Feedback Control Goes Wireless: Guaranteed Stability over Low-power Multi-hop Networks

Fabian Mager*
TU Dresden
fabian.mager@tu-dresden.de

Dominik Baumann*
MPI for Intelligent Systems
dominik.baumann@tuebingen.mpg.de

Romain Jacob
ETH Zurich
romain.jacob@tik.ee.ethz.ch

Lothar Thiele
ETH Zurich
thiele@ethz.ch

Sebastian Trimpe
MPI for Intelligent Systems
trimpe@is.mpg.de

Marco Zimmerling
TU Dresden
marco.zimmerling@tu-dresden.de

ABSTRACT

Closing feedback loops fast and over long distances is key to emerging applications; for example, robot motion control and swarm coordination require update intervals of tens of milliseconds. Low-power wireless technology is preferred for its low cost, small form factor, and flexibility, especially if the devices support multi-hop communication. So far, however, feedback control over wireless multi-hop networks has only been shown for update intervals on the order of seconds. This paper presents a wireless embedded system that tames imperfections impairing control performance (*e.g.*, jitter and message loss), and a control design that exploits the essential properties of this system to provably guarantee closed-loop stability for physical processes with linear time-invariant dynamics. Using experiments on a cyber-physical testbed with 20 wireless nodes and multiple cart-pole systems, we are the first to demonstrate and evaluate feedback control and coordination over wireless multi-hop networks for update intervals of 20 to 50 milliseconds.

CCS CONCEPTS

• **Computer systems organization** → **Sensors and actuators**; *Embedded systems*; *Real-time system architecture*; *Dependable and fault-tolerant systems and networks*; • **Networks** → **Cyber-physical networks**; *Network protocol design.*

KEYWORDS

Wireless control, Closed-loop stability, Multi-agent systems, Multi-hop networks, Synchronous transmissions, Cyber-physical systems

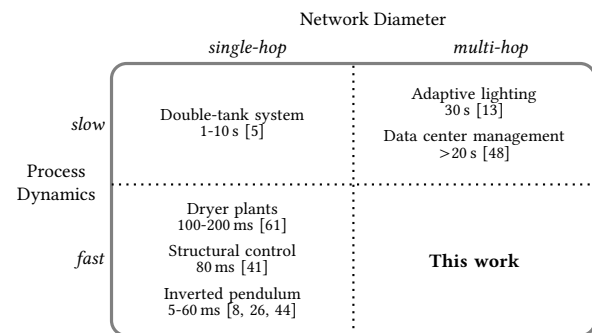*Both authors contributed equally to this work.

Figure 1: Design space of wireless CPS that have been validated on physical platforms and real wireless networks.

1 INTRODUCTION

Cyber-physical systems (CPS) use embedded computers and networks to monitor and control physical systems [17]. While monitoring using *sensors* allows, for example, to better understand certain characteristics of environmental processes [15], it is control and coordination through *actuators* what nurtures the CPS vision exemplified by robotic materials [16], smart transportation [10], and multi-robot swarms for disaster response and manufacturing [25].

A key hurdle to realizing this vision is how to close the *feedback loops* between sensors and actuators as these may be numerous, mobile, distributed over large spaces, and attached to devices with size, weight, and cost constraints. Wireless multi-hop communication among low-power, possibly battery-powered devices[1] provides the cost efficiency and flexibility to overcome this hurdle [36, 57] if two requirements are met. First, fast feedback is needed to keep up with the dynamics of physical systems [7]; for example, robot motion control and drone swarm coordination require update intervals of tens of milliseconds [1, 45]. Second, as feedback control modifies the dynamics of physical systems [6], guaranteeing *closed-loop stability* under imperfect wireless communication is a major concern.

Hence, this paper investigates the following question: *Is it possible to enable fast feedback control and coordination across **real-world** multi-hop low-power wireless networks with formal guarantees on closed-loop stability?* Prior works on control over wireless that validate their design through experiments on physical platforms do not provide an affirmative answer. As shown in Figure 1 and detailed in

---

[1]While actuators may need wall power, low-power operation is crucial for sensors and controllers, which may run on batteries and harvest energy from the environment [2].

Section 2, solutions based on *multi-hop* communication have only been demonstrated for physical systems with *slow* dynamics (*i.e.*, update intervals of seconds) and do not provide stability guarantees. Practical solutions with stability guarantees for *fast* process dynamics (*i.e.*, update intervals of tens of milliseconds as typical of, *e.g.*, mechanical systems) exist, but these are only applicable to *single-hop* networks and therefore lack the scalability and flexibility required by many future CPS applications [25, 40].

Contribution and road-map. This paper presents the design, analysis, and practical validation of a wireless CPS that fills this gap. Section 3 highlights the main challenges and corresponding system design goals we must achieve when closing feedback loops fast over wireless multi-hop networks. Our approach is based on a careful co-design of the wireless embedded components (in terms of hardware and software) and the closed-loop control system, as described in Sections 4 and 5. More concretely, we tame typical wireless network imperfections, such as message loss and jitter, so that they can be tackled by well-known control techniques or safely neglected. As a result, our solution is amenable to a formal end-to-end analysis of all CPS components (*i.e.*, wireless embedded, control, and physical systems), which we exploit to provably guarantee closed-loop stability for physical systems with *linear time-invariant (LTI)* dynamics. Moreover, unlike prior work, our solution supports control and coordination of multiple physical systems out of the box, which is a key asset in many CPS applications [1, 25, 45].

To evaluate our design in Section 6, we have developed a cyber-physical testbed that consists of 20 wireless embedded devices forming a three-hop network and multiple cart-pole systems whose dynamics match a range of real-world mechanical systems [6, 55]. As such, our testbed addresses an important need in CPS research [36]. Our experiments reveal the following key findings: (*i*) two inverted pendulums can be safely stabilized by two remote controllers across the three-hop wireless network; (*ii*) the movement of five cart-poles can be synchronized reliably over the network; (*iii*) increasing message loss rates and update intervals can be tolerated at reduced control performance; and (*iv*) the experiments confirm our analyses.

In summary, this paper contributes the following:

- We are the first to demonstrate feedback control and coordination across real multi-hop low-power wireless networks at update intervals of 20 to 50 milliseconds.
- We formally prove that our end-to-end CPS design guarantees closed-loop stability for physical systems with LTI dynamics.
- Experiments on a novel cyber-physical testbed show that our solution can stabilize and synchronize multiple inverted pendulums despite significant message loss.

2 RELATED WORK

Feedback control over wireless communication networks has been extensively studied. For instance, the control community has investigated control design and stability analysis for wireless (and wired) networks based on different system architectures, delay models, and message loss processes (see, *e.g.*, [3, 22, 37, 46, 50, 51, 56, 59, 63]); recent surveys provide an overview of this body of fundamental research [27, 62]. However, the majority of those works focuses on theoretical analyses or validates new wireless CPS designs (*e.g.*,

based on WirelessHART [35, 42]) only in simulation, thereby ignoring many fundamental challenges that may complicate or prevent a real implementation [36]. One of the challenges, as detailed in Section 3, is that even slight variations in the quality of a wireless link can trigger drastic changes in the routing topology [13]—and this can happen several times per minute [23]. Hence, to establish trust in feedback control over wireless, a real-world validation against these *dynamics* on a realistic CPS testbed is absolutely essential [36], as opposed to considering setups with a *statically configured* routing topology and only a few nodes on a desk as, for example, in [49].

Figure 1 classifies prior control-over-wireless solutions that have been validated using experiments on physical platforms and against the dynamics of real wireless networks along two dimensions: the diameter of the network (*single-hop* or *multi-hop*) and the dynamics of the physical system (*slow* or *fast*). While not representing absolute categories, we use slow to refer to update intervals on the order of seconds, which is typically insufficient for feedback control of, for example, mechanical systems.

In the *single-hop/slow* category, Araújo et al. [5] investigate resource efficiency of aperiodic control with closed-loop stability in a single-hop wireless network of IEEE 802.15.4 devices. Using a double-tank system as the physical process, update intervals of one to ten seconds are sufficient.

A number of works in the *single-hop/fast* class stabilize an inverted pendulum via a controller that communicates with a sensor-actuator node at the cart. The update interval is 60 ms or less, and the interplay of control and network performance, as well as closed-loop stability are investigated for different wireless technologies: Bluetooth [19], IEEE 802.11 [44], and IEEE 802.15.4 [8, 26]. Belonging to the same class, Ye et al. use three IEEE 802.11 nodes to control two dryer plants at update intervals of 100–200 ms [61], and Lynch et al. use four proprietary wireless nodes to demonstrate control of a three-story test structure at an update interval of 80 ms [41].

For *multi-hop* networks, there are only solutions for *slow* process dynamics and without stability analysis. For example, Ceriotti et al. study adaptive lighting in road tunnels [13]. Owing to the length of the tunnels, multi-hop communication becomes unavoidable, yet the required update interval of 30 seconds allows for a reliable solution built out of mainstream sensor network technology. Similarly, Saifullah et al. present a multi-hop solution for power management in data centers, using update intervals of 20 seconds or greater [48].

In contrast to these works, as illustrated in Figure 1, we demonstrate *fast* feedback control over wireless *multi-hop* networks at update intervals of 20–50 ms, which is significantly faster than existing multi-hop solutions. Moreover, we provide a formal stability proof, and our solution seamlessly supports both control and coordination of multiple physical systems, which we validate through experiments on a real-world cyber-physical testbed.

3 PROBLEM SETTING AND APPROACH

Scenario. We consider wireless CPS that consist of a set of embedded devices equipped with low-power wireless radios. The devices execute different *application tasks* (*i.e.*, sensing, control, and actuation) that exchange *messages* over a wireless multi-hop network. Each node may execute multiple application tasks, which may belong to different distributed feedback loops. As an example, Figure 2
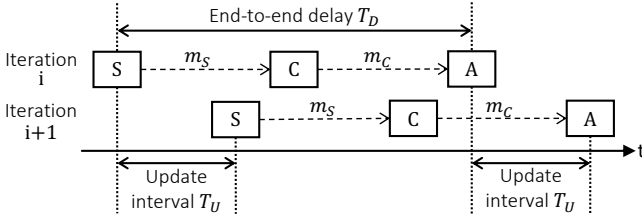
Figure 2: Application tasks and message transfers of a single feedback loop. *In every iteration, the sensing task (S) takes a measurement of the physical system and sends it to the control task (C), which computes a control signal and sends it to the actuation task (A).*

shows the execution of application tasks and the exchange of messages for a single periodic feedback loop with one sensor and one actuator. The *update interval* T_U is the time between consecutive sensing or actuation tasks. The *end-to-end delay* T_D is the time between corresponding sensing and actuation tasks.

Challenges. Fast feedback control over wireless multi-hop networks is an open problem due to the following challenges:

- *Lower end-to-end throughput.* Multi-hop networks have a lower end-to-end throughput than single-hop networks because of interference: the theoretical multi-hop upper bound is half the single-hop upper bound [43]. This limits the number of sensors and actuators that can be supported for a given maximum update interval.
- *Significant delays and jitter.* Multi-hop networks also incur longer end-to-end delays, and the delays are subject to larger variations because of retransmissions or routing dynamics [13], introducing significant jitter. Delays and jitter can both destabilize a feedback system [56, 58].
- *Constrained traffic patterns.* In a single-hop network, each node can communicate with every other node due to the broadcast property of the wireless medium. This is generally not the case in a multi-hop network. For example, WirelessHART only supports communication to and from a gateway that connects the wireless network to the control system. Feedback control under constrained traffic patterns is more challenging and may imply poor control performance or even infeasibility of closed-loop stability [60].
- *Correlated message loss.* Message loss is a common phenomenon in wireless networks, which complicates control design. Further, because there is often significant correlation among the message losses [53], a valid theoretical analysis to provide strong guarantees is hard, if not impossible.
- *Message duplicates and out-of-order message delivery* are typical in wireless multi-hop protocols [18, 23] and may further hinder control design and stability analysis [62].

Approach. We adopt the following co-design approach to enable fast feedback control over wireless multi-hop networks: *Address the challenges on the wireless embedded system side to the extent possible, and then consider the resulting key properties in the control design.* This entails the design of a wireless embedded system that aims to:

- **G1** reduce and bound imperfections impairing control performance (*e.g.*, reduce T_U and T_D and bound their jitter);
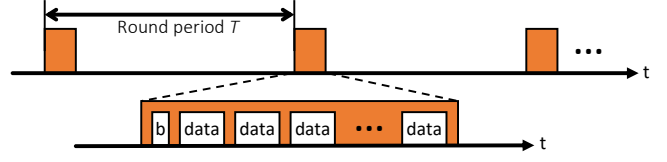

Figure 3: Time-triggered operation of multi-hop low-power wireless protocol. *Communication occurs in rounds that are scheduled with a given round period T. Every beacon (b) and data slot in a round corresponds to an efficient, reliable one-to-all Glossy flood [21].*

- **G2** support arbitrary traffic patterns in multi-hop networks with real dynamics (*e.g.*, time-varying link qualities);
- **G3** operate efficiently in terms of limited resources, while accommodating the computational needs of the controller.

On the other hand, the control design aims to:

- **G4** incorporate all essential properties of the wireless embedded system to guarantee closed-loop stability for the entire CPS for physical systems with LTI dynamics;
- **G5** enable an efficient implementation of the control logic on state-of-the-art low-power embedded devices;
- **G6** exploit the support for arbitrary traffic patterns for straightforward distributed control and multi-agent coordination.

4 WIRELESS EMBEDDED SYSTEM DESIGN

To achieve goals **G1**–**G3**, we design a wireless embedded system that consists of three key building blocks:

1) a *low-power wireless protocol* that provides multi-hop many-to-all communication with minimal, bounded end-to-end delay and accurate network-wide time synchronization;
2) a *hardware platform* that enables a predictable and efficient execution of all application tasks and message transfers;
3) a *scheduling framework* to schedule all application tasks and message transfers so that given bounds on T_U and T_D are met at minimum energy costs for wireless communication.

We describe each building block below, followed by an analysis of the resulting properties that matter for the control design.

4.1 Low-power Wireless Protocol

To support arbitrary traffic patterns (**G2**), we require a multi-hop protocol capable of many-to-all communication. Moreover, the protocol must be highly reliable and the time needed for many-to-all communication must be tightly bounded (**G1**). It has been shown that a solution based on Glossy floods [21] can meet these requirements with high efficiency (**G3**) in the face of wireless dynamics (**G2**) [65]. Thus, similar to other recent proposals [20, 29], we design a wireless protocol on top of Glossy, but aim at a new design point: bounded end-to-end delays of at most a few tens of milliseconds *for the many-to-all exchange of multiple messages* in a control cycle.

As shown in Figure 3, the operation of the protocol proceeds as a series of periodic *communication rounds* with *period T*. Each round consists of a sequence of non-overlapping time *slots*. In every time slot, all nodes in the network participate in a Glossy flood, where a message is sent from one node to all other nodes. Glossy approaches
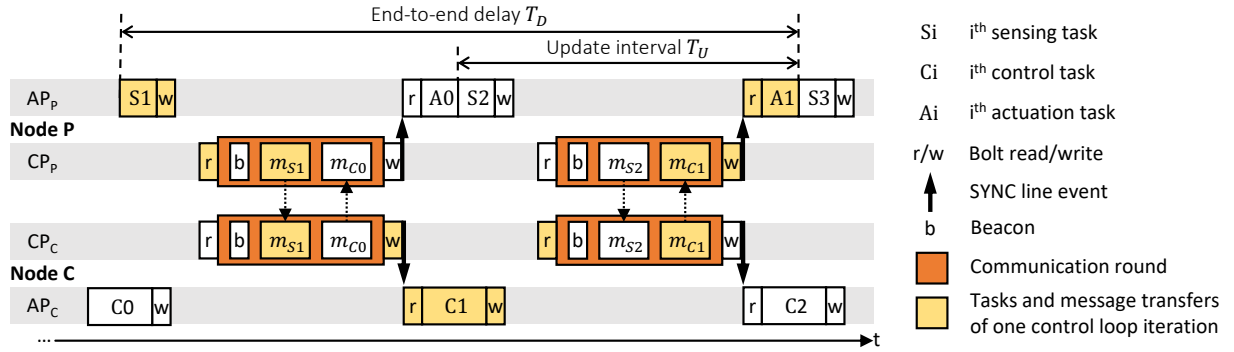
Figure 4: Example schedule of application tasks and message transfers between two dual-processor platforms. *Node P senses and acts on a physical system, while node C runs the controller. In this example, the update interval T_U is half the end-to-end delay T_D.*

the theoretical minimum latency for one-to-all flooding at a reliability above 99.9 %, and provides microsecond-level network-wide time synchronization [21]. In particular, a flood initiated by a dedicated node in the *beacon* slot (b) at the beginning of every round is used for synchronization. Nodes exploit the synchronization to remain in a low-power sleep mode between rounds and to awake in time for the next round, as specified by the round period T.

It is important to note that, because of the way Glossy exploits *synchronous transmissions* [21], the wireless protocol operates *independently* of the time-varying network topology. In particular, this means that the wireless protocol and any logic built on top of it, such as a control algorithm, need not worry about the state of individual wireless links in the network. This is a key difference to existing wireless protocols based on routing, such as WirelessHART and 6TiSCH, which, as we shall see, simplifies the control design and allows for providing formal guarantees that also hold in practice.

As detailed in Section 4.3, we compute the communication schedule offline based on the traffic demands, and distribute it to all nodes before the application operation starts. A schedule includes the assignment of messages to *data* slots in each round (see Figure 3) and the round period T. Using static schedules brings several benefits. First, we can a priori verify if closed-loop stability can be guaranteed for the achievable latencies (see Section 5). Second, compared to prior solutions [20, 29, 31, 65], we can support significantly shorter latencies, and the protocol is more energy efficient (no need to send schedules) and more reliable (schedules cannot be lost).

4.2 Hardware Platform

CPS devices need to concurrently handle application tasks and message transfers. While message transfers involve little but frequent computations, sensing and especially control tasks may require less frequent, but more demanding computations (*e.g.*, floating-point operations). An effective approach to achieve low latency and high energy efficiency for such diverse workloads is to exploit hardware heterogeneity (**G3**).

For this reason, we leverage a heterogeneous *dual-processor platform (DPP)*. Application tasks execute exclusively on a 32-bit MSP432P401R ARM Cortex-M4F *application processor* (AP) running at 48 MHz, while the wireless protocol executes on a dedicated 16-bit CC430F5147 *communication processor* (CP) running at 13 MHz.

The AP has a floating-point unit and a rich instruction set, accelerating computations related to sensing and control. The CP features a low-power microcontroller and a low-power wireless radio operating at 250 kbit/s in the 868 MHz frequency band.

AP and CP are interconnected using Bolt [54], an ultra-low-power processor interconnect that supports asynchronous bidirectional message passing with formally verified worst-case execution times. Bolt decouples the two processors with respect to time, power, and clock domains, enabling energy-efficient concurrent executions with only small and bounded interference, thereby limiting jitter and preserving the time-sensitive operation of the wireless protocol.

All CPs are time-synchronized via the wireless protocol. In addition, AP and CP must be synchronized locally on each DPP to minimize end-to-end delay and jitter among application tasks running on different APs (**G1**). To this end, we use a GPIO line between the two processors, called *SYNC* line. Every CP asserts the SYNC line in response to an update of Glossy's time synchronization. Every AP schedules application tasks and message passing over Bolt with specific offsets relative to these SYNC line events and resynchronizes its local time base. Likewise, the CPs execute the communication schedule and perform SYNC line assertion and message passing over Bolt with specific offsets relative to the start of communication rounds. As a result, all APs and CPs act in concert.

4.3 Scheduling Framework

We illustrate the scheduling problem we need to solve with a simple example, where node P senses and acts on a physical system and node C runs the controller.

Figure 4 shows a possible schedule of the application tasks and message transfers. After sensing (S1), AP_P writes a message containing the sensor reading into Bolt (w). CP_P reads out the message (r) before the communication round in which that message (m_{S1}) is sent using the wireless protocol. CP_C receives the message and writes it into Bolt. After reading out the message from Bolt, AP_C computes the control signal (C1) and writes a message containing it into Bolt. The message (m_{C1}) is sent to CP_P in the next round, and then AP_P applies the control signal on the physical system (A1).

This schedule resembles a pipelined execution, where in each communication round the last sensor reading and the next control signal (computed based on the previous sensor reading) are

exchanged ($m_{S1}\,m_{C0}$, $m_{S2}\,m_{C1}$, ...). Note that while it is indeed possible to send the corresponding control signal in the same round ($m_{S1}\,m_{C1}$, $m_{S2}\,m_{C2}$, ...), doing so would increase the update interval T_U at least by the sum of the execution times of the control task, Bolt read, and Bolt write. For the example schedule in Figure 4, the update interval T_U is exactly half the end-to-end delay T_D.

In general, the scheduling problem entails computing the communication schedule and the offsets with which all APs and CPs in the system perform wireless communication, execute application tasks, transfer messages over Bolt, and assert the SYNC line. The problem gets extremely complex for any realistic scenario with more nodes or multiple feedback loops that are closed over the same wireless network, so solving it must be automated.

To this end, we leverage Time-Triggered Wireless (TTW) [30], an existing framework tailored to solve this type of scheduling problem. TTW takes as main input a dependency graph among application tasks and messages, similar to Figure 2. Based on an integer linear program, it computes the communication schedule and all offsets mentioned above. TTW provides three important guarantees: (*i*) a feasible solution is found if one exists, (*ii*) the solution minimizes the energy consumption for wireless communication, and (*iii*) the solution can additionally optimize user-defined metrics (*e.g.*, minimize the update interval T_U as for the schedule in Figure 4).

4.4 Essential Properties and Jitter Analysis

Essential properties. The presented wireless embedded system design provides the following key properties for the control design:

P1 As analyzed below, for update intervals T_U and end-to-end delays T_D up to 100 ms, the worst-case jitter on T_U and T_D is bounded by ±50 µs. It holds $T_D = 2T_U$.

P2 Statistical analysis of millions of Glossy floods [64] and percolation theory for time-varying networks [32] have shown that the spatio-temporal diversity in a Glossy flood reduces the temporal correlation in the series of received and lost messages by a node, to the extent that the series can be safely approximated by an i.i.d. Bernoulli process. The success probability in real multi-hop networks is typically larger than 99.9 % [21].

P3 By provisioning for multi-hop many-to-all communication, arbitrary traffic patterns are efficiently supported.

P4 It is guaranteed by design that message duplicates and out-of-order message deliveries do not occur.

Worst-case jitter analysis. To underpin **P1**, we analyze the *worst-case* jitter on T_U and T_D. We refer to $\widetilde{T}_{end}$ as the nominal time interval between the end of two tasks executed on (possibly) different APs. Due to jitter J, this interval may vary, resulting in an actual length of $\widetilde{T}_{end} + J$. In our system, the jitter is bounded by

$$|J| \le 2\left(\hat{e}_{ref} + \hat{e}_{SYNC} + \widetilde{T}_{end}(\hat{\rho}_{AP} + \hat{\rho}_{CP})\right) + \hat{e}_{task} \qquad (1)$$

where each term on the right-hand side of (1) is detailed below.

1) Time synchronization error between CPs. Using Glossy, each CP computes an estimate $\hat{t}_{ref}$ of the reference time [21] to schedule subsequent activities. In doing so, each CP makes an error e_{ref} with respect to the reference time of the initiator. Using the approach from [21], we measure e_{ref} for our Glossy implementation and a network diameter of up to nine hops. Based on 340,000 data points, we find that e_{ref} ranges always between −7.1 µs and 8.6 µs. We thus

consider $\hat{e}_{ref} = 10$ µs a safe bound for the jitter on the reference time between CPs.

2) Independent clocks on CP and AP. Each AP schedules activities relative to SYNC line events. As AP and CP are sourced by independent clocks, it takes a variable amount of time until an AP detects that CP asserted the SYNC line. The resulting jitter is bounded by $\hat{e}_{SYNC} = 1/f_{AP}$, where $f_{AP} = 48$ MHz is the frequency of APs clock.

3) Different clock drift at CPs and APs. The real offsets and durations of activities on the CPs and APs depend on the frequency of their clocks. Various factors contribute to different frequency drifts ρ_{CP} and ρ_{AP}, including the manufacturing process, ambient temperature, and aging effects. State-of-the-art clocks, however, drift by at most $\hat{\rho}_{CP} = \hat{\rho}_{AP} = 50$ ppm [34].

4) Varying task execution times. The difference between the task's best- and worst-case execution time $\hat{e}_{task}$ adds to the jitter. For the jitter on the update interval T_U and the end-to-end delay T_D, only the execution time of the actuation task matters, which typically exhibits little variance as it is short and highly deterministic. For example, the actuation task in our experiments has a jitter of ±3.4 µs. To be safe, we consider $\hat{e}_{task} = 10$ µs for our analysis.

Using (1) and the above values, we can compute the worst-case jitter for a given interval $\widetilde{T}_{end}$. Fast feedback control as considered in this paper requires $\widetilde{T}_{end} = T_D = 2T_U \le 100$ ms, which gives a worst-case jitter of ±50 µs on T_U and T_D, as stated by property **P1**.

5 CONTROL DESIGN AND ANALYSIS

Building on the design of the wireless embedded system and its properties **P1**–**P4**, this section addresses the design of the control system to accomplish goals **G4**–**G6** from Section 3. Because the wireless system supports arbitrary traffic patterns (**P3**), various control tasks can be solved regardless of whether sensors, actuators, physical system(s), and controller(s) are co-located or spatially distributed. This includes typical single-loop tasks such as stabilization, disturbance rejection, or set-point tracking, as well as multi-agent scenarios such as synchronization, consensus, or formation control.

Here, we focus on remote stabilization and synchronization of multiple agents over wireless multi-hop networks as prototypical examples for both the single- and multi-agent case. For stabilization, modeling and control design are presented in Sections 5.1 and 5.2, thus achieving **G5**. The stability analysis is provided in Section 5.3, which fulfills **G4**. Synchronization is discussed in Section 5.4, highlighting support for straightforward distributed control **G6**.

5.1 Model of Wireless Control System

We address the remote stabilization task depicted in Figure 5 (left), where controller and physical system are associated with different nodes, which can communicate via the wireless network. Such a scenario is relevant for instance in process control, where the controller often resides at a remote location [42]. We consider stochastic LTI dynamics for the physical process

$$x(k+1) = Ax(k) + Bu(k) + v(k). \qquad (2a)$$

This model describes the evolution of the system state $x(k) \in \mathbb{R}^n$ with discrete time index $k \in \mathbb{N}$ in response to the control input

$u(k) \in \mathbb{R}^m$ and random process noise $v(k) \in \mathbb{R}^n$. As typical in the literature [6, 27], the process noise is modeled as an i.i.d. Gaussian random variable with zero mean and variance Σ_{proc}, $v(k) \sim \mathcal{N}(0, \Sigma_{\text{proc}})$, and captures, for example, uncertainty in the model.

We assume that the full system state $x(k)$ can be measured through appropriate sensors, that is,

$$y(k) = x(k) + w(k), \qquad (2b)$$

with sensor measurements $y(k) \in \mathbb{R}^n$ and sensor noise $w(k) \in \mathbb{R}^n$, $w(k) \sim \mathcal{N}(0, \Sigma_{\text{meas}})$. If the complete state vector cannot be measured directly, it can typically be reconstructed via state estimation techniques [6].

The process model in (2) is stated in discrete time. This representation is particularly suitable here as the wireless embedded system offers a constant update interval T_U with worst case jitter of $\pm 50\,\mu s$ (**P1**), which can be neglected from controls perspective [14, p. 48]. Thus, $u(k)$ and $y(k)$ in (2) represent sensing and actuation at periodic intervals T_U, as illustrated in Figure 4.

As shown in Figure 5, measurements $y(k)$ and control inputs $\hat{u}(k)$ are sent over the wireless network. According to **P1** and **P2**, both arrive at the controller, respectively system, with a delay of T_U and with a probability governed by two independent Bernoulli processes. We represent the Bernoulli processes by $\theta(k)$ and $\phi(k)$, which are i.i.d. binary variables, indicating lost ($\theta(k) = 0$, $\phi(k) = 0$) or successfully received ($\theta(k) = 1$, $\phi(k) = 1$) messages. To ease notation and since both variables are i.i.d., we can omit the time index in the following without any confusion. We denote the probability of successful message delivery by μ_θ (i.e., $\mathbb{P}[\theta = 1] = \mu_\theta$), respectively μ_ϕ. As both, measurements and control inputs, are delayed, it also follows that in case of no message losses, the applied control input $u(k)$ depends on the measurement two steps ago $y(k - 2)$. If a control input message is lost, the input stays constant since zero-order hold is used at the actuator, that is,

$$u(k) = \phi \hat{u}(k) + (1 - \phi)\, u(k - 1). \qquad (3)$$

The model proposed in this section thus captures the properties **P1**, **P2**, and **P4**. While **P1** and **P2** are incorporated in the presented dynamics and message loss models, **P4** means that there is no need to take duplicated or out-of-order sensor measurements and control inputs into account. Overall, these properties allow for accurately describing the wireless CPS by a fairly straightforward model, which greatly facilitates subsequent control design and analysis. Property **P3** is not considered here, where we deal with a single control loop, but will become essential in Section 5.4.

5.2 Controller Design

Designing a feedback controller for the system in (2), we proceed by first discussing state-feedback control for the nominal system (i.e., without delays, message loss, and noise), and then enhance the design to cope with the network and sensing imperfections.

Nominal design. Assuming ideal measurements, $y(k) = x(k)$ holds. A common strategy in this setting is static state-feedback control, $u(k) = Fx(k)$, where F is a constant feedback matrix, which can be designed, for instance, via *pole placement* or methods from optimal control, such as the *linear quadratic regulator (LQR)* [4, 6]. Under the assumption of controllability [6], desired (in particular, stable) dynamics can be obtained for the state in (2a) in this way.
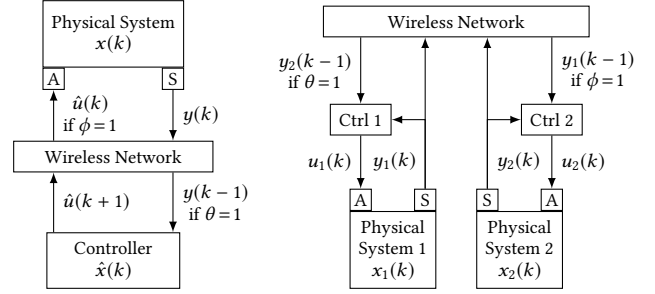

Figure 5: Wireless control tasks: stabilization (left) and synchronization (right). *The feedback loop for stabilizing the physical system (left) is closed over the multi-hop low-power wireless network. This induces delays and message loss, which is captured by i.i.d. Bernoulli variables θ and ϕ. Two physical systems, each with a local controller (Ctrl), are synchronized over the wireless network (right).*

Actual design. We augment the nominal state-feedback design to cope with non-idealities, in particular, delayed measurements and message loss, as shown in Figure 5 (left).

Because the measurement arriving at the controller $y(k - 1)$ represents information that is one time step in the past, the controller propagates the system for one step as follows:

$$\hat{x}(k) = \theta Ay(k-1) + (1-\theta)(A\hat{x}(k-1)) + B\hat{u}(k-1) \qquad (4)$$
$$= \theta Ax(k-1) + (1-\theta)A\hat{x}(k-1) + B\hat{u}(k-1) + \theta Aw(k-1),$$

where $\hat{x}(k)$ is the predicted state, and $\hat{u}(k)$ is the control input computed by the controller (to be made precise below). Both variables are computed by the controller and represent its internal states. The rationale of (4) is as follows: If the measurement message is delivered (the controller has information about θ because it knows when to expect a message), we compute the state prediction based on this measurement $y(k-1) = x(k-1) + w(k-1)$; if the message is lost, we propagate the previous prediction $\hat{x}(k-1)$. As there is no feedback on lost control messages (i.e., about ϕ) and thus a potential mismatch between the computed input $\hat{u}(k-1)$ and the actual $u(k-1)$, the controller can only use $\hat{u}(k-1)$ in the prediction.

Using $\hat{x}(k)$, the controller has an estimate of the current state of the system. However, it will take another time step for the currently computed control input to arrive at the physical system. For computing the next control input, we thus propagate the system another step,

$$\hat{u}(k + 1) = F\left(A\hat{x}(k) + B\hat{u}(k)\right), \qquad (5)$$

where F is as in the nominal design. The input $\hat{u}(k + 1)$ is then transmitted over the wireless network (see Figure 5, left).

The overall controller design requires only a few matrix multiplications per execution. This can be efficiently implemented on embedded devices, thus satisfying goal **G5**.

5.3 Stability Analysis

We now present a stability proof for the closed-loop system given by the dynamic system described in Section 5.1 and the proposed

controller from Section 5.2. Because the model in Section 5.1 incorporates the physical process and the essential properties of the wireless embedded system, we achieve goal **G4**.

While the process dynamics in (2) are time invariant, message loss introduces time variation and randomness into the system dynamics. Therefore, we leverage stability results for linear, stochastic, time-varying systems [12]. For ease of exposition, we consider (2) without process and measurement noise (*i.e.*, $v(k) = 0$ and $w(k) = 0$), and comment later on extensions. We first introduce required definitions and preliminary results, and then apply those results to our problem.

Consider the system

$$z(k + 1) = \tilde{A}(k)z(k) \qquad (6)$$

with state $z(k) \in \mathbb{R}^n$ and $\tilde{A}(k) = \tilde{A}_0 + \sum_{i=1}^{L} \tilde{A}_i p_i(k)$; the $p_i(k)$ are i.i.d. random variables with mean $\mathbb{E}[p_i(k)] = 0$, variance $\mathrm{Var}[p_i(k)] = \sigma_{p_i}^2$, and $\mathbb{E}[p_i(k)p_j(k)] = 0 \ \forall i, j$.

A common notion of stability for stochastic systems like the one in (6) is mean-square stability:

DEFINITION 1 ([12, p. 131]). *Let $Z(k) := \mathbb{E}[z(k)z^{\mathrm{T}}(k)]$ denote the state correlation matrix. The system in (6) is mean-square stable (MSS) if $\lim_{k \to \infty} Z(k) = 0$ for any initial $z(0)$.*

That is, a system is called MSS if the state correlation vanishes asymptotically for any intial state. MSS implies, for example, that $z(k) \to 0$ almost surely as $k \to \infty$ [12, p. 131].

In control theory, linear matrix inequalities (LMIs) are often used as computational tools to check for system properties such as stability (see [12] for an introduction and details). For MSS, we employ the following LMI stability result:

LEMMA 1 ([12, p. 131]). *System (6) is MSS if, and only if, there exists a positive definite matrix $P > 0$ such that*

$$\tilde{A}_0^{\mathrm{T}} P \tilde{A}_0 - P + \sum_{i=1}^{N} \sigma_{p_i}^2 \tilde{A}_i^{\mathrm{T}} P \tilde{A}_i < 0. \qquad (7)$$

We now apply this result to the system and controller from Sections 5.1 and 5.2. The closed-loop dynamics are given by (2)–(5), which we rewrite as an augmented system

$$\underbrace{\begin{pmatrix} x(k+1) \\ \hat{x}(k+1) \\ u(k+1) \\ \hat{u}(k+1) \end{pmatrix}}_{z(k+1)} = \underbrace{\begin{pmatrix} A & 0 & B & 0 \\ \theta A & (1-\theta)A & 0 & B \\ 0 & \phi FA & (1-\phi)I & \phi FB \\ 0 & FA & 0 & FB \end{pmatrix}}_{\tilde{A}(k)} \underbrace{\begin{pmatrix} x(k) \\ \hat{x}(k) \\ u(k) \\ \hat{u}(k) \end{pmatrix}}_{z(k)}. \qquad (8)$$

The system has the form of (6); the transition matrix depends on θ and ϕ, and thus on time (omitted for simplicity). We can thus apply Lemma 1 to obtain our main stability result, whose proof is given in Appendix A.1.

THEOREM 1. *The system (8) is MSS if, and only if, there exists a $P > 0$ such that (7) holds with*

$$\tilde{A}_0 = \begin{pmatrix} A & 0 & B & 0 \\ \mu_\theta A & (1-\mu_\theta)A & 0 & B \\ 0 & \mu_\phi FA & (1-\mu_\phi)I & \mu_\phi FB \\ 0 & FA & 0 & FB \end{pmatrix}, \quad \tilde{A}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\mu_\theta A & \mu_\theta A & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\tilde{A}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\mu_\phi FA & \mu_\phi I & -\mu_\phi FB \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \sigma_{p_1}^2 = 1/\mu_\theta - 1, \quad \sigma_{p_2}^2 = 1/\mu_\phi - 1.$$

Using Theorem 1, we can analyze stability for any concrete physical system (2), a state-feedback controller F, and probabilities μ_θ and μ_ϕ. Searching for a $P > 0$ that satisfies the LMI in (7) can be done using efficient numerical tools based on convex optimization (*e.g.*, [33]). If such a P is found, we have the stability guarantee (**G4**).

The stability analysis can be extended to account for process and measurement noise so that MSS then implies bounded $Z(k)$ (see [12, p. 138]). Moreover, other combinations of end-to-end delay T_D and update interval T_U are possible, including $T_D = nT_U$ ($n \in \mathbb{N}$). Also the sensor-to-controller and controller-to-actuator delays may be different.

5.4 Multi-agent Synchronization

In distributed or decentralized control architectures, different controllers have access to different measurements and inputs, and thus, in general, different information. This is the core reason for why such architectures are more challenging than centralized ones [24, 38]. Which information a controller has access to depends on the traffic pattern and topology of the network. For instance, an agent may only be able to communicate with its nearest neighbor via point-to-point communication, or with other agents in a certain range. Property **P3** of the wireless embedded system in Section 4 offers a key advantage compared to these structures because every agent in the network has access to all information (except for rare message losses). We can thus carry out a centralized design, but implement the resulting controllers in a distributed fashion (cf. Figure 5, right). Such schemes have been used before for wired-bus networks (*e.g.*, in [55]).

Here, we present synchronization of multiple physical systems as an *example* of how distributed control tasks can easily be achieved with the proposed wireless control system (**G6**). We assume multiple physical processes as in (2), but with possibly different dynamics parameters (A_i, B_i, etc.). We understand synchronization in this setting as the goal of having the system state of different agents evolve together as close as possible. That is, we want to keep the error $x_i(k) - x_j(k)$ between the states of systems i and j small. Instead of synchronizing the whole state vector, also a subset of all states can be considered. Synchronization of multi-agent systems is a common problem and also known as consensus or coordination [39].

We demonstrate feasibility of synchronization with multiple systems in Section 6.3. The synchronizing controller is based on an LQR [4]; details of the concrete design are given in Appendix A.3.

6 EXPERIMENTAL EVALUATION

This section uses measurements from a cyber-physical testbed (see Figure 6) consisting of 20 wireless embedded devices (forming a three-hop network) and several cart-pole systems to evaluate the performance of the proposed wireless CPS design. Our experiments reveal the following key findings:

- We can safely stabilize two inverted pendulums via two remote controllers across the three-hop low-power wireless network.
- Using the same CPS design with a different control logic, we can reliably synchronize the movement of five cart-poles thanks to the support for arbitrary traffic patterns.
- Our system can stabilize an inverted pendulum at update intervals of 20–50 ms. Increasing the update interval decreases the
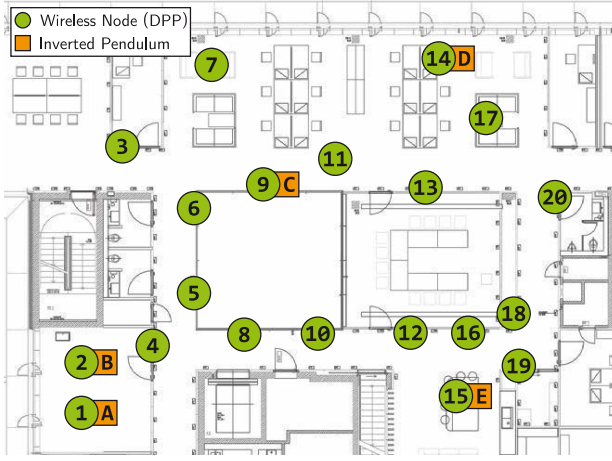
Figure 6: Layout of cyber-physical testbed consisting of 20 DPP nodes that form a three-hop low-power wireless network and five cart-pole systems (two real ones attached to nodes 1 and 2, three simulated ones at nodes 9, 14, and 15).

Figure 7: Schematic of a cart-pole system used in our testbed as physical systems. *By controlling the force applied to the cart, the pole can be stabilized in the upright position around $\theta = 0°$.*

control performance, but leads to significant energy savings on the wireless communication side.

- Our system is highly robust to message loss. Specifically, it can stabilize an inverted pendulum at an update interval of 20 ms despite 75% i.i.d. Bernoulli losses and in situations with bursts of 40 consecutively lost messages.
- The measured jitter on the update interval and the end-to-end delay is less than ±25 µs, which validates our analysis of the theoretical worst-case jitter of ±50 µs from Section 4.4.

6.1 Cyber-physical Testbed

Realistic cyber-physical testbeds are essential for the validation and evaluation of CPS solutions [9, 36]. We developed the wireless cyber-physical testbed depicted in Figure 6. It consists of 20 DPP nodes, two real physical systems (A and B), and three simulated physical systems (C, D, and E). The testbed is deployed in an office building and extends across an area of 15 m by 20 m. All nodes transmit at 10 dBm, which results in a network diameter of three hops. The wireless signals need to penetrate various types of walls, from glass to reinforced concrete, and are exposed to different sources of interference from other electronics and human activity.

We use *cart-pole systems* as physical systems. As shown in Figure 7, a cart-pole system consists of a cart that can move horizontally on a track and a pole attached to it via a revolute joint. The cart is equipped with a DC motor that can be controlled by applying a voltage to influence the speed and the direction of the cart. Moving the cart exerts a force on the pole and thus influences the pole angle θ. This way, the pole can be stabilized in an upright position around $\theta = 0°$, which represents an unstable equilibrium and is called the *inverted pendulum*. The inverted pendulum has fast dynamics, which are typical of real-world mechanical systems [11], and requires feedback with update intervals of tens of milliseconds.

For small deviations from the equilibrium (*i.e.*, $\sin(\theta) \approx \theta$), the inverted pendulum can be well approximated by an LTI system.
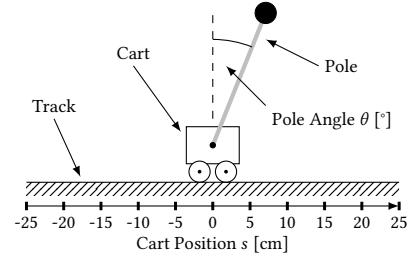
The state $x(k)$ of the system consists of four variables. Two of them, the pole angle $\theta(k)$ and the cart position $s(k)$, are measured by angle sensors. Their derivatives, the angular velocity $\dot{\theta}(k)$ and the cart velocity $\dot{s}(k)$, are estimated using finite differences and low-pass filtering. The voltage applied to the motor is the control input $u(k)$. In this way, the APs of nodes 1 and 2 interact with the two real pendulums A and B, while the APs of nodes 9, 14, and 15 run simulation models of the inverted pendulum.

The cart-pole system has a few constraints. Control inputs are capped at ±10 V. The track has a usable length of ±25 cm from the center (see Figure 7). Surpassing the track limits immediately ends an experiment. At the beginning of an experiment, we move the carts to the center and the poles in the upright position; then the controller takes over. Appendices A.2 and A.3 detail the implementation of the controllers for multi-hop stabilization and multi-hop synchronization, following the design outlined in Sections 5.2 and 5.4.

Using this cyber-physical testbed, we measure the control performance in terms of pole angle, cart position, and control input. In addition, we measure the radio duty cycle at each node in software and record messages that are lost over the wireless network.

6.2 Multi-hop Stabilization

In our first experiment, we want to answer the main question of this work and investigate the feasibility of fast feedback control over multi-hop low-power wireless networks.

Setup. We use two controllers running on nodes 14 and 15 to stabilize the two real pendulums A and B at $\theta = 0°$ and $s = 0$ cm. Hence, there are two independent control loops sharing the same wireless network, and it takes in total six hops to close each loop. We configure the wireless embedded system and the controllers for an update interval of $T_U = 45$ ms. As per property **P2** and confirmed by our measurements discussed below, we expect a message delivery rate of at least 99.9 %. With this we have $\mu_\theta = \mu_\phi = 0.999$, and we can prove stability of the overall system using Theorem 1.

Results. The experimental results confirm the theoretical analysis: We are able to safely stabilize both pendulums over the three-hop wireless network, even while carrying around the controller because our design is independent of the network topology [2] (see Section 4.1). Figure 8 shows a characteristic 30 s trace of one of the pendulums. Cart position, pole angle, and control input oscillate,

---

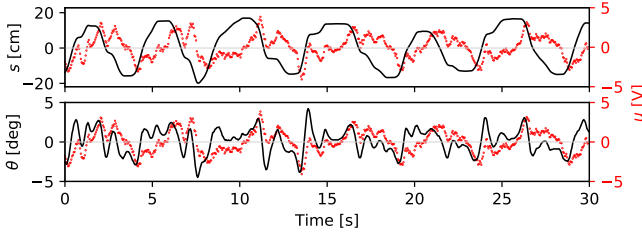[2]A video of this experiment can be found at https://youtu.be/19xPHjnobkY.

Figure 8: Cart position s, pole angle θ, and control input u of one cart-pole system when concurrently stabilizing two cart-pole systems over a multi-hop network at an update interval of 45 ms. *The cart position and the pole angle always stay within safe regimes, and less than half of the possible control input is needed.*
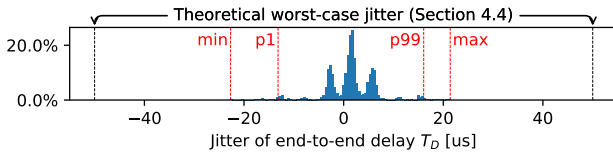

Figure 9: Distribution of the jitter on the end-to-end delay T_D; results for the update interval T_U are similar. *The measurements are within the worst-case bounds determined in Section 4.4.*

but always stay within safe regimes. For example, the cart never comes close to either end of the track and less than half of the possible control input is needed to stabilize the pendulum. Not a single message was lost in this experiment, which demonstrates the reliability of our wireless embedded system design.

During the same experiment, we also use a logic analyzer to continuously measure the update interval T_U and the end-to-end delay T_D (see Figure 4). Figure 9 shows the measured jitter on T_D; the results for T_U look very similar. We see that the empirical results are well within the theoretical worst-case bounds, which validates our analysis in Section 4.4 and assumptions in Section 5.

6.3 Multi-hop Synchronization

We now apply the same wireless CPS design to a distributed control task to demonstrate its flexibility and versatility.

Setup. We use the two real pendulums A and B and the three simulated pendulums C, D, and E. The goal is to synchronize the cart positions of the five pendulums over the wireless multi-hop network, while each pendulum is stabilized by a local control loop. This scenario is similar to drone swarm coordination, where each drone stabilizes its flight locally, but exchanges its position with all other drones to keep a desired swarm formation [45]. In our experiment, stabilization runs with $T_U = 10$ ms, and nodes 1, 2, 9, 14, and 15 exchange their current cart positions every 50 ms.

Results. The left plot in Figure 10 shows the cart positions over time without synchronization. We see that the carts of the real pendulums move with different amplitude, phase, and frequency due to slight differences in their physics and imperfect measurements. The simulated pendulums, instead, are perfectly balanced and behave deterministically as they all start in the same initial state.

In the middle plot of Figure 10, we can observe the behavior of the pendulums when they synchronize their cart positions over

the wireless multi-hop network. Now, all five carts move in concert. The movements are not perfectly aligned because, besides the synchronization, each cart also needs to locally stabilize its pole at $\theta = 0°$ and $s = 0$ cm. Since no message is lost during the experiment, the simulated pendulums all receive the same state information and, therefore, show identical behavior.

This effect can also be seen in our third experiment, shown in the right plot of Figure 10, where we hold pendulum A for some time at $s = -20$ cm. The other pendulums now have two conflicting control goals: stabilization at $s = 0$ cm and $\theta = 0°$, as well as synchronization while one pendulum is fixed at about $s = -20$ cm. As a result, they all move towards this position and oscillate between $s = 0$ and $s = -20$ cm. Clearly, this experiment demonstrates that the cart-pole systems influence each other, which is enabled by the many-to-all communication over the wireless multi-hop network.

6.4 Impact of Update Interval

The next experiment takes a look at the impact of different update intervals (and hence end-to-end delays) on control performance.

Setup. To minimize effects that we cannot control, such as external interference, we use two nodes close to each other: pendulum A (node 1) is stabilized via a remote controller running on node 2. We test different update intervals in consecutive runs. Starting with the smallest update interval of 20 ms that the wireless embedded system can support in this scenario, we increase the update interval in steps of 10 ms until stabilization is no longer possible.

Results. Figure 11 shows control performance and radio duty cycle for different update intervals based on more than 12,500 data points. We see that a longer update interval causes larger pole angles and more movement of the cart. Indeed, the total distance the cart moves during an experiment increases from 3.40 m for 20 ms to 9.78 m for 50 ms. This is consistent with the wider distribution of the control input for longer update intervals. At the same time, the radio duty cycle decreases from 40 % for 20 ms to 15 % for 50 ms. Hence, there is a trade-off between control performance and energy efficiency, which may be exploited based on the application requirements.

6.5 Resilience to Message Loss

Finally, we evaluate how control performance is affected by message loss, which is a well-known phenomenon in wireless networks [52].

Setup. We use again the two-node setup from before, but now we fix the update interval at 20 ms. We let both nodes intentionally drop messages in two different ways. In a first experiment, the two nodes independently drop a received message according to a Bernoulli process with given failure probability. Specifically, we test three different failure probabilities in different runs: 15 %, 45 %, and 75 %. In a second experiment, the two nodes drop a certain number of consecutive messages every 10 s, namely between 10 and 40 messages in different runs. This artificially violates property **P2** of the wireless embedded system, yet allows us to evaluate the robustness of our control design to unexpected conditions.

Results. Figures 12a and 12b show the distributions of the pole angle and the control input for varying i.i.d. Bernoulli message loss rates. We see that the control performance decreases for higher loss rates, but the pendulum can be stabilized even at a loss rate of 75%. One reason for this is the short update interval. For example, losing
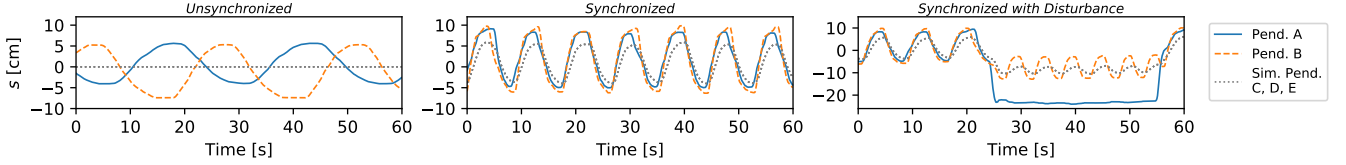
Figure 10: Cart positions of five cart-pole systems stabilized locally at an update interval of 10 ms and synchronizing their cart positions (middle and right plot) over the multi-hop low-power wireless network at an update interval of 50 ms. *With synchronization enabled, all five carts move in concert and even try to mimic the temporary disturbance of pendulum A (right plot).*
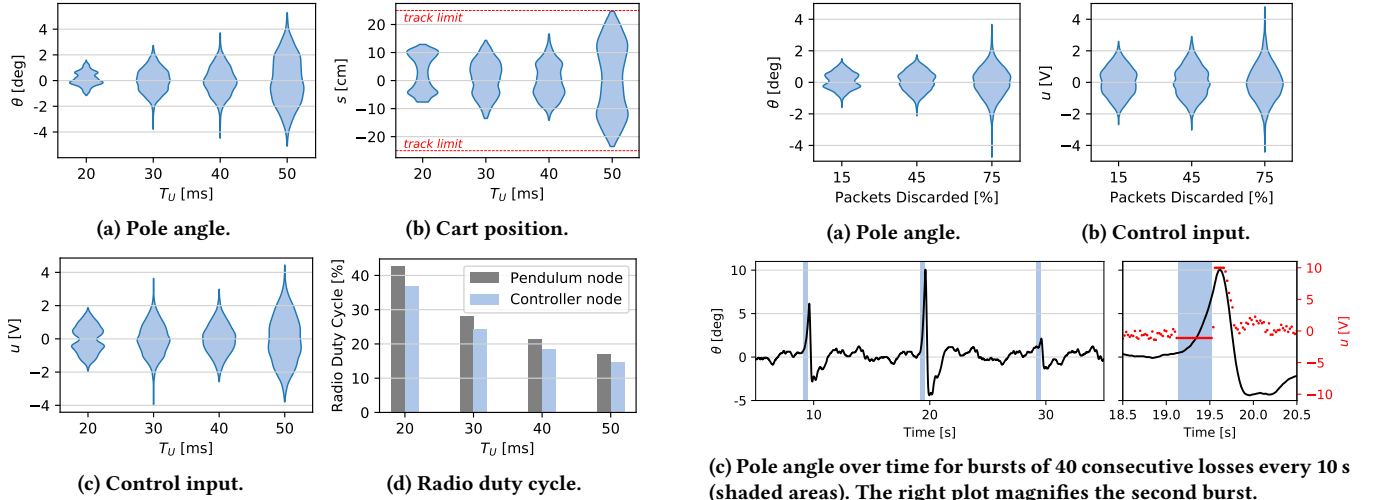

Figure 11: Distribution of control performance metrics and average radio duty cycle when stabilizing an inverted pendulum over low-power wireless at different update intervals. *A larger update interval leads to larger pole angles and more movement of the cart, but also reduces the average radio duty cycle.*

(c) Pole angle over time for bursts of 40 consecutive losses every 10 s (shaded areas). The right plot magnifies the second burst.
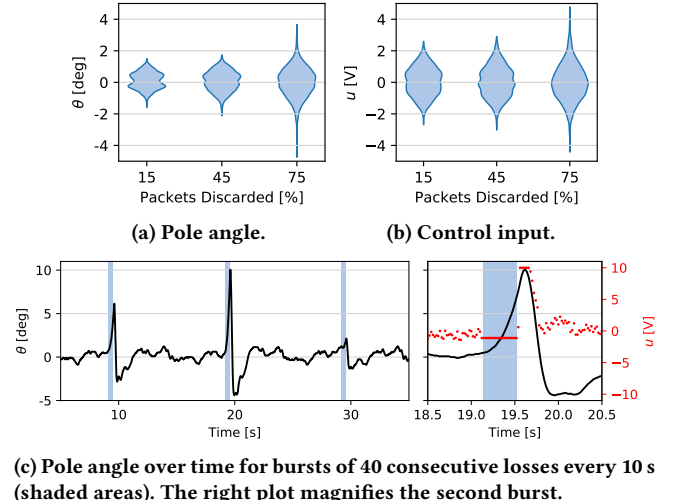
Figure 12: Control performance and input when stabilizing one pendulum over low-power wireless under artificially injected message loss, for i.i.d. Bernoulli losses (in a and b) and for longer bursts of multiple consecutive losses (in c). *Depending on the update interval, the pendulum can be stabilized despite significant and bursty message loss, albeit with reduced performance.*

50 % of the messages at an update interval of 20 ms is comparable to an update interval of 40 ms without any losses, which is enough to stabilize the pendulums as we know from the previous experiment.

Figure 12c plots the pole angle as a function of time for a burst length of 40 consecutively lost messages, with the right plot zooming into the time around the second burst phase. No control inputs are received during a burst, and depending on the state of the pendulum and the control input right before a burst, the impact of a burst may be very different as visible in Figure 12c. The magnified plot shows that the pole angle diverges from around 0° with increasing speed. As soon as the burst ends, the control input rises to its maximum value of 10 V in order to bring the pendulum back to a non-critical state, which usually takes 1-2 s. These results show that while property **P2** of our wireless embedded system design significantly simplifies control design and analysis, the overall system remains stable even if this property is dramatically violated, which is nevertheless very unlikely as demonstrated in prior work [32, 64].

7 CONCLUSIONS

We have presented a CPS design that enables, for the first time, fast closed-loop control over multi-hop low-power wireless networks with update intervals of 20-50 ms. Other existing solutions for feedback control over wireless are either limited to the single-hop case

or to systems with slow dynamics, where update intervals of several seconds are sufficient. Through a tight co-design approach, we tame network imperfections and take the resulting properties of the communication network into account in the control design. This enables to formally prove closed-loop stability of the entire CPS. Experiments on a novel cyber-physical testbed with multiple physical systems further demonstrate the applicability, versatility, and robustness of our design. By demonstrating how to close feedback loops quickly and reliably over multiple wireless hops, this paper is an important stepping stone toward realizing the CPS vision.

REFERENCES

[1] Jannik Abbenseth, Felipe Garcia Lopez, Christian Henkel, and Stefan Dörr. 2017. Cloud-Based Cooperative Navigation for Mobile Service Robots in Dynamic Industrial Environments. In *Proceedings of the 32rd ACM Symposium on Applied Computing (SAC)*.

[2] Johan Åkerberg, Mikael Gidlund, and Mats Björkman. 2011. Future Research Challenges in Wireless Sensor and Actuator Networks Targeting Industrial Automation. In *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN)*.

[3] Rajeev Alur, Alessandro D'Innocenzo, Karl H. Johansson, George J. Pappas, and Gera Weiss. 2011. Compositional Modeling and Analysis of Multi-Hop Control Networks. *IEEE Trans. Automat. Control* 56, 10 (2011).

[4] Brian D. O. Anderson and John B. Moore. 2007. *Optimal Control: Linear Quadratic Methods*. Prentice Hall.

[5] José Araújo, Manuel Mazo, Adolfo Anta, Paulo Tabuada, and Karl H. Johansson. 2014. System Architectures, Protocols and Algorithms for Aperiodic Wireless Control Systems. *IEEE Transactions on Industrial Informatics* 10, 1 (2014).

[6] Karl Johan Åström and Richard M. Murray. 2008. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

[7] Karl Johan Åström and Björn Wittenmark. 1996. *Computer-Controlled Systems: Theory and Design* (3rd ed.). Prentice Hall.

[8] Nicolas W. Bauer, S.J.L.M. Bas Van Loon, Nathan Van De Wouw, and W.P.M.H. Maurice Heemels. 2014. Exploring the Boundaries of Robust Stability Under Uncertain Communication: An NCS Toolbox Applied to a Wireless Control Setup. *IEEE Control Systems Magazine* 34, 4 (2014).

[9] Dominik Baumann, Fabian Mager, Harsoveet Singh, Marco Zimmerling, and Sebastian Trimpe. 2018. Evaluating Low-Power Wireless Cyber-Physical Systems. In *Proceedings of the 1st IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*.

[10] Bart Besselink, Valerio Turri, Sebastian H. Van De Hoef, Kuo-Yun Liang, Assad Alam, Jonas Mårtensson, and Karl H. Johansson. 2016. Cyber-Physical Control of Road Freight Transport. *Proc. IEEE* 104, 5 (2016).

[11] Olfa Boubaker. 2012. The Inverted Pendulum: A Fundamental Benchmark in Control Theory and Robotics. In *Proceedings of the International Conference on Education and e-Learning Innovations (ICEELI)*.

[12] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. 1994. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics (SIAM).

[13] Matteo Ceriotti, Michele Corrà, Leandro D'Orazio, Roberto Doriguzzi, Daniele Facchin, Stefan Guna, Gian Paolo Jesi, Renato Lo Cigno, Luca Mottola, Amy L. Murphy, Massimo Pescalli, Gian Pietro Picco, Denis Pregnolato, and Carloalberto Torghele. 2011. Is There Light at the Ends of the Tunnel? Wireless Sensor Networks for Adaptive Lighting in Road Tunnels. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

[14] Anton Cervin. 2003. *Integrated Control and Real-Time Scheduling*. Ph.D. Dissertation. Lund University.

[15] Peter Corke, Tim Wark, Raja Jurdak, Wen Hu, Philip Valencia, and Darren Moore. 2010. Environmental Wireless Sensor Networks. *Proc. IEEE* 98, 11 (2010).

[16] Nikolaus Correll, Prabal Dutta, Richard Han, and Kristofer Pister. 2017. New Directions: Wireless Robotic Materials. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys)*.

[17] Patricia Derler, Edward A. Lee, and Alberto Sangiovanni-Vincentelli. 2012. Modeling Cyber-Physical Systems. *Proc. IEEE* 100, 1 (2012).

[18] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. 2015. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[19] Johan Eker, Anton Cervin, and Andreas Hörjel. 2001. Distributed Wireless Control Using Bluetooth. In *Proceedings of the IFAC Conference on New Technologies for Computer Control (NTCC)*.

[20] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. 2012. Low-Power Wireless Bus. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SenSys)*.

[21] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. 2011. Efficient Network Flooding and Time Synchronization with Glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

[22] Konstantinos Gatsis, Alejandro Ribeiro, and George J. Pappas. 2014. Optimal Power Management in Wireless Control Systems. *IEEE Trans. Automat. Control* 59, 6 (2014).

[23] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. 2009. Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[24] Pulkit Grover. 2014. *Information Structures, the Witsenhausen Counterexample, and Communicating Using Actions*. Springer.

[25] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. 2016. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Communications Surveys and Tutorials* 18, 4 (2016).

[26] Aitor Hernandez, Joao Faria, José Araújo, Pangun Park, Henrik Sandberg, and Karl H. Johansson. 2011. Inverted Pendulum Control over an IEEE 802.15.4 Wireless Sensor and Actuator Network. In *Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN)*.

[27] Joao P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. 2007. A Survey of Recent Results in Networked Control Systems. *Proc. IEEE* 95, 1 (2007).

[28] Quanser Inc. 2012. IP02 - Self-Erecting Single Inverted Pendulum - Linear Experiment #6: PV and LQR Control - Instructor Manual.

[29] Timofei Istomin, Amy L. Murphy, Gian Pietro Picco, and Usman Raza. 2016. Data Prediction + Synchronous Transmissions = Ultra-low Power Wireless Sensor Networks. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems (SenSys)*.

[30] Romain Jacob, Licong Zhang, Marco Zimmerling, Jan Beutel, Samarjit Chakraborty, and Lothar Thiele. 2018. TTW: A Time-Triggered-Wireless Design for CPS. In *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

[31] Romain Jacob, Marco Zimmerling, Pengcheng Huang, Jan Beutel, and Lothar Thiele. 2016. End-to-End Real-Time Guarantees in Wireless Cyber-Physical Systems. In *Proceedings of the 36th IEEE Real-Time Systems Symposium (RTSS)*.

[32] Jens Karschau, Marco Zimmerling, and Benjamin M. Friedrich. 2018. Renormalization Group Theory for Percolation in Time-varying Networks. *Scientific Reports* 8 (2018).

[33] Yann Labit, Dimitri Peaucelle, and Didier Henrion. 2002. SEDUMI INTERFACE 1.02: A tool for solving LMI problems with SEDUMI. In *Proceedings of the 12 IEEE International Symposium on Computer Aided Control System Design (CACSD)*.

[34] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. 2015. PulseSync: An Efficient and Scalable Clock Synchronization Protocol. *IEEE/ACM Transactions on Networking* 23, 3 (2015).

[35] Bo Li, Yehan Ma, Tyler Westenbroek, Chengjie Wu, Humberto Gonzalez, and Chenyang Lu. 2016. Wireless Routing and Control: A Cyber-Physical Case Study. In *Proceedings of the 7th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*.

[36] Chenyang Lu, Abusayeed Saifullah, Bo Li, Mo Sha, Humberto Gonzalez, Dolvara Gunatilaka, Chengjie Wu, Lanshun Nie, and Yixin Chen. 2016. Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems. *Proc. IEEE* 104, 5 (2016).

[37] Rogelio Luck and Asok Ray. 1990. An Observer-based Compensator for Distributed Delays. *Automatica* 26, 5 (1990).

[38] Jan Lunze. 1992. *Feedback Control of Large-Scale Systems*. Prentice Hall.

[39] Jan Lunze. 2012. Synchronization of Heterogeneous Agents. *IEEE Trans. Automat. Control* 57, 11 (2012).

[40] Michele Luvisotto, Zhibo Pang, and Dacfey Dzung. 2017. Ultra High Performance Wireless Control for Critical Applications: Challenges and Directions. *IEEE Transactions on Industrial Informatics* 13, 3 (2017).

[41] J. P. Lynch, Y. Wang, R. A. Swartz, K. C. Lu, and C. H. Loh. 2007. Implementation of a Closed-loop Structural Control System Using Wireless Sensor Networks. *Structural Control and Health Monitoring* 15, 4 (2007).

[42] Yehan Ma, Dolvara Gunatilaka, Bo Li, Humberto Gonzalez, and Chenyang Lu. 2018. Holistic Cyber-Physical Management for Dependable Wireless Control Systems. *ACM Transactions on Cyber-Physical Systems* 3, 1 (2018).

[43] Fredrik Österlind and Adam Dunkels. 2008. Approaching the Maximum 802.15.4 Multi-hop Throughput. In *Proceedings of the 5th ACM Workshop on Embedded Networked Sensors (HotEmNets)*.

[44] Nicholas J. Ploplys, Paul A. Kawka, and Andrew G. Alleyne. 2004. Closed-Loop Control over Wireless Networks. *IEEE Control Systems Magazine* 24, 3 (2004).

[45] James A. Preiss, Wolfgang Hönig, Gaurav S. Sukhatme, and Nora Ayanian. 2017. Crazyswarm: A Large Nano-Quadcopter Swarm. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

[46] Daniel E. Quevedo, Anders Ahlén, Alex S. Leong, and Subhrakanti Dey. 2012. On Kalman Filtering over Fading Wireless Channels with Controlled Transmission Powers. *Automatica* 48, 7 (2012).

[47] Matt Rich and Nicola Elia. 2015. Optimal Mean-square Performance for MIMO Networked Systems. In *Proceedings of the American Control Conference (ACC)*.

[48] Abusayeed Saifullah, Sriram Sankar, Jie Liu, Chenyang Lu, Ranveer Chandra, and Bodhi Priyantha. 2014. CapNet: A Real-Time Wireless Management Network for Data Center Power Capping. In *Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS)*.

[49] Craig B. Schindler, Thomas Watteyne, Xavier Vilajosana, and Kristofer S. J. Pister. 2017. Implementation and Characterization of a Multi-hop 6TiSCH Network for Experimental Feedback Control of an Inverted Pendulum. In *Proceedings of the 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*.

[50] Bruno Sinopoli, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, Michael I. Jordan, and Shankar S. Sastry. 2004. Kalman Filtering With Intermittent Observations. *IEEE Trans. Automat. Control* 49, 9 (2004).

[51] F. Smarra, A. D'Innocenzo, and M. D. Di Benedetto. 2012. Optimal Co-design of Control, Scheduling and Routing in Multi-hop Control Networks. In *Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*.

[52] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. 2010. An Empirical Study of Low-Power Wireless. *ACM Transactions on Sensor Networks* 6, 2 (2010).

[53] Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. 2008. The β-factor: Measuring Wireless Link Burstiness. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*.

[54] Felix Sutton, Marco Zimmerling, Reto Da Forno, Roman Lim, Tonio Gsell, Georgia Giannopoulou, Federico Ferrari, Jan Beutel, and Lothar Thiele. 2015. Bolt: A Stateful Processor Interconnect. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[55] Sebastian Trimpe and Raffaello D'Andrea. 2012. The Balancing Cube: A Dynamic Sculpture as Test Bed for Distributed Estimation and Control. *IEEE Control Systems Magazine* 32, 6 (2012).

[56] Gregory C. Walsh, Hong Ye, and Linda G. Bushnell. 2002. Stability Analysis of Networked Control Systems. *IEEE Transactions on Control Systems Technology* 10, 3 (2002).

[57] Thomas Watteyne, Vlado Handziski, Xavier Vilajosana, Simon Duquennoy, Oliver Hahm, Emmanuel Baccelli, and Adam Wolisz. 2016. Industrial Wireless IP-Based Cyber-Physical Systems. *Proc. IEEE* 104, 5 (2016).

[58] Björn Wittenmark, Johan Nilsson, and Martin Törngren. 1995. Timing Problems in Real-time Control Systems. In *Proceedings of the American Control Conference (ACC)*.

[59] Junlin Xiong and James Lam. 2007. Stabilization of Linear Systems over Networks with Bounded Packet Loss. *Automatica* 43, 1 (2007).

[60] T. C. Yang, H. Yu, M. R. Fei, and L. X. Li. 2005. Networked Control Systems: A Historical Review and Current Research Topics. *Measurement & Control* 38, 1 (2005).

[61] Hong Ye, Gregory C. Walsh, and Linda G. Bushnell. 2001. Real-Time Mixed-Traffic Wireless Networks. *IEEE Transactions on Industrial Electronics* 48, 5 (2001).

[62] Lixian Zhang, Huijun Gao, and Okyay Kaynak. 2013. Network-Induced Constraints in Networked Control Systems - A Survey. *IEEE Transactions on Industrial Informatics* 9, 1 (2013).

[63] Wei Zhang, Michael S. Branicky, and Stephen M. Phillips. 2001. Stability of Networked Control Systems. *IEEE Control Systems Magazine* 21, 1 (2001).

[64] Marco Zimmerling, Federico Ferrari, Luca Mottola, and Lothar Thiele. 2013. On Modeling Low-Power Wireless Protocols Based on Synchronous Packet Transmissions. In *Proceedings of the 21st IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*.

[65] Marco Zimmerling, Luca Mottola, Pratyush Kumar, Federico Ferrari, and Lothar Thiele. 2017. Adaptive Real-Time Communication for Wireless Cyber-Physical Systems. *ACM Transactions on Cyber-Physical Systems* 1, 2 (2017).

A CONTROL DETAILS

In this appendix, we provide further details of the control design and analysis. We present the proof of Theorem 1, implementation details of the controllers we use for the multi-hop stabilization experiments, and outline the approach to multi-agent synchronization.

A.1 Proof of Theorem 1

For clarity, we reintroduce time index k for θ and ϕ. Following a similar approach as in [47], we transform $\theta(k)$ as $\theta(k) = \mu_\theta (1 - \delta_\theta(k))$ with the new binary random variable $\delta_\theta(k) \in \{1, 1 - 1/\mu_\theta\}$ with $\mathbb{P}[\delta_\theta(k) = 1] = 1 - \mu_\theta$ and $\mathbb{P}[\delta_\theta(k) = 1 - 1/\mu_\theta] = \mu_\theta$; and analogously for $\phi(k)$ and $\delta_\phi(k)$. We thus have that $\delta_\theta(k)$ is i.i.d. because θ is i.i.d. with $\mathbb{E}[\delta_\theta(k)] = 0$ and $\text{Var}[\delta_\theta(k)] = \sigma_{p_1}^2$, and similarly for $\delta_\phi(k)$. Employing this transformation, $\tilde{A}(k)$ in (8) is rewritten as $\tilde{A}(k) = \tilde{A}_0 + \sum_{i=1}^2 \tilde{A}_i p_i(k)$ with $p_1(k) = \delta_\theta(k)$, $p_2(k) = \delta_\phi(k)$, and $\tilde{A}_i$ as stated in Theorem 1. Thus, all properties of (6) are satisfied, and Lemma 1 yields the result.

A.2 Stabilizing Controllers

For the stability experiments of Section 6.2, we employ the design outlined in Section 5.2. The system matrices A and B of the cart-pole

system that are used for predictions and nominal controller design are given by the manufacturer in [28]. The nominal controller is designed for an update interval $T_U = 40$ ms via pole placement, and we choose F such that we get closed-loop eigenvalues at 0.8, 0.85, and 0.9 (twice). In experiments with update intervals different from 40 ms, we adjust the controller to achieve similar closed-loop behavior.

To derive more accurate estimates of the velocities, filtering can be done at higher update intervals than communication occurs. For the experiments in Section 6, estimation and filtering occur at intervals between 10 ms and 20 ms, depending on the experiment.

A.3 Synchronization

For simplicity, we consider synchronization of two agents in the following, but the approach directly extends to more than two, as we show in the experiments in Section 6.3.

We consider the architecture in Figure 5, where each physical system is associated with a local controller that receives local observations directly, and observations from other agents over the network. We present an approach based on an optimal LQR [4] to design the synchronizing controllers. We choose the quadratic cost function

$$J = \lim_{K \to \infty} \frac{1}{K} \mathbb{E}\Big[\sum_{k=0}^{K-1} \sum_{i=1}^2 \Big(x_i^{\mathrm{T}}(k) Q_i x_i(k) + u_i^{\mathrm{T}}(k) R_i u_i(k) \Big)$$
$$+ (x_1(k) - x_2(k))^{\mathrm{T}} Q_{\mathrm{sync}} (x_1(k) - x_2(k)) \Big] \quad (9)$$

which expresses our objective of keeping $x_1(k) - x_2(k)$ small (through the weight $Q_{\mathrm{sync}} > 0$), next to usual penalties on states ($Q_i > 0$) and control inputs ($R_i > 0$). Using augmented state $\tilde{x}(k) = (x_1(k), x_2(k))^{\mathrm{T}}$ and input $\tilde{u}(k) = (u_1(k), u_2(k))^{\mathrm{T}}$, the term in the summation over k can be rewritten as

$$\tilde{x}^{\mathrm{T}}(k) \begin{pmatrix} Q_1 + Q_{\mathrm{sync}} & -Q_{\mathrm{sync}} \\ -Q_{\mathrm{sync}} & Q_2 + Q_{\mathrm{sync}} \end{pmatrix} \tilde{x}(k) + \tilde{u}^{\mathrm{T}}(k) \begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix} \tilde{u}(k).$$

Thus, the problem is in standard LQR form and can be solved with standard tools [4]. The optimal stabilizing controller that minimizes (9) has the structure $u_1(k) = F_{11} x_1(k) + F_{12} x_2(k)$ and $u_2(k) = F_{21} x_1(k) + F_{22} x_2(k)$; that is, agent 1 ($u_1(k)$) requires state information from agent 2 ($x_2(k)$), and vice versa. Because of many-to-all communication, the wireless embedded system directly supports this (as well as any other possible) controller structure (**P3**).

As the controller now runs on the node that is co-located with the physical process, local measurements and inputs are not sent over the wireless network and the local sampling time can be shorter than the update interval of the network over which the states of other agents are received. While the analysis in Section 5.3 can be generalized to the synchronization setting, a formal stability proof is beyond the scope of this paper. In general, stability is less critical here because of shorter update intervals in the local feedback loop.

For the synchronization experiments in Section 6.3, we choose Q_i in (9) for all pendulums as suggested by the manufacturer in [28] and set $R_i = 0.1$. As we here care to synchronize the cart positions, we set the first diagonal entry of Q_{sync} to 5 and all others to 0.