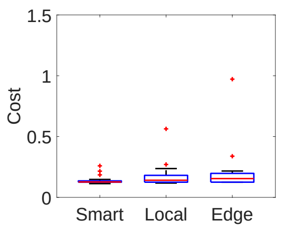
Fig. 20. Architecture for semi-physical simulations



(a) 5 Hz



(b) 8 Hz



(c) 10 Hz

Fig. 21. Costs of 2-state nonlinear system in semi-physical simulation under different sampling rates when there is 40% packet loss

is 30 s. In order to emulate packet losses in a controlled fashion, we intentionally drop packets on the receiver side following similar approaches employed in previous wireless control experiments [45], [58]. In Fig. 21, each sub-figure presents the costs of 3 different cases, *Smart*, *Local*, and *Edge*, for the second order nonlinear system. The simulation of each case is 20 rounds. When there is 40% packet loss, the *Smart* case outperforms the *Local* and *Edge* cases in the sense that the cost distribution of the *Smart* has a lower mean value and smaller variance than the *Local* and *Edge* cases. Comparing the performance of various sampling rates, at all three sampling rates, the control costs are decreased as the sampling rates increases at the costs of communication and computation. *Smart* performs well even at 5 Hz, at which *Local* and *Edge* cannot provide operable control performance.

VII. Conclusions and Future Work

We proposed smart actuation strategies for end-edge industrial automation. The *smart actuation* architecture combines features of direct and hierarchical architectures: an edge controller accounts for optimality, uncertainties, and constraints by executing computationally expensive operations; a smart actuator executes a local control policy and accounts for system safety in view of network imperfections. It can benefit from fast and reliable communication on the end side, and powerful computation capacity on the edge side. We have proposed the end-edge co-design strategies and cooperation logic for both performance and stability. Stability for linear and nonlinear plant cases can be guaranteed by co-design procedures when the edge controller employs MPC and PI-based learning. Adaptability to parameter mismatches can be provided when the edge controller employs reinforcement learning. Simulation results show that the smart actuation strategies work well in both reliable and unreliable networks. Many interesting issues remain open, for instance, if the smart actuation can outperform other architectures when communication and computation latencies are considered, what if the plant models are partially known, and how to extend these results to large-scale end-edge industrial automation systems including multiple control loops with various sampling rates.

Appendix

Model-based Policy Iteration for systems in discrete-time is included below for completeness. Interested readers are referred to [52] for more details. The notion of goal-directed optimal behavior is captured by defining the cost function

$$V\big(x(k)\big) = \sum_{i=k}^{\infty} \gamma^{i-k} l\Big(x(i), h\big(x(i)\big)\Big), \tag{20}$$

with $0 < \gamma \leq 1$ a discount factor, and $l\Big(x(i), h\big(x(i)\big)\Big)$ the stage cost, which measures the one-step cost of control. PI for system (3) begins with an initialization step, where a stabilizing control policy $h_0\big(x(k)\big)$ is designed; then repeats the following two steps until convergence, for $0 \leq j < \infty$,

1) Policy evaluation: solving for $V_{j+1}(x)$ satisfying

$$V_{j+1}\big(x(k)\big) = l\Big(x(k), h_j\big(x(k)\big)\Big) + \gamma V_{j+1}\big(x(k+1)\big); \tag{21}$$

2) Policy improvement: updating control policy as follows

$$\begin{aligned} h_{j+1}\big(x(k)\big) =& \arg\min_{h(.)} \big(l\big(x(k), h_j(x(k))\big) \\ &+ \gamma V_{j+1}\big(x(k+1)\big)\big). \end{aligned} \tag{22}$$

Policy evaluation (21) involves solving an infinite dimension problem. A common treatment resorts to approximating the value function and control policy [57], i.e., $V(x) = W^T\Phi(x), u(x) = \Gamma^T\Psi(x)$, where W and Γ are the coefficient vectors, and $\Phi(x) = [\phi_1(x)\ \ \phi_2(x)\ \ ...\phi_N(x)]^T, \Psi(x) = [\psi_1(x)\ \ \psi_2(x)\ \ ...\psi_q(x)]^T$ are the basis vectors. The policy evaluation formula is given by

$$W^T\Big(\Phi\big(x(k)\big) - \Phi\big(x(k+1)\big)\Big) = l\Big(x(k), h\big(x(k)\big)\Big). \tag{23}$$

If $l(x,u) = x^TQx + u^TRu$ with Q and R being positive definite, the policy improvement has a closed-form solution:

$$h\big(x(k)\big) = -\frac{1}{2}R^{-1}g^T\big(x(k)\big)\frac{\partial\Phi^T\big(x(k+1)\big)}{\partial x}W. \tag{24}$$

References

[1] X. Li, D. Li, J. Wan, A. V. Vasilakos, C.-F. Lai, and S. Wang, “A review of industrial wireless networks in the context of industry 4.0,” *Wireless Networks*, vol. 23, no. 1, pp. 23–41, 2017.

[2] A. Ahlén, J. Akerberg, M. Eriksson, A. J. Isaksson, T. Iwaki, K. H. Johansson, S. Knorn, T. Lindh, and H. Sandberg, “Toward wireless control in industrial process automation: A case study at a paper mill,” *IEEE Control Systems Magazine*, vol. 39, no. 5, pp. 36–57, 2019.

[3] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu, “Wireless routing and control: a cyber-physical case study,” in *Proceedings of the 7th International Conference on Cyber-Physical Systems*. IEEE Press, 2016, p. 32.

[4] ISA100-WIRELESS, “ISA100: Wireless Systems for Automation,” http://www.isa100wci.org, 2022.

[5] HART Communication Foundation, “WirelessHART Specification,” https://www.fieldcommgroup.org/technologies/wirelesshart, 2022.

[6] H. Grichi, O. Mosbahi, M. Khalgui, and Z. Li, “Rwin: New methodology for the development of reconfigurable wsn,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 109–125, 2016.

[7] F. Zhu, C. Zhang, Z. Zheng, and A. Farouk, “Practical network coding technologies and softwarization in wireless networks,” *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5211–5218, 2021.

[8] D. Gunatilaka and C. Lu, “React: an agile control plane for industrial wireless sensor-actuator networks,” in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 53–65.

[9] M. Sha, D. Gunatilaka, C. Wu, and C. Lu, “Empirical study and enhancements of industrial wireless sensor-actuator network protocols,” *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 696–704, June, 2017.

[10] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, “Kalman filtering with intermittent observations,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004.

[11] D. Zhang, Z. Zhou, and X. Jia, “Network-based pi control for output tracking of continuous-time systems with time-varying sampling and network-induced delays,” *Journal of the Franklin Institute*, vol. 355, no. 12, pp. 4794–4808, 2018.

[12] T. H. Truong, P. Seiler, and L. E. Linderman, “Analysis of networked structural control with packet loss,” *IEEE Transactions on Control Systems Technology*, vol. 30, no. 1, pp. 344–351, 2021.

[13] Y. Yuan, H. Yuan, D. W. Ho, and L. Guo, “Resilient control of wireless networked control system under denial-of-service attacks: A cross-layer design approach,” *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 48–60, 2018.

[14] K. Halder, S. Das, D. K. Panda, S. Das, and A. Gupta, “QoS aware joint observer and networked PI/PID controller design using LMIs under specified rate of packet dropouts,” *Applied Mathematics and Computation*, vol. 401, p. 126125, 2021.

[15] D. Kim, Y. Won, S. Kim, Y. Eun, K.-J. Park, and K. H. Johansson, “Sampling rate optimization for ieee 802.11 wireless control systems,” in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 87–96.

[16] L. Zhou and P. Tokekar, “Active target tracking with self-triggered communications in multi-robot teams,” *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1085–1096, 2018.

[17] K. Gatsis, A. Ribeiro, and G. J. Pappas, “Random access design for wireless control systems,” *Automatica*, vol. 91, pp. 1–9, 2018.

[18] Y. Ma, J. Guo, Y. Wang, A. Chakrabarty, H. Ahn, P. Orlik, X. Guan, and C. Lu, “Optimal dynamic transmission scheduling for wireless networked control systems,” *IEEE Transactions on Control Systems Technology*, 2022.

[19] K. Gatsis, A. Ribeiro, and G. J. Pappas, “Optimal power management in wireless control systems,” *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1495–1510, 2014.

[20] 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.

[21] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, “Wireless network design for control systems: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 978–1013, 2017.

[22] Y. Tipsuwan and M.-Y. Chow, “Control methodologies in networked control systems,” *Control Engineering Practice*, vol. 11, no. 10, pp. 1099–1111, 2003.

[23] S. Di Cairano, U. V. Kalabić, and I. V. Kolmanovsky, “Reference governor for network control systems subject to variable time-delay,” *Automatica*, vol. 62, pp. 77–86, 2015.

[24] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Mobile-edge computing: Partial computation offloading using dynamic voltage scaling,” *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[25] C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, “Videoedge: Processing camera streams using hierarchical clusters,” in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 115–131.

[26] M. Cui, S. Zhong, B. Li, X. Chen, and K. Huang, “Offloading autonomous driving services via edge computing,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 535–10 547, 2020.

[27] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, “A survey on computation offloading modeling for edge computing,” *Journal of Network and Computer Applications*, vol. 169, p. 102781, 2020.

[28] N. Vreman and M. Maggio, “Multilayer distributed control over 5g networks: Challenges and security threats,” in *Proceedings of the Workshop on Fog Computing and the IoT*, 2019, pp. 31–35.

[29] P. Skarin, W. Tärneberg, K.-E. Årzen, and M. Kihl, “Towards mission-critical control at the edge and over 5g,” in *EDGE*. IEEE, 2018, pp. 50–57.

[30] P. Skarin, J. Eker, M. Kihl, and K.-E. Årzén, “Cloud-assisted model predictive control,” in *EDGE*. IEEE, 2019, pp. 110–112.

[31] Y. Ma, C. Lu, B. Sinopoli, and S. Zeng, “Exploring edge computing for multitier industrial control,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3506–3518, 2020.

[32] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, “A survey of recent results in networked control systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[33] C. Ju and H. I. Son, “A hybrid systems-based hierarchical control architecture for heterogeneous field robot teams,” *IEEE Transactions on Cybernetics*, 2021.

[34] B. Fateh, M. Govindarasu, and V. Ajjarapu, “Wireless network design for transmission line monitoring in smart grid,” *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 1076–1086, 2013.

[35] C. L. Robinson and P. Kumar, “Optimizing controller location in networked control systems with packet drops,” *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, 2008.

[36] M. A. Saez, F. P. Maturana, K. Barton, and D. M. Tilbury, “Context-sensitive modeling and analysis of cyber-physical manufacturing systems for anomaly detection and diagnosis,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 29–40, 2019.

[37] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, and N. Meskin, “Cybersecurity for industrial control systems: A survey,” *Computers & Security*, vol. 89, p. 101677, 2020.

[38] L. Hu, Z. Wang, Q.-L. Han, and X. Liu, "State estimation under false data injection attacks: Security analysis and system protection," *Automatica*, vol. 87, pp. 176–183, 2018.

[39] R. Candell, T. Zimmerman, K. Stouffer *et al.*, "An industrial control system cybersecurity performance testbed," *National Institute of Standards and Technology. NISTIR*, vol. 8089, 2015.

[40] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: a survey of recent results," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 308–322, 2009.

[41] 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.

[42] X. Liu and A. Goldsmith, "Kalman filtering with partial observation losses," in *2004 43rd IEEE Conference on Decision and Control (CDC)*, vol. 4. IEEE, 2004, pp. 4180–4186.

[43] Y. Shi and H. Fang, "Kalman filter-based identification for systems with randomly missing measurements in a network environment," *International Journal of Control*, vol. 83, no. 3, pp. 538–551, 2010.

[44] V. Stehel, C. Bradley, P. Suler, and S. Bilan, "Cyber-physical system-based real-time monitoring, industrial big data analytics, and smart factory performance in sustainable manufacturing internet of things," *Economics, Management, and Financial Markets*, vol. 16, no. 1, pp. 42–51, 2021.

[45] 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 *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 97–108.

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

[47] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[48] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[49] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.

[50] H. Fukushima, T.-H. Kim, and T. Sugie, "Adaptive model predictive control for a class of constrained linear systems based on the comparison model," *Automatica*, vol. 43, no. 2, pp. 301–308, 2007.

[51] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning systems*, vol. 29, no. 6, pp. 2042–2062, 2017.

[52] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 40–58, 2009.

[53] Y. Wang, Y. Zhao, S. A. Bortoff, and K. Ueda, "A real-time energy-optimal trajectory generation method for a servomotor system," vol. 62, no. 2, pp. 1175–1188, Feb. 2015.

[54] Y. Wang, H. Fang, L. Zhou, and T. Wada, "Revisiting the state-of-charge estimation for lithium-ion batteries: A methodical investigation of the extended kalman filter approach," *IEEE Control Systems Magazine*, vol. 37, no. 4, pp. 73–96, 2017.

[55] M. Krstić, I. Kanellakopoulos, and P. Kokotović, *Nonlinear and Adaptive Control Design*. New York, NY: John Wiley and Sons, 1995.

[56] V. Shilpiekandula, S. A. Bortoff, J. C. Barnwell, and K. El Rifai, "Load positioning in the presence of base vibrations," in *American Control Conference*, 2012, pp. 6282–6287.

[57] Y. Jiang, Y. Wang, S. A. Bortoff, and Z.-P. Jiang, "Optimal codesign of nonlinear control systems based on a modified policy iteration method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 409–414, 2015.

[58] D. Baumann, F. Mager, R. Jacob, L. Thiele, M. Zimmerling, and S. Trimpe, "Fast feedback control over multi-hop wireless networks with mode changes and stability guarantees," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 2, pp. 1–32, 2019.