KTH Electrical Engineering

Fast and Resource-Efficient Control of Wireless Cyber-Physical Systems

DOMINIK BAUMANN

Licentiate Thesis
Stockholm, Sweden, 2019

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
Automatic Control Lab
SE-100 44 Stockholm
SWEDEN

TRITA-EECS-AVL-2019:7
ISBN: 978-91-7873-067-4

Akademisk avhandling som med tillstånd av Kungliga Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie licenciatexamen i reglerteknik fredagen den 15 februari 2019 klockan 14.00 i sal Q2 Kungliga Tekniska högskolan, Malvinas väg 10, KTH, Stockholm.

© Dominik Baumann, February 2019. All rights reserved.

Tryck: Universitetsservice US AB

Abstract

Cyber-physical systems (CPSs) tightly integrate physical processes with computing and communication to autonomously interact with the surrounding environment. This enables emerging applications such as autonomous driving, coordinated flight of swarms of drones, or smart factories. However, current technology does not provide the reliability and flexibility to realize those applications. Challenges arise from wireless communication between the agents and from the complexity of the system dynamics. In this thesis, we take on these challenges and present three main contributions.

We first consider imperfections inherent in wireless networks, such as communication delays and message losses, through a tight co-design. We tame the imperfections to the extent possible and address the remaining uncertainties with a suitable control design. That way, we can guarantee stability of the overall system and demonstrate feedback control over a wireless multi-hop network at update rates of 20-50 ms.

If multiple agents use the same wireless network in a wireless CPS, limited bandwidth is a particular challenge. In our second contribution, we present a framework that allows agents to predict their future communication needs. This allows the network to schedule resources to agents that are in need of communication. In this way, the limited resource communication can be used in an efficient manner.

As a third contribution, to increase the flexibility of designs, we introduce machine learning techniques. We present two different approaches. In the first approach, we enable systems to automatically learn their system dynamics in case the true dynamics diverge from the available model. Thus, we get rid of the assumption of having an accurate system model available for all agents. In the second approach, we propose a framework to directly learn actuation strategies that respect bandwidth constraints. Such approaches are completely independent of a system model and straightforwardly extend to nonlinear settings. Therefore, they are also suitable for applications with complex system dynamics.

Sammanfattning

Cyber-physical systems (CPSs) integrerar fysiska processer med beräkningar och kommunikation för att autonomt interagera med omgivningen. Detta möjliggör nya applikationer som autonom körning, koordinerat flyg av dronsvärmar eller smarta fabriker. Den nuvarande tekniken ger dock inte tillräcklig tillförlitlighet och flexibilitet för att förverkliga dessa applikationer. Utmaningar uppkommer från den trådlösa kommunikationen mellan agenterna och från komplexiteten av systemets dynamik. I denna avhandling tar vi oss an dessa utmaningar och presenterar tre huvudbidrag.

Vi betraktar först imperfektioner som är naturligt förekommande i trådlösa nätverk, såsom kommunikationsfördröjningar och meddelandestörningar, genom en tät samdesign. Vi tämjer dessa begränsningar i den utsträckning det är möjligt och tar itu med de återstående osäkerheterna med en lämplig kontrolldesign. På det sättet kan vi garantera stabiliteten hos det övergripande systemet och visa återkopplingskontroll över ett trådlöst multihopp-nätverk vid uppdateringsfrekvenser av 20-50 ms.

Om flera agenter använder samma trådlösa nätverk i ett trådlöst CPS är begränsad bandbredd en speciell utmaning. I vårt andra bidrag presenterar vi ett ramverk som gör det möjligt för agenter att förutsäga deras framtida kommunikationsbehov. Detta gör det möjligt för nätverket att schemalägga resurser till agenter i behov av kommunikation. På så sätt kan den begränsade kommunikationen användas på ett effektivt sätt.

Som ett tredje bidrag, för att öka flexibiliteten i nätverk introducerar vi maskininlärningstekniker. Vi presenterar två olika tillvägagångssätt. I det första tillvägagångssättet gör vi det möjligt för system att automatiskt lära sig systemdynamiken om den verkliga dynamiken avviker från den tillgängliga modellen. Således blir vi av med antagandet om att ha en exakt systemmodell tillgänglig för alla agenter. I det andra tillvägagångssättet föreslår vi ett ramverk för att direkt lära sig aktiveringsstrategier som tar hänsyn till begränsningar i bandbredd. Sådana tillvägagångssätt är helt oberoende av en systemmodell och kan enkelt utökas till icke-linjära inställningar. Därför är de också lämpliga för applikationer med komplicerad systemdynamik.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Sebastian Trimpe for his insightful feedback, his support, and for providing me the freedom to explore own research ideas. I would also like to thank my co-supervisor Prof. Karl Henrik Johansson for his guidance and for providing me the possibility to spend time at KTH.

I wish to acknowledge the work of Harsoveet Singh on the physical devices I have been working with throughout my thesis. His careful design saved me a lot of time when getting started with this project. Most of the work presented in this thesis has been carried out at the Max Planck Institute for Intelligent Systems in Germany. I would especially like to thank Alonso Marco Valle, Friedrich Solowjow, Dr. Manuel Wütrich, Dr. Michal Rolinek, Felix Grimminger, Joel Bessekon Akpo, Dr. Jia-Jie Zhu, Dr. Maximilien Naveau, and Bilal Hammoud for collaboration, insightful discussions, and nice moments outside of work and academia.

At the Automatic Control Department at KTH, I want to thank Lars Lindemann, David Umsonst, Mladen Cicic, Alexandros Nikou, and Rui Oliveira for helping me with getting started at KTH and the stimulating working environment during the time I spent there. I also want to thank Fabian Mager and Dr. Marco Zimmerling from TU Dresden and Romain Jacob from ETH Zürich for the tight collaboration.

Finally, I would like to thank my family and friends for their constant support.

The work presented in this thesis was supported in part by the German Research Foundation (DFG) within the priority programme SPP 1914 (grant TR 1433/1-1), the Cyber Valley Initiative, and the Max Planck Society.

Dominik Baumann
January 14, 2019.

Contents

Abbreviations

Table 1: Symbols and Notations

Symbol	Meaning
CPS	Cyber-Physical System
ETC	Event-Triggered Control
LQR	Linear Quadratic Regulator
ETSE	Event-Triggered State Estimation
DNN	Deep Neural Network
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
ETL	Event-Triggered Learning
DPP	Dual Processor Platform
AP	Application Processor
CP	Communication Processor
TTW	Time-Triggered Wireless
LTI	Linear Time-Invariant
DETSE	Distributed Event-Triggered State Estimation
KF	Kalman Filter
PDF	Probability Density Function
PT	Predictive Trigger
ST	Self Trigger

Chapter 1

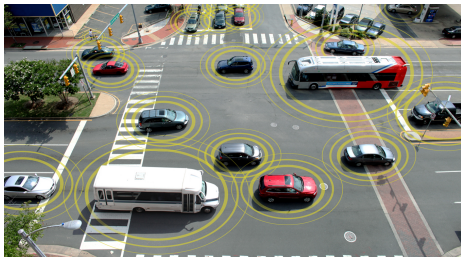
Introduction

Cyber-physical systems (CPSs) are key to many emerging applications of recent interest, but severe challenges have to be overcome to release their full potential. We start with some motivating examples for CPSs and discuss the challenges that they impose for control design. After having discussed the challenges, we present the problem setting investigated in this thesis, review related literature, and present the contributions and the outline of the thesis.

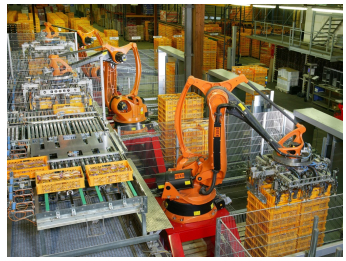
1.1 Motivation

Engineering systems that are used in industry or available for home usage today are usually tailored to specific applications in an isolated environment. Industrial robotic systems are often inside a cage to prevent them from hurting people and are unable to interact with other robots or humans. Also mobile robots that are meant for home usage, such as robotic lawn movers or vacuum cleaners, only work in specific domains, are unable to communicate, and easy to trick. In contrast to that, future engineering systems will be in tight connection with the surrounding world. These CPSs will be connected with each other and with the Internet, often called the *Internet of Things*, and will be able to act autonomously in the real world. The connection with each other will enable CPSs to work collaboratively and coordinate their actions. While sharing information between different CPSs is clearly beneficial for collaborative tasks, it also makes a lot of data available for each single agent. This is further amplified through the connection to the Internet and can be exploited by *learning* from data. If the available data is used for learning, CPSs can be enabled to improve their behavior over time or adapt in case the environmental conditions change. The ability to communicate also opens the possibility to carry out heavy computations that are often needed for learning at cloud computing services. In the following, we will highlight some examples of CPSs that are expected to have high impact in the future.

Autonomous cars, depicted in Figure 1.1a, have been a field of growing interest since the end of the last century. If autonomous cars are able to interact with



(a) Autonomous cars on a road [U.S. Department of Transportation].



(b) Factory automation [Kuka Roboter GmbH].

Figure 1.1: Two examples of CPSs.

each other, they can, for instance, share information about planned paths, what would allow for adaptive traffic control. Adaptive traffic control is expected to lead to reduced fuel consumption and fewer traffic jams. As a concrete example, connected vehicles, sometimes referred to as the *Internet of vehicles*, can form platoons with short inter-vehicle distance. Platooning of autonomous cars has been widely researched and its potential, e.g., in terms of fuel savings, has been shown [1]. Models of vehicle dynamics and their fuel consumption, which are used for planning in such frameworks, can become complex. Therefore, using data to improve the models or to find optimal actuation strategies would be beneficial. Moreover, the dynamics might change over time due to replacement of parts or because of external factors, such as different load or road conditions. Learning from data would eliminate the need to reprogram controllers or foresee all possible scenarios at design time.

In smart factories (see Figure 1.1b), robotic systems are expected to interact with each other and with human collaborators. The connection of plants with remote control stations, where humans can influence the processes, through wired bus networks, is already common practice in process industry today. However, cable-based solutions limit the flexibility and increase the installation and maintenance cost of the overall system. Cables are for instance subject to wearout and will break after sufficient time. This leads to errors that are difficult to find and lower productivity. Future smart factories that will be connected over wireless networks, do not encounter such problems. The availability of data will further advance smart factories. Today, controllers in process industry are typically tuned manually. This does not necessarily lead to good system performance [2], so already nowadays the ability to detect bad performance and automatically retune controllers would be beneficial. Additionally, plants in process industry, as well as any other system, are subject to wearout, thus, the system dynamics also change over time. To guarantee high-quality performance throughout the whole working life of the system, this change in the dynamics has to be compensated for by retuning the controller. Moreover, in future smart factories, the specific tasks of plants will change over time and the ability to adapt to that will be necessary.

Another interesting area for CPSs is medical health-care. In traditional clinical

practice, medical devices mostly act as sensors and actuators for caregivers. If these different systems are able to communicate, the information of all sensors can be used to generate smart alarms, in contrast to nowadays threshold-based alarms, that call the caregiver and provide useful context information. Such smart alarm systems are a challenging topic, as they require the integration and filtering of multiple sensor signals that have to be used in concert with a patient model. A patient model has to be created individually for each patient and is in general not trivial to come up with. Thus, learning it from data or improving it using available data would clearly be beneficial. Going beyond smart alarms, a next step would be to use the information to automatically act, e.g., inject a drug. But this demands for a very reliable communication between devices and an accurate patient model to prevent wrong treatments.

1.2 Challenges

All of the above mentioned examples represent interesting application areas of wireless CPSs, but current technology is lacking the reliability and flexibility to realize them.

In autonomous driving and medical health-care, safety is clearly an issue and systems need to meet strict requirements. Classical control theory usually assumes perfect communication when providing stability guarantees, but this assumption does not hold for wireless CPSs. Wireless channels are orders of magnitude less reliable than wired setups, i.e., there is a significant probability of losing messages. Especially, when message losses are correlated, providing stability guarantees becomes challenging.

When taking the example of autonomous driving or robots collaborating in a smart factory, we are dealing with fast physical systems. Such systems are difficult to control over wireless networks, as the end-to-end delay of message passing is non-negligible and subject to, possibly huge, variations. The problem of delays becomes even more apparent, if communication occurs over long distances, as is typically the case in smart factories. To cover large distances, intermediate relay nodes are necessary that retransmit messages, as the agents cannot talk with each other directly. In such a *multi-hop* network, the delay increases when more hops are added. In its first part, this thesis addresses the challenge of fast and reliable feedback control over wireless multi-hop networks.

In all of the examples described above, we have multiple agents that need to use the network for communication. This reveals another shortcoming of wireless networks. Wireless networks have bandwidth constraints, thus, not all agents may be able to communicate at the same time. In order to allocate communication slots to agents, the network needs to be informed in advance about future communication needs. Apart from bandwidth, communication is also costly in terms of energy. This is a particular challenge for wireless CPSs, as they are usually realized through embedded devices with constraints on size and weight and, therefore, limited energy

resources. For both reasons, communication should only occur when needed, and not in a time-triggered, periodic fashion as done in classical control theory. The second part of this thesis takes on the challenge of limiting communication between agents and, in particular, prediction of communication needs.

Methods that address these challenges are typically based on models of the system dynamics. However, in examples like smart factories or autonomous vehicles, we deal with many systems with potentially complex dynamics. Moreover, dynamics might change over time, e.g., due to wearout. Therefore, the assumption of always having accurate system models available is not realistic. Instead of assuming accurate models to be given, available data can be leveraged to either learn models or to directly learn actuation strategies. Learning dynamics models and control policies from data is the third challenge that this thesis aims to address.

1.3 Problem Setting

The main focus of this thesis is on feedback control of wireless CPSs. We consider a general wireless CPS as shown in Figure 1.2. While Figure 1.2 shows only one physical system, we typically consider multiple systems using the same network for communication. Each physical system is equipped with sensors and actuators and connected to a controller over a wireless network. That is, sensor signals and actuation commands need to be communicated over a wireless network, subject to delays and message losses. The first problem we consider is how to come up with suitable control laws that can deal with these network imperfections while still guaranteeing stability. If multiple systems use the same network for communication, the limited bandwidth of wireless channels becomes an issue. The second problem we consider is how to use the limited bandwidth in an efficient manner. If we have multiple systems with possibly complex dynamics, the assumption of having an accurate model for all of them is not realistic. As a third problem we will consider settings, where such accurate models are not available. In order to cope with these challenges, we will present a control design that addresses challenges imposed by the wireless network, different ways of reducing communication, and introduce learning techniques to automatically learn system dynamics or arrive at control policies without the need for a dynamics model. The concrete problem formulations and contributions towards these problems will be contained in the Chapters 2 - 5. Here, we will introduce the general class of systems and setup that we consider in this thesis to provide a unifying view on the different contributions.

The physical system in Figure 1.2 is represented by a differential equation of the form

$$\mathrm{d}x(t) = f\left(x(t), u(t)\right)\mathrm{d}t + Q\,\mathrm{d}W(t), \tag{1.1}$$

with $x(t) \in \mathbb{R}^n$ the state, $u(t) \in \mathbb{R}^m$ the control input, and $W(t) \in \mathbb{R}^n$ a multi-dimensional Wiener process capturing process noise. For most parts of the thesis,

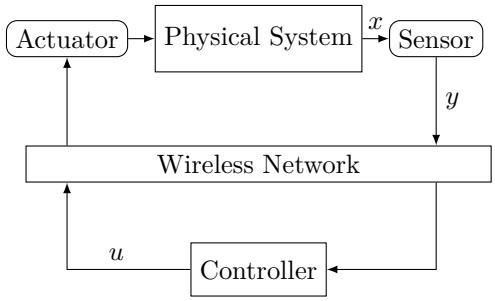


Figure 1.2: Schematic of a wireless CPS.

we restrict to linear systems, i.e.,

$$\mathrm{d}x(t) = Ax(t)\,\mathrm{d}t + Bu(t)\,\mathrm{d}t + Q\,\mathrm{d}W(t), \tag{1.2}$$

with the state transition matrix $A \in \mathbb{R}^{n\times n}$ and the input matrix $B \in \mathbb{R}^{n\times m}$. As CPSs are realized through embedded devices, control laws are implemented digitally. We will thus often consider a discretized version of (1.2),

$$x(k+1) = A_\mathrm{d}x(k) + B_\mathrm{d}u(k) + v(k), \tag{1.3}$$

with the discrete-time index k, the discrete-time process noise $v(k)$, and the discrete-time state and input matrices A_d and B_d. The index d will be dropped if clear from context.

In parts of the thesis, we will also deal with state estimation, i.e., reconstructing the state x from the measurements y in Figure 1.2. In discrete time, y is defined as

$$y(k) = C_\mathrm{d}x(k) + w(k), \tag{1.4}$$

with the measurements $y(k) \in \mathbb{R}^l$, the output matrix $C_\mathrm{d} \in \mathbb{R}^{l\times n}$, and the measurement noise $w(k) \in \mathbb{R}^l$ a Gaussian random variable with probability density function (PDF) $\mathcal{N}(w(k); 0, \Sigma_\mathrm{meas})$.

The main focus of the thesis, however, is on feedback control of wireless CPSs. For feedback control of a system as in (1.3), we will often use the linear quadratic regulator (LQR) [3]. In the LQR setting, the (time-invariant) control law $u(k) = Fx(k)$ is obtained as the *optimal* feedback controller that minimizes a quadratic cost function

$$J = \lim_{K\to\infty} \frac{1}{K}\,\mathbb{E}\Big[\sum_{k=0}^{K-1} x(k)^\mathrm{T}Qx(k) + u(k)^\mathrm{T}Ru(k)\Big]. \tag{1.5}$$

The positive definite matrices Q and R are design parameters, which represent the designer's trade-off in achieving a fast response (large Q) or low control energy (large R).

The first problem we consider is how to come up with provably stable control laws when messages are sent over wireless channels, subject to delays and message losses. In classical control theory, communication is usually assumed to be perfect. That is, the control input $u(k)$, as also presented in (1.5), depends on the system state $x(k)$ at the same time instant and messages sent between system and controller always arrive, i.e., communication delays and message losses are neglected. As discussed in Section 1.2, both assumptions do not hold if communication happens over wireless networks as illustrated in Figure 1.2. We will, thus, present a suitable control strategy, taking into account network imperfections, and prove stability of the overall system in Chapter 2.

We will then, as a second problem, consider the limited bandwidth of wireless channels. In control, we typically use time-triggered control laws, i.e., $t_{k+1} = t_k + T$ with a constant sampling time T. Wireless communication channels have limited bandwidth, thus, if all systems transmit information at high rates, this will overload the channel and in turn lead to higher transmission delays and higher probability of message losses [4]. Moreover, CPSs usually consist of battery-powered embedded devices with constraints on size and weight. As communication is also costly in terms of energy, frequent communication lowers the lifetime of those systems. We will therefore turn the attention to event-triggered control (ETC), where $t_{k+1} = \inf\{t > t_k | \mathcal{C}(x(t), x(t_k)) \geq 0\}$, with a cost function $\mathcal{C}$, i.e., we only communicate in case of an event (e.g., some error growing too large). As discussed in Section 1.2, making instantaneous decisions about communication may not be sufficient, as the communication system then does not have the possibility to reschedule unused resources. Thus, we will present with triggering laws that at time t_k decide about communication demands at time $t_{k+\mathrm{M}}$, with $M > 0$ in Chapter 3.

The third problem we consider is how to come up with control laws that respect the limited resource communication when accurate system models are not available. The LQR presented in (1.5) depends on the system matrices A and B from (1.2). It is a typical assumption in linear feedback control that these matrices are known. If we, for example, look at large-scale factory automation systems, where we have a lot of plants, manually deriving system matrices for each of them becomes infeasible. Moreover, as already motivated, they may also change over time. We will thus investigate learning approaches that allow us to 1) automatically identify the system matrices from (1.2); 2) automatically learn a control policy that does not depend on a system model and therefore is not restricted to the linear case, but applicable to complex, nonlinear systems (1.1). These approaches will be presented in Chapters 4 and 5, respectively.

1.4 Literature Overview

Cyber-physical systems are a topic of emerging interest and have drawn increasing attention both in academia and industry due to their potential benefits to society, economy, and environment [5–7]. Application areas are broad, as discussed above,

and include for instance autonomous driving [1, 8], factory automation [9, 10], and health-care systems [11, 12]. In the following subsections, we provide an overview over literature related to the topics of the thesis, covering literature on wireless CPSs (Section 1.4.1), resource savings through ETC (Section 1.4.2), and approaches for combining machine learning with control theory, with focus on ways of using machine learning in settings with limited bandwidth (Section 1.4.3).

1.4.1 Wireless Cyber-Physical Systems

Control systems that are connected over a communication network, also named networked control systems, have received considerable attention in literature, see for instance [13, 14] and references therein for an overview. Major concerns in networked control systems are transmission delays and the unreliability, i.e., the non-negligible probability of message losses, of wireless networks.

The control community has extensively studied design and stability analysis for different architectures, delay models, and message loss processes [15–19]. Toolboxes have been developed to evaluate control designs in simulation based on an abstract model of an imperfect network [20, 21]. Similarly, co-design based on an integration of control and real-time scheduling theory [22] and formal analysis of closed-loop properties using hybrid automata modeling physical, control, and network-induced timing aspects [23] have been proposed.

Turning to the sensor network, embedded, and real-time communities, we find work on how to achieve real-time communication across distributed, unreliable, and dynamic networks of resource-constrained devices [24]. Early efforts based on asynchronous multi-hop routing provide soft guarantees on end-to-end message deadlines [25, 26]. Solutions from industry and academia have been proposed [27–30] and analyzed [31–33], targeting real-time monitoring in static networks with a few sinks. Using a flooding-based approach, real-time communication in dynamic networks with any number of sinks has been demonstrated [34]. The problem of lifting real-time guarantees from the network to the application level is studied in [35], but the achievable end-to-end latencies on the order of seconds are too long for emerging closed-loop control applications [36].

Co-design of control and routing based on WirelessHART has been studied in simulation [37, 38]. While [37] focuses on the impact of the routing strategy on control performance, the work in [38] proposes to adapt the network protocol at runtime in response to changes in the state of the physical system.

Practical efforts on control over wireless fall in two categories. First, multi-hop solutions based on low-power 802.15.4 devices exist for physical systems with *slow dynamics* achieving update intervals on the order of seconds, such as adaptive lighting in road tunnels [39] and power management in data centers [40]. Second, solutions for physical systems with *fast dynamics* providing update intervals below 100 ms are exclusively based on single-hop networks of 802.11 [41, 42], Bluetooth [43], or 802.15.4 [21, 44] devices.

This leaves a gap, as results for feedback control with update intervals below

100 ms over multi-hop networks have not been reported yet. We will fill this gap in Chapter 2, where we demonstrate feedback control over a multi-hop network with update intervals of 20-50 ms. Apart from the practical demonstration, we also provide theoretical stability guarantees.

1.4.2 Event-Triggered State Estimation and Control

The above literature mainly covers challenges introduced through the unreliability of wireless networks, but leaves out the fact that communication is a scarce resource in wireless CPSs. This problem is addressed by event-triggered methods. Because of the promise to achieve high-performance control on resource-limited systems, the area of ETC and event-triggered state estimation (ETSE) has seen substantial growth in the last decades. For general overviews, see [45–48] for control and [45, 49–51] for state estimation.

Especially in early works on ETC, impulse control has often been considered, see for instance [52–54]. Event-triggered impulse control can be regarded as a replacement for periodic proportional controllers. The problem of finding a suitable replacement for the integral part that is often used in periodic control to cope for instance with load disturbances, has also been addressed. In [55], a disturbance observer is used. A typical example for a periodic controller that combines proportional and integral part are PID-controllers, which are the most common controllers used in industry. Event-triggered PID-control has also been investigated starting from [56]. A particular problem here is the replacement of the integral part of the PID-controller [57]. Mostly, a network between sensor and controller is considered, thus, the main problem for the integral part is the non-constant sampling time of the event-triggered mechanism. In [58] this is dealt with by explicitly taking into account the actual sampling time instead of assuming a nominal, constant sampling time. A different approach is presented in [59], where the event detector is connected to the sensor. Instead of looking at the absolute value of the integrator, the difference between current value and the value at the last triggering instant is used to trigger communication, as a constant value of the integrator indicates a control error of zero. For more advanced ETC techniques, we refer the reader to [45–48].

Also for ETSE various design methods have been proposed in literature, and, in particular, for its core components, the estimation algorithms and event triggers. For the former, different types of Kalman filters [60–62], modified Luenberger-type observers [63, 64], and set-membership filters [65, 66] have been used, for example. Variants of event triggers include triggering based on the innovation [60, 67], estimation variance [61, 68], or entire PDFs [69]. In these works it has been shown that high performance can be achieved with a significantly reduced amount of samples. However, the triggers proposed therein make instantaneous transmit decisions, i.e., there is no time for the communication system to reschedule resources.

The concept of *self triggering* has been proposed [70] to address the problem of predicting future sampling instants. In contrast to event triggering, which requires the continuous monitoring of a triggering signal, self-triggered approaches predict

the next triggering instant already at the previous trigger. Several approaches to self-triggered control have been proposed in literature (e.g., [46, 71–73]). Self triggering for state estimation has received considerably less attention. Some exceptions are discussed next.

Self triggering is considered for set-valued state estimation in [74], and for high-gain continuous-discrete observers in [75]. In [74], a new measurement is triggered when the uncertainty set about some part of the state vector becomes too large. In [75], the triggering rule is designed so as to ensure convergence of the observer. The recent works [76] and [77] propose self triggering approaches, where transmission schedules for multiple sensors are optimized at a-priori fixed, periodic time instants. While the re-computation of the schedule happens periodically, the transmission of sensor data does generally not. In [78], a discrete-time observer is used as a component of a self-triggered output feedback control system. Therein, triggering instants are determined by the controller to ensure closed-loop stability.

In Chapter 3, we take on the challenge of predicting future communication demands. We propose the predictive trigger, which continuously monitors the trigger signal, but still makes communication decisions ahead of time. We show that the performance of the predictive trigger is between the known concepts of self triggering and event triggering.

1.4.3 Learning Resource-Aware Control

Using machine learning techniques to learn feedback controllers from data has been considered in previous works, see e.g., [79–90] and references therein. These works typically consider learning of control policies only, without incorporating the cost of communication such as when controller and plant are connected over a network link.

Model-free reinforcement learning (RL) for event-triggered controllers has for example been proposed in [91], where an actor-critic method is used to learn an event-triggered controller and stability of the resulting system is proved. However, the authors consider a predefined communication trigger (a threshold on the difference between current and last communicated state); that is, they do not learn the communication policy from scratch. Similarly, in [92], an approximate dynamic programming approach using neural networks is implemented to learn event-triggered controllers, again with a fixed error threshold for triggering communication. In [93], the authors propose an algorithm to update the weights of a neural network in an event-triggered fashion. Model-based RL is used in [94] to simultaneously learn an optimal event-triggered controller with a predefined fixed communication threshold, and a model of the system. In [95], an architecture for control of interconnected systems using RL is proposed. There, the focus is on increasing the efficiency of learning algorithms that only get feedback at event times. The algorithms are independent of the triggering condition.

Solving scheduling problems with deep reinforcement learning (DRL) has been proposed in [96]. Given M agents that use the same communication network, which supports simultaneous communication of N agents, where $N < M$, the algorithm

assigns communication slots to the agents.

The recent work [97] uses learning to improve communication behavior for ETSE. There, the idea is to improve accuracy of state predictions through model-learning. A second event-trigger is introduced that triggers learning experiments only if the mathematical model deviates from the real system.

In Chapters 4 and 5, we will present two different approaches how learning can be used for ETC. As a first approach we will, similar as in [97], introduce a second trigger to trigger learning experiments, but here in the context of event-triggered pulse control. Moreover, we will show, how we can extend this approach to cope with load disturbances and thus replace the integrator from periodic control, a particular challenge for ETC as discussed in Section 1.4.2. As a second approach, we will demonstrate how DRL can be used to learn event-triggered controllers. Other than existing approaches, we will learn both, the control law and the triggering condition, simultaneously.

1.5 Thesis Outline and Contributions

The thesis is subdivided into four main parts in Chapters 2 - 5. These are next described in more detail.

Chapter 2

The first part of the thesis takes on the challenges imposed by using wireless technology for control. Through a tight integration at design time, we present an approach that enables fast closed-loop control over low-power wireless networks. We give theoretical stability guarantees and demonstrate the feasibility of the approach on a real testbed, consisting of physical systems and a low-power multi-hop network. Like that, we demonstrate for the first time feedback control over low-power multi-hop networks with update rates of 20-50 ms. Moreover, we show that our design is flexible enough to also deal with synchronization tasks in a straightforward manner. This part is based on the following contributions:

- Dominik Baumann[1], Fabian Mager[1], Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe, “Fast feedback control over low-power wireless with guaranteed stability and mode changes”, *in preparation.*

- Fabian Mager[1], Dominik Baumann[1], Romain Jacob, Lothar Thiele, Sebastian Trimpe, and Marco Zimmerling, “Feedback control goes wireless: Guaranteed stability over low-power multi-hop networks”, The 10th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS), Montreal, Canada, 2019, *accepted.*

[1] Equal Contribution

- Dominik Baumann[1], Fabian Mager[1], Harsoveet Singh, Marco Zimmerling, and Sebastian Trimpe, "Evaluating low-power wireless cyber-physical systems", IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench), Porto, Portugal, 2018.

Chapter 3

After having shown that through integration at design time we are able to achieve provably stable closed-loop control over wireless networks, we next look at the problem of limited bandwidth. Existing approaches for limiting the number of communication slots typically take instantaneous decisions about whether to send information or not. Different from these approaches, we present a framework that predicts future communication demands in advance and, therefore, allows the communication system to reschedule resources. Having knowledge about future communication demands, network resources can be rescheduled. This part is based on the following contribution:

- Sebastian Trimpe and Dominik Baumann, "Resource-aware IoT control: Saving communication through predictive triggering", IEEE Internet of Things Journal, *accepted*.

Chapter 4

The approaches until here demand for an accurate dynamics model of the system to be controlled. In Chapter 4, we drop the assumption of having such a model, but look at the system performance to detect, whether the current model is accurate or a new model needs to be learned. We further propose a new design for event-triggered pulse control that takes into account load disturbances, thus, replacing the integral part of periodic controllers, and input saturations. This part is based on the following contribution:

- Dominik Baumann, Friedrich Solowjow, Karl H. Johansson, and Sebastian Trimpe, "Event-triggered pulse control with adaptation through learning", The American Control Conference (ACC), Philadelphia, Pa, USA, 2019, *under review*.

Chapter 5

In Chapter 5, we propose *end-to-end* learning of resource-aware controllers, as an alternative to model-based control strategies. That is, we do not design a specific control strategy, but include the task of saving resources in the reward function of a RL algorithm. Different than other approaches for learning resource-aware controllers, we do not assume a fixed triggering rule, but learn communication strategy and control policy simultaneously. A main advantage of this approach is

that it straightforwardly generalizes to nonlinear settings. This part is based on the following publication:

- Dominik Baumann[1], Jia-Jia Zhu[1], Georg Martius, and Sebastian Trimpe, "Deep reinforcement learning for event-triggered control", The 57th IEEE International Conference on Decision and Control (CDC), Miami Beach, Fl, USA, 2018.

The last chapter, i.e., Chapter 6, concludes this thesis and gives an outline of work that is already ongoing or work that is planned in the near future.

1.5.1 Contributions by the author

As pointed out above, this thesis is based on several papers, or papers under submission, by the author of this thesis and different co-authors. The order of the authors in the mentioned papers generally reflects the workload and contributions of the authors (first author being the main contributor). However, in all the listed publications, all authors contributed and were actively involved in formulating the problems, developing the solutions, evaluating the results, and writing the paper. For the papers that are the basis of Chapters 2 and 5, the first two authors contributed equally.

Chapter 2

Feedback Control with Guaranteed Stability over Wireless Multi-Hop Networks

As discussed in the previous chapter, the interconnection of CPSs over wireless networks has a lot of benefits. But at the same time, the introduction of wireless technology poses severe challenges for control design. Current solutions of wireless CPSs are not able to stabilize systems that require update intervals below 100 ms over multi-hop networks. In this chapter, we will show, how imperfections of wireless communication can be tamed and addressed through a tight co-design of communication and control strategy. That way, we will come up with a design that enables for the first time fast feedback over low-power wireless multi-hop networks with update intervals of 20-50 ms. Stability of the overall system will be proved formally and demonstrated on a cyber-physical testbed.

2.1 Introduction

CPSs use embedded computers and networks to monitor and control physical systems [98]. While monitoring using *sensors* allows, for example, to better understand environmental processes [99], it is control and coordination through *actuators* what nurtures the CPS vision of robotic materials [100], smart transportation [1], multi-robot swarms for disaster reponse and manufacturing [101], etc.

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 across large spaces, and attached to devices with size, weight, and cost constraints. Low-power wireless multi-hop communication provides the cost efficiency and flexibility to overcome this hurdle [102, 103] if two requirements are fulfilled. First, fast feedback is required to keep up with the dynamics of physical systems [104]; for example, robot motion control and drone swarm coordination require update intervals of tens of milliseconds [105, 106]. Second, as feedback control modifies the dynamics of physical systems [107], guaranteeing *closed-loop stability* under imperfect wireless communication is a major concern.

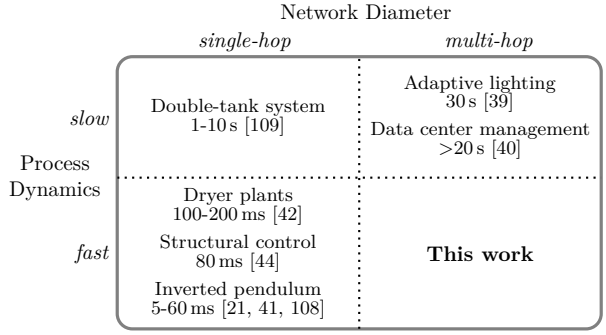


Figure 2.1: Design space of wireless CPS that have been validated on real-world devices and networks.

Hence, this chapter 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?* One of the challenges, as detailed in Section 2.2, is that even slight variations in the quality of a wireless link can trigger drastic changes in the routing topology [39]—and this can happen several times per minute [110]. Hence, to establish trust in feedback control over wireless, a real-world validation against these *dynamics* on a realistic CPS testbed is absolutely essential [102], as opposed to considering setups with a *statically configured* routing topology and only a few nodes on a desk as, e.g., in [111]. Prior works on control over wireless that validate their design through experiments on physical platforms do not provide an affirmative answer. Figure 2.1 classifies prior control-over-wireless solutions that have been validated using experiments on real devices 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 of seconds, which is typically insufficient for feedback control of, e.g., mechanical systems.

In the *single-hop/slow* category, Araujo et al. [109] 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 1 to 10 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 [43], IEEE 802.11 [41], and IEEE 802.15.4 [21, 108]. 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 [42], 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 [44].

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 [39]. 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 [40].

In contrast to these works, 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. Moverover, we provide a formal stability proof, and our solution seamlessly supports control and coordination of multiple physical systems, validated through experiments on a realistic cyber-physical testbed.

Contribution and road-map. This chapter presents the design, analysis, and real-world validation of a wireless CPS that fills the gap visualized in Figure 2.1. Section 2.2 highlights the main challenges and corresponding system design goals we need to achieve when closing feedback loops over wireless multi-hop networks. Underlying our approach is a careful co-design of the wireless embedded components (in terms of hardware and software) and the closed-loop control system, as described in Section 2.3 and Section 2.4. We tame typical wireless network imperfections, such as message losses and end-to-end communication 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 guarantee closed-loop stability for linear dynamic systems. Moreover, unlike prior work, our solution supports control and coordination of multiple physical systems out of the box—a key asset in many CPS applications [101, 105, 106].

To evaluate our design in Section 2.5, we developed a cyber-physical testbed that consists of 20 wireless embedded devices forming a 3-hop network and multiple cart-pole systems whose dynamics match a range of real-world mechanical systems [107, 112]. As such, this testbed addresses an important need in CPS research [102]. Our experiments reveal the following key findings: (*i*) two inverted pendulums can be safely stabilized by two remote controllers across the 3-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*) experiments match the theoretical results.

In summary, this chapter 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 20-50 ms.

- We formally prove that our end-to-end CPS design guarantees closed-loop stability for linear dynamic systems.

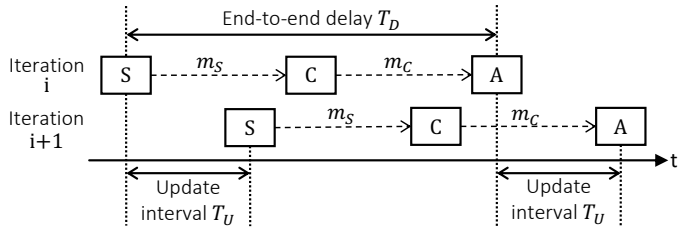


Figure 2.2: Application tasks and message transfers for a single feedback loop. *In every iteration, the sensing task (S) takes a measurement of the physical system and sends it to the control task (C), which computes a control signal and sends it to the actuation task (A).*

- Experiments on a novel cyber-physical testbed show that our solution can stabilize and synchronize multiple inverted pendulums despite significant message loss.

2.2 Problem Formulation and Approach

Scenario. We consider wireless CPSs that consist of a set of embedded devices equipped with low-power wireless radios. The devices execute different *application tasks* (i.e., sensing, control, or 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 2.2 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 [113]. 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 [39], introducing significant jitter. Delays and jitter can both destabilize a feedback system [19, 114].

- *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 communiation 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 performance or even infeasibility of closed-loop stability [115].

- *Correlated message losses.* Message losses are a common phenomenon in wireless networks, which complicate control design. Further, due to significant correlation among the message losses [116], 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 [110, 117] and may further hinder control design and stability analysis [14].

Approach. We adopt the following co-design approach to solve the above problems: *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);

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 linear dynamics;

G5 enable an efficient implementation of the control logic on state-of-the-art low-power embedded devices;

G6 use support for arbitrary traffic patterns for straightforward distributed control and multi-agent coordination.

2.3 Wireless Embedded System Design

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

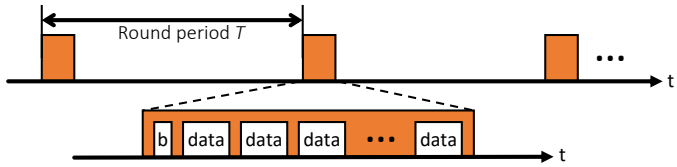


Figure 2.3: Operation of low-power wireless protocol.

1) a *low-power wireless protocol* providing multi-hop many-to-all communication with bounded end-to-end delay and accurate network-wide time synchronization;

2) a *hardware platform* that enables an efficient, predictable 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 communication energy costs.

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

2.3.1 Low-power Wireless Protocol

To support arbitrary traffic patterns (**G2**), we need 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 [118] can meet these requirements with high efficiency (**G3**) in the face of wireless dynamics (**G2**) [34]. Thus, similar to other recent proposals [119, 120], 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 2.3, 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 the theoretical minimum latency for one-to-all flooding at a reliability above 99.9 %, operates independently of the time-varying network topology, and provides microsecond-level network-wide time synchronization [118]. Nodes exploit the accurate time synchronization to sleep as long as possible between rounds and to awake in time for the next round, as specified by the round period T. A *beacon* slot *(b)* initiated by a dedicated node is used for synchronization at the beginning of each round.

As detailed in Section 2.3.3, we compute the communication schedules offline based on the traffic demands, and distribute them to all nodes before the application

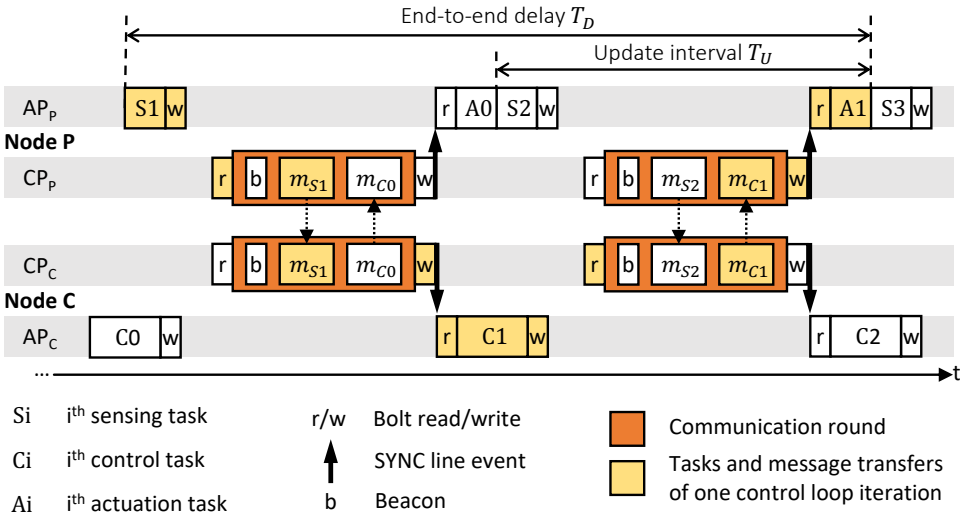


Figure 2.4: Example schedule of application tasks and message transfers between two DPP nodes C and P.

operation starts. A schedule includes the assignment of messages to *data* slots in each round (see Figure 2.3) and the round period T. Using static schedules brings several benefits. We can a priori verify if closed-loop stability can be guaranteed for the achievable latencies (see Section 2.4). Moreover, compared to prior solutions [34, 119, 120], 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).

2.3.2 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 needs 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 operations related to sensing and control. The CP has a low-power microcontroller and a radio operating at 250 kbit s^{-1} in the 868 MHz band.

AP and CP are interconnected using Bolt [121], an ultra-low-power processor interconnect that supports asynchronous bi-directional 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. Locally, AP and CP must also be synchronized to minimize end-to-end delays and jitter between 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 schedules 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.

2.3.3 Scheduling Framework

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

Figure 2.4 shows a possible schedule of the application tasks and message transfers. After sensing (S1), the $\mathrm{AP_P}$ writes a message containing the sensor reading into Bolt (w). $\mathrm{CP_P}$ reads out the message (r) before the communication round in which that message (m_{S1}) is sent using the wireless protocol. $\mathrm{CP_C}$ receives the message and writes it into Bolt. After reading out the message from Bolt, $\mathrm{AP_C}$ computes the control signal (C1) and writes a message containing it into Bolt. The message (m_{C1}) is sent to $\mathrm{CP_P}$ in the next round, and then $\mathrm{AP_P}$ 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 ($m_{S1}\, m_{C1}$, ...), this 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 schedule in Figure 2.4, T_U is exactly half the end-to-end delay T_D.

In general, the scheduling problem entails computing the communication schedules and the offsets with which all APs and CPs perform wireless communication, application tasks, message transfers over Bolt, and SYNC line assertion. The problem gets very complex for any realistic scenario with more nodes or multiple feedback loops that are closed over the same network, so solving it must be automated.

To this end, we use time-triggered wireless (TTW) [122], 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 2.2. Based on an integer linear program, it computes all communication schedules and offsets. 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., the update interval T_U as for the schedule in Figure 2.4).

2.3.4 Essential Properties and Jitter Analysis

The presented wireless embedded system design provides the following 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 $\pm 50\,\mu s$. It holds $T_D = 2T_U$.

P2 Statistical analysis of millions of Glossy floods [123] and percolation theory for time-varying networks [124] have shown that the spatio-temporal diversity in a 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 is typically above 99.9 % [118].

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.

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

$$|\,J\,| \leq 2\left(\hat{e}_{ref} + \hat{e}_{SYNC} + \widetilde{T}_{end}\,(\hat{\rho}_{AP} + \hat{\rho}_{CP})\right) + \hat{e}_{task} \tag{2.1}$$

where each term in (2.1) is detailed below.

1) Time synchronization error between CPs. Using Glossy, each CP computes an estimate $\hat{t}_{ref}$ of the reference time [118] 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 [118], 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 $-7.1\,\mu s$ and $8.6\,\mu s$. We thus consider $\hat{e}_{ref} = 10\,\mu s$ 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 $\hat{e}_{SYNC} = (2f_{AP})^{-1}$, where $f_{AP} = 48\,\text{MHz}$ is the frequency of APs clock that can detect SYNC line events on both falling and rising edges.

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 such as manufacturing process, temperature, and aging lead to different frequency drifts ρ_{CP} and ρ_{AP}. State-of-the-art clocks, however, drift by at most $\hat{\rho}_{CP} = \hat{\rho}_{AP} = 50\,\text{ppm}$ [125].

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 T_U and 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, actuation in our experiments has a jitter of $\pm 3.4\,\mu\text{s}$. To be safe, we consider $\hat{e}_{task} = 10\,\mu\text{s}$ for our analysis.

Using (2.1) 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 chapter requires $\widetilde{T}_{end} = T_D = 2T_U \leq 100\,\text{ms}$, which gives a worst-case jitter of $\pm 50\,\mu\text{s}$, as stated in **P1**.

2.4 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 2.2. Because the wireless system supports arbitrary traffic patterns (**P3**), various control tasks can be solved including 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 over wireless and synchronization of multiple agents as prototypical examples for both the single- and multi-agent case. For stabilization, modeling and control design are presented in Section 2.4.1 and Section 2.4.2, thus achieving **G5**. The stability analysis is provided in Section 2.4.3, which fulfills **G4**. Synchronization is discussed in Section 2.4.4, highlighting support for straightforward distributed control **G6**.

2.4.1 Model of Wireless Control System

We address the remote stabilization task depicted in Figure 2.5 (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 [126]. We consider stochastic linear time-invariant (LTI) dynamics for the physical process as expressed by (1.3).

We assume that the full system state $x(k)$ can be measured through appropriate sensors, but is corrupted by Gaussian nosie. Thus, we have (1.4) with $C = I$, i.e.,

$$y(k) = x(k) + w(k). \tag{2.2}$$

If the complete state vector cannot be measured directly, it can typically be reconstructed via state estimation techniques [107].

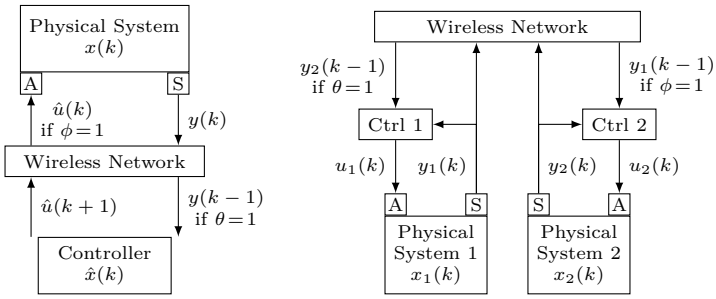


Figure 2.5: Considered wireless control tasks: stabilization (left) and synchronization (right). *The feedback loop for stabilizing the physical system (left) is closed over the (multi-hop) low-power wireless network, which induces delay and message losses (captured by i.i.d. Bernoulli variables θ and ϕ). Two physical systems, each with a local controller (Ctrl), are synchronized over the wireless network (right).*

The process model is stated in discrete time. This representation is particularly suitable here as the wireless 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 [22, p. 48]. Thus, $u(k)$ and $y(k)$ in (1.3) and (2.2) represent sensing and actuation at periodic intervals T_U as in Figure 2.4.

As shown in Figure 2.5, 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 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; i.e.,

$$u(k) = \phi \hat{u}(k) + (1 - \phi)\, u(k-1). \tag{2.3}$$

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 2.4.4.

2.4.2 Controller Design

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

Nominal design. Assuming ideal measurements, we have $y(k)=x(k)$. 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 optimal control such as the *linear quadratic regulator* (LQR) [3, 107]. Under the assumption of controllability [107], desired (in particular, stable) dynamics can be obtained for the state (1.3) in this way.

Actual design. We augment the nominal state-feedback design to cope with non-idealities, in particular, delayed measurements and message losses as shown in Figure 2.5 (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:

$$\begin{aligned}\hat{x}(k) &= \theta A y(k-1) + (1-\theta)(A\hat{x}(k-1)) + B\hat{u}(k-1) \\ &= \theta A x(k-1) + (1-\theta)A\hat{x}(k-1) + B\hat{u}(k-1) + \theta A w(k-1),\end{aligned} \tag{2.4}$$

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 (2.4) 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,

$$\hat{u}(k+1) = F\left(A\hat{x}(k) + B\hat{u}(k)\right), \tag{2.5}$$

where F is as in the nominal design. The input $\hat{u}(k+1)$ is then transmitted over the wireless network (see Figure 2.5).

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

2.4.3 Stability Analysis

We now present a stability proof for the closed-loop system given by the dynamic system of Section 2.4.1 and the proposed controller from Section 2.4.2. Because the model in Section 2.4.1 incorporates the physical process and the essential properties of the wireless embedded system, we will thus achieve goal **G4**.

While the process dynamics (1.3) are time invariant, the message losses introduce time variation and randomness into the system dynamics. Therefore, we will leverage stability results for linear, stochastic, time-varying systems [127]. For ease of presentation, we will consider (1.3) and (2.2) without process and measurement noise ($v(k) = 0$ and $w(k) = 0$), and comment later on extensions. We first introduce required definitions and preliminary results, and then apply these to our problem.

Consider the system

$$z(k+1) = \tilde{A}(k)z(k) \tag{2.6}$$

with state $z(k) \in \mathbb{R}^n$ and $\tilde{A}(k) = \tilde{A}_0 + \sum_{i=1}^{L} \tilde{A}_i p_i(k)$, where $p_i(k)$ 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 stability notion for stochastic systems like (2.6) is mean-square stability:

Definition 2.1 ([127, p. 131])**.** Let $Z(k) := \mathbb{E}[z(k)z^{\mathrm{T}}(k)]$ denote the state correlation matrix. The system (2.6) 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 initial state. MSS implies, for example, that $z(k) \to 0$ almost surely as $k \to \infty$, [127, p. 131].

In control theory, linear matrix inequalities (LMIs) are often used as computational tools to check for system properties such as stability (see [127] for an introduction and details). For MSS, we shall employ the following LMI stability result:

Lemma 2.1 ([127, p. 131])**.** *System* (2.6) *is MSS if, and only if, there exists a positive definite matrix* $P > 0$ *such that*

$$\tilde{A}_0^{\mathrm{T}} P \tilde{A}_0 - P + \sum\nolimits_{i=1}^{N} \sigma_{p_i}^2 \tilde{A}_i^{\mathrm{T}} P \tilde{A}_i < 0. \tag{2.7}$$

We will now apply this result to the system and controller from Section 2.4.1 and Section 2.4.2. The closed-loop dynamics are given by (1.3) and (2.2)–(2.5), which

we rewrite as an augmented system

$$\underbrace{\begin{pmatrix} x(k+1) \\ \hat{x}(k+1) \\ u(k+1) \\ \hat{u}(k+1) \end{pmatrix}}_{z(k+1)} = \underbrace{\begin{pmatrix} A & 0 & B & 0 \\ \theta A & (1-\theta)A & 0 & B \\ 0 & \phi FA & (1-\phi)I & \phi FB \\ 0 & FA & 0 & FB \end{pmatrix}}_{\tilde{A}(k)} \underbrace{\begin{pmatrix} x(k) \\ \hat{x}(k) \\ u(k) \\ \hat{u}(k) \end{pmatrix}}_{z(k)}. \tag{2.8}$$

The system has the form of (2.6); the transition matrix depends on θ and ϕ, and thus on time (omitted for simplicity). We can thus apply Lemma 2.1 to obtain our main stability result, whose proof is given in the appendix.

Theorem 2.1. *The system* (2.8) *is MSS if, and only if, there exists a* $P > 0$ *such that* (2.7) *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 2.1, we can analyze stability for any concrete physical system (1.3), a state-feedback controller F, and probabilities μ_θ and μ_ϕ. Searching for a $P > 0$ that satisfies the LMI (2.7) can be done using efficient numerical tools based on convex optimization (e.g., [128]). 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 $Z(k)$. Such an analysis, taking into account process and measurement noise, is shown in the appendix. 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.

2.4.4 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 [129, 130]. 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 2.3 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 2.5, right). Such schemes have been used before for wired-bus networks (e.g., in [112]).

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 (1.3) and (2.2), 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 occurs under the terms consensus or coordination [131].

We demonstrate feasibility of synchronization with multiple systems in Section 2.5.3. The synchronizing controller is based on an LQR [3]; details of the concrete design are given in Section A.4.

2.5 Experimental Evaluation

This section uses measurements from a cyber-physical testbed (see Figure 2.6) 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 3-hop 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 of arbitrary traffic patterns.
- Our system can stabilize an inverted pendulum at update intervals of 20-50 ms. Increasing the update interval decreases the control performance, but leads to significant energy savings on the wireless communication side.
- The system is highly robust to message losses. Specifically, it can stabilize an inverted pendulum at an update interval of 20 ms despite 75 % i.i.d. Bernoulli losses and situations with bursts of 40 consecutively lost messages.
- The measured jitter on the update interval and the end-to-end delay is less than $\pm 25\,\mu s$, which validates our analysis of the theoretical worst-case jitter of $\pm 50\,\mu s$ (**P1**).

2.5.1 Cyber-physical Testbed

Realistic cyber-physical testbeds are essential for the validation and evaluation of CPS solutions [102, 132]. We developed the wireless cyber-physical testbed depicted

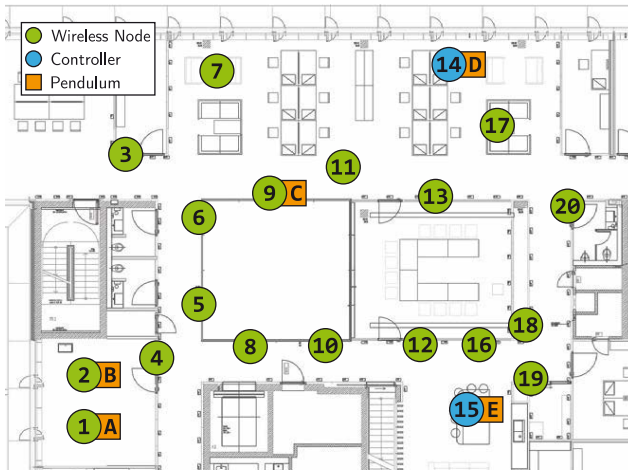


Figure 2.6: Cyber-physical testbed consisting of 20 DPP nodes that form a three-hop wireless network and five cart-pole systems (two real ones attached to nodes 1 and 2, and three simulated ones at nodes 9, 14, and 15).

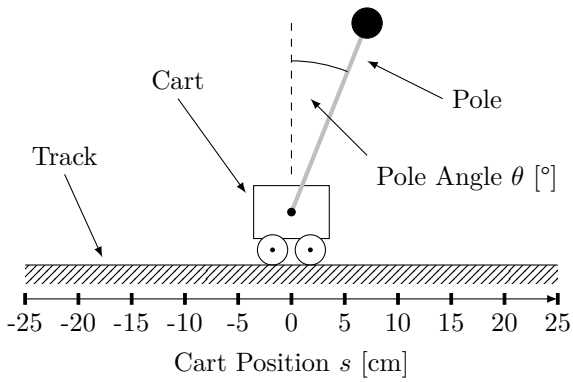


Figure 2.7: Schematic of a cart-pole system.

in Figure 2.6. 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 people's activity.

We use *cart-pole systems* as physical systems. As shown in Figure 2.7, 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 that are typical for real-world mechanical systems [133] and require 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.

The cart-pole system is subject to a few constraints. Control inputs are capped at $\pm 10\,\mathrm{V}$. The track has a usable length of $\pm 25\,\mathrm{cm}$ from the center (see Figure 2.7). 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. The appendix details the implementation of the controllers, following the design outlined in Section 2.4.2 and Section 2.4.4.

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

2.5.2 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 that run on nodes 14 and 15 to stabilize the two real pendulums A and B at $\theta = 0°$ and $s = 0\,\mathrm{cm}$. Hence, there are two independent control loops sharing the same wireless network, and it takes six hops to close each of them. We configure the wireless embedded system and the controller for an update interval of $T_U = 45\,\mathrm{ms}$. Using Theorem 2.1, we can prove stability for the overall system using $\mu_\theta = \mu_\phi = 0.999$ as per property **P2**.

Results. The experimental results confirm the theoretical analysis: We are able to safely stabilize both pendulums over the three-hop wireless network. Figure 2.8 shows a characteristic 30 s trace of one of the pendulums. Cart position, pole angle, and control input oscillate, 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 used a logic analyzer to continuously

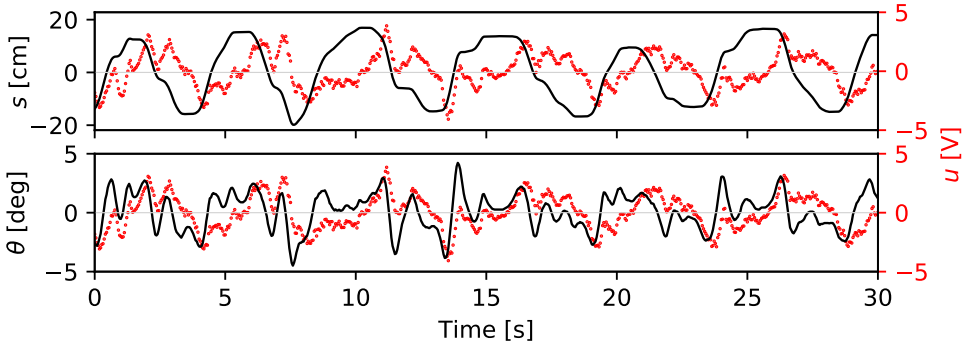


Figure 2.8: Cart position s, pole angle θ, and control input u of a cart-pole system when concurrently stabilizing two cart-pole systems over a multi-hop network.

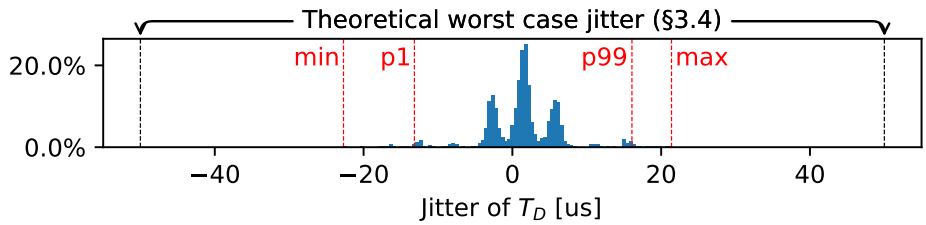


Figure 2.9: Measured jitter on the end-to-end delay T_D.

measure the update interval T_U and the end-to-end delay T_D (see Figure 2.4). Figure 2.9 shows the 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 2.3.4 and assumptions in Section 2.4.

2.5.3 Multi-hop Synchronization

We now apply the same wireless CPS design to a distributed control task to demonstrate its 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 [105]. In our experiment, stabilization runs with $T_U = 10\,\text{ms}$, and nodes 1, 2, 9, 14, and 15 exchange their current cart positions every 50 ms.

Results. The left plot in Figure 2.10 shows the cart positions without synchronization. The carts of the real pendulums move with different amplitude, phase, and frequency due to slight differences in their physics and imperfect measurements.

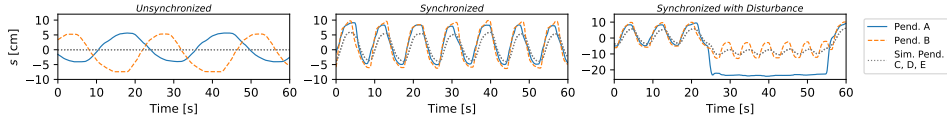


Figure 2.10: Cart positions of five cart-pole systems stabilized locally at an update interval of 10 ms and synchronizing their cart positions (middle and right plot) over the network at an update interval of 50 ms.

The simulated pendulums are perfectly balanced and behave deterministically as they all start in the same state.

In the middle plot, we see 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 $s = 0\,\text{cm}$. 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 2.10, where we hold pendulum A for some time at about $s = -20\,\text{cm}$. The other pendulums now have two conflicting control goals: stabilization at $s = 0\,\text{cm}$ and $\theta = 0°$, as well as synchronization while one pendulum is fixed at about $s = -20\,\text{cm}$. As a result, they all move towards this position and oscillate between $s = 0$ and $s = -20\,\text{cm}$. Clearly, the experiments show that the cart-pole systems influence each other, which is enabled through the network.

2.5.4 Impact of Update Interval

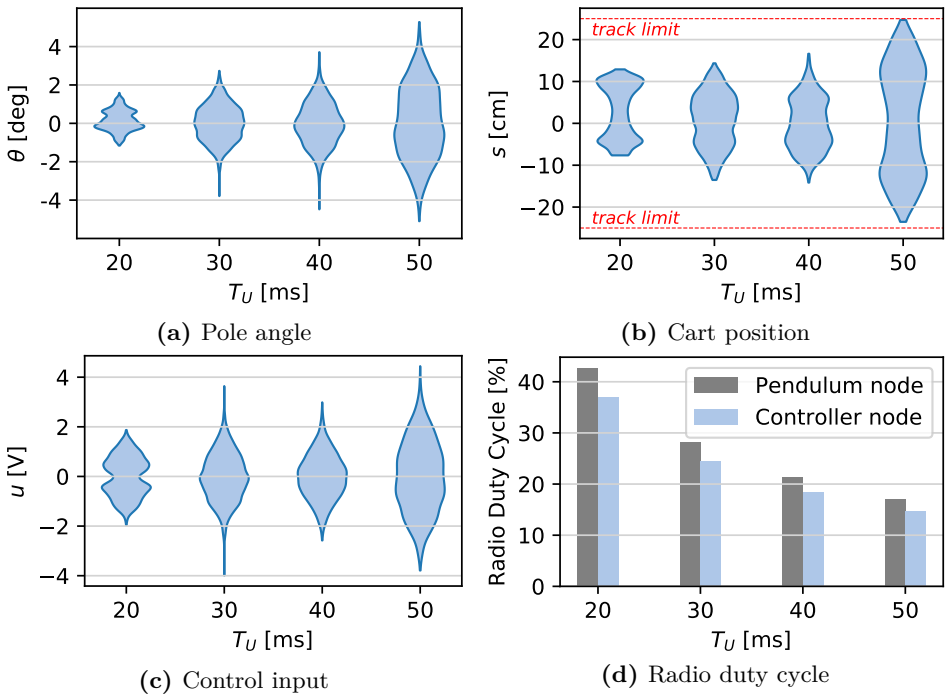
We take a closer look at the impact of different update intervals (and hence end-to-end delays) on control performance.

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 controller on node 2.

Results. Figure 2.11 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 leads to 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 about 40 % for 20 ms to about 15 % for 50 ms.

2.5.5 Resilience to Message Losses

Next, we evaluate the impact of message losses, which are a well-known phenomenon in wireless networks [134].



(a) Pole angle **(b)** Cart position

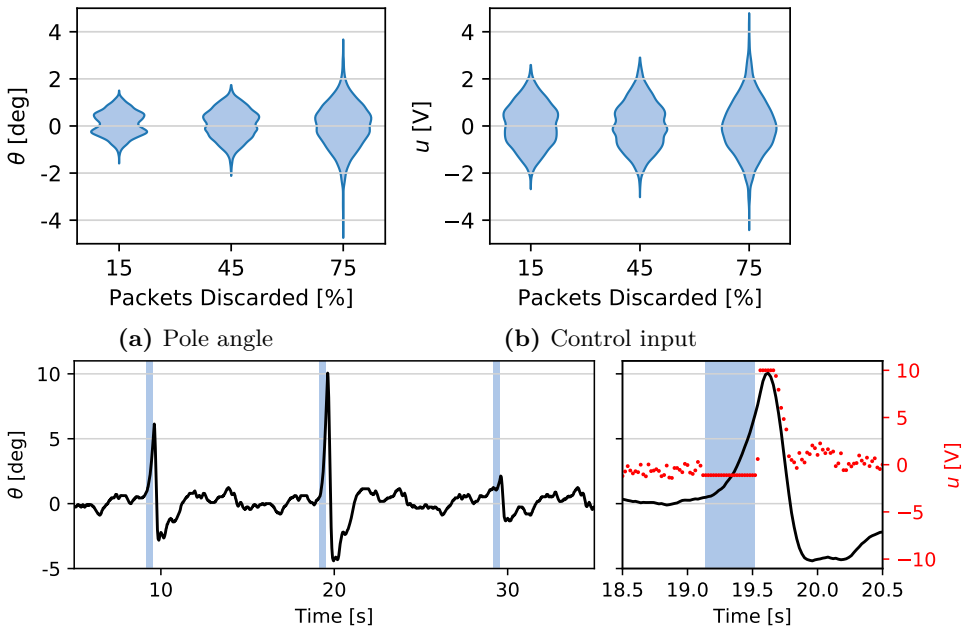
(c) Control input **(d)** Radio duty cycle

Figure 2.11: Distribution of control performance metrics and average radio duty cycle when remotely stabilizing a pendulum at different update intervals.

Setup. We use again the two-node setup from before, but now fix the update interval at 20 ms. Both nodes intentionally drop messages in two different ways. In a first experiment, they independently drop a received message according to a Bernoulli process with given failure probability. In a second experiment, they drop a certain number of consecutive messages every 10 s. This violates the i.i.d. assumption and allows us to evaluate the robustness of our control design.

Results. Figure 2.12a and Figure 2.12b show results for varying i.i.d. Bernoulli message loss rates. 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 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.

To study the impact of correlated message losses, we test different burst lengths. Figure 2.12c shows the pole angle 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 2.12c. The magnified plot shows that the pole angle diverges



(a) Pole angle **(b)** Control input

(c) Pole angle for bursts of 40 consecutive losses every 10 s (shaded areas). The right plot magnifies the second burst.

Figure 2.12: Control performance when remotely stabilizing one pendulum under artificially injected message loss, for i.i.d. Bernoulli losses (a,b) and for longer bursts of multiple consecutive losses (c).

from around 0° with increasing speed. As soon as the burst ends, the control input rises to its maximum value of 10 V to bring the pendulum back to a non-critical state, which usually takes 1-2 s.

2.6 Conclusions

In this chapter 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 chapter is an important stepping stone

toward realizing the CPS vision.

Chapter 3

Saving Communication through Predictive Triggering

In the last chapter, we demonstrated feasibility of feedback control over low-power wireless networks. But communication was running at the highest, periodic rates that the network was able to support. To enable resource savings, we will now present event-triggered methods that only communicate if needed. Existing approaches typically instantaneously decide, whether communication is necessary, thus, there is no time for the communication system to reschedule resources. In contrast to that, we will present a framework that in adavance decides about future communication needs.

3.1 Introduction

In this chapter, we will consider state estimation and control problems for distributed CPSs as discussed in Section 2.4.4. Figure 3.1 shows an abstraction of multiple CPSs connected over a communication network. We assume a network that provides many-to-all communication, i.e., a network as the one presented in the previous chapter.

Advantages and challenges of using wireless networks instead of wires to connect multiple agents have already been discussed. While the general feasibility of such approaches has been demonstrated in the previous chapter, one challenge remains open. Because the network bandwidth is shared by multiple entities, each agent should use the communication resource *only when necessary*. Developing such resource-aware control methods is the focus of this chapter. This is in contrast to traditional feedback control, where data transmission typically happens periodically at a priori fixed update rates.

Owing to the shortcomings of traditional control, *event-triggered methods* for state estimation and control have emerged since pioneering work at the beginning of the century [56, 135]. The key idea of event-triggered approaches is to apply feedback only upon certain *events* indicating that transmission of new data is necessary (e.g.,

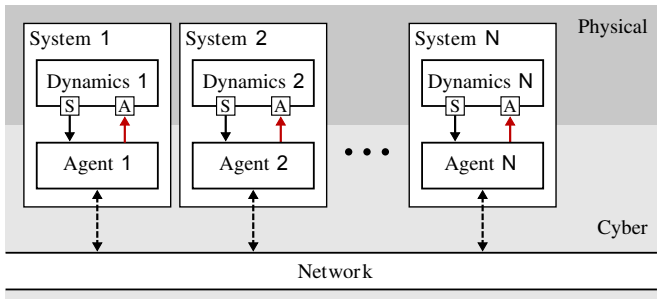


Figure 3.1: Abstraction of multiple CPSs connected over a communication network. Each *System* is composed of *Dynamics* representing its physical entity and an *Agent* representing its algorithm unit. Dynamics and Agent are interconnected via sensors (S) and actuators (A). The *Network* connects all systems.

a control error passing a threshold level, or estimation uncertainty growing too large). Core research questions concerning the design of the event triggering laws, which decide when to transmit data, and the associated estimation and control algorithms with stability and performance guarantees have been solved in recent years (see [45, 46, 48, 51] for overviews).

This chapter builds on a framework for distributed event-triggered state estimation (DETSE) developed in prior work [60, 61, 63, 64], which is applied to resource-aware control as in Figure 3.1. The key idea of DETSE is to employ model-based predictions of other systems to avoid the need for continuous data transmission between the agents. Only when the model-based predictions become too inaccurate (e.g., due to a disturbance or accumulated error), an update is sent. Figure 3.2 represents one agent of the overall system in Figure 3.1 and depicts the key components of the DETSE architecture:

- *Local control:* Each agent makes local control decisions for its actuator; for coordinated action across the network, it also needs information from other agents in addition to its local sensors.

- *Prediction of other agents:* State estimators and predictors (e.g., of Kalman filter type) are used to predict the states of all, or a subset of all agents, based on agents' dynamics models; these predictions are reset (or updated) when new data is received from the other agents.

- *Event trigger:* Decides when an update is sent to all agents. For this purpose, the local agent implements a copy of the predictor of its own behavior (*Prediction System i*) to replicate locally the information the other agents have about itself. The event trigger compares the prediction with the local state estimate: the current state estimate is transmitted to other agents only if the prediction is not sufficiently accurate.

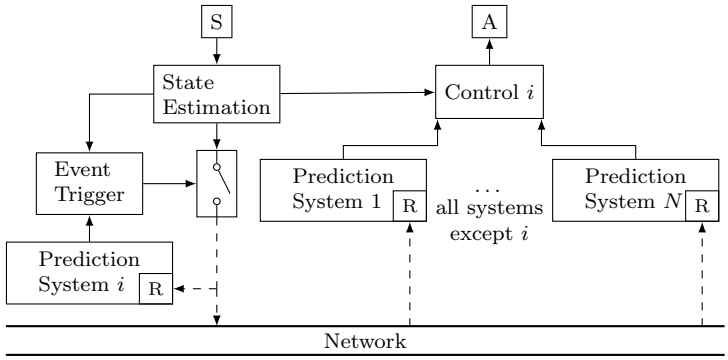


Figure 3.2: Algorithmic components implemented on each agent $i = 1, \ldots, N$ of the control system in Figure 3.1. Agent i's control decision is based on local information (*State Estimation* i) and predictions of all (or a subset of) other systems (*Prediction System 1 to N*). Each agent sends an update (*Event Trigger*) to all other agents whenever the prediction of its own state (*Prediction System* i) deviates too far from the truth, so that predictions can be reset (R).

Key benefits of this architecture are: each agent has all relevant information available for coordinated decision making, but inter-agent communication is limited to the necessary instants (whenever model-based predictions are not good enough).

Experimental studies [60, 63] demonstrated that DETSE can achieve significant communication savings, which is inline with many other studies in event-triggered estimation and control. The research community has had remarkable success in showing that the number of samples in feedback loops can be reduced significantly as compared to traditional time-triggered designs, which can be translated into increased battery life [109] in wireless sensor systems, for example. Despite these successes, better utilization of shared communication resources has typically *not* been demonstrated. A fundamental problem of most event-triggered designs (incl. DETSE) is that they make decisions about whether a communication is needed *instantaneously*. This means that the resource must be held available at all times in case of a positive triggering decision. Conversely, if a triggering decision is negative, the reserved slot remains unused because it cannot be reallocated to other users immediately.

In order to translate the reduction in average sampling rates to better *actual* resource utilization, it is vital that the event-triggered system is able to *predict* resource usage ahead of time, rather than requesting resources instantaneously. This allows the processing or communication system to reconfigure and make unneeded resources available to other users or set to sleep for saving energy. Developing such predictive triggering laws for DETSE and their use for resource-aware control are the main objectives of this chapter.

Contributions

This chapter proposes a framework for resource-aware control based on DETSE. The main contributions are summarized as follows:

1. Proposal of a Bayesian decision framework for deriving predictive triggering mechanisms, which provides a new perspective on the triggering problem in estimation;

2. Derivation of two novel triggers from this framework: the *self trigger*, which predicts the next triggering instant based on information available at a current triggering instant; and the *predictive trigger*, which predicts triggering for a given future horizon of M steps;

3. Demonstration and comparison of the proposed triggers in experiments on an inverted pendulum testbed; and

4. Simulation study of a multi-vehicle system.

The Bayesian decision framework extends previous work [49] on event trigger design to the novel concept of predicting trigger instants. The proposed self trigger is related to the concept of variance-based triggering [61], albeit this concept has not been used for self triggering before. To the best of the authors' knowledge, predictive triggering is a completely new concept in both event-triggered estimation and control. Predictive triggering is shown to reside between the known concepts of event triggering and self triggering.

3.2 Fundamental Triggering Problem

In this section, we formulate the predictive triggering problem that each agent in Figure 3.2 has to solve, namely predicting when local state estimates shall be transmitted to other agents. We consider the setup in Figure 3.3, which has been reduced to the core components required for the analysis in subsequent sections. Agent i, called *sensor agent*, sporadically transmits data over the network to agent j. Agent j here stands representative for any of the agents that require information from agent i. Because agent j can be at a different location, it is called *remote agent* here. We next introduce the components of Figure 3.3 and then make the predictive triggering problem precise.

3.2.1 Process dynamics

We consider each agent i to be governed by dynamics as described in (1.3) and (1.4),

$$x^i(k) = A_i x^i(k-1) + B_i u^i(k-1) + v^i(k-1) \tag{3.1}$$

$$y^i(k) = C_i x^i(k) + w^i(k), \tag{3.2}$$

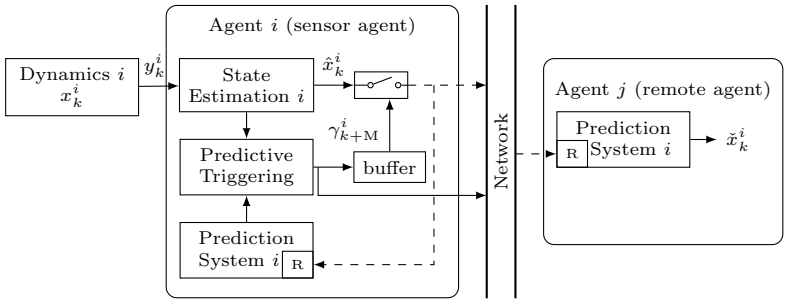


Figure 3.3: Predictive triggering problem. The sensor agent i runs a local *State Estimator* and transmits its estimate $\hat{x}^i(k)$ to the remote agent j in case of a positive triggering decision ($\gamma^i(k) = 1$). The predictive trigger computes the triggering decisions ($\gamma^i(k+M) \in \{0,1\}$) M steps ahead of time. This information can be used by the network to allocate resources. Local control (cf. Figure 3.2) is omitted here for clarity, but treated in the analysis.

where $x^i(0)$ is a mutually independent random variable with PDF $\mathcal{N}(x^i(0); \bar{x}_i, X_i)$.

Equations (3.1) and (3.2) represent decoupled agents' dynamics, which we consider in this work (cf. Figure 3.1). Agents will be coupled through their inputs (see Section 3.2.3 below). While the results are developed herein for the time-invariant dynamics (3.1), (3.2) to keep notation uncluttered, they readily extend to the linear time-variant case (i.e., A_i, B_i, C_i, Q_i, and R_i being functions of time k). Such a problem is discussed in Section 3.8.

The sets of all measurements and inputs up to time k are denoted by $\mathcal{Y}^i(k) := \{y^i(1), y^i(2), \ldots, y^i(k)\}$ and $\mathcal{U}^i(k) := \{u^i(1), u^i(2), \ldots, u^i(k-1)\}$, respectively.

3.2.2 State estimation

The local state estimator on agent i has access to all measurements $\mathcal{Y}^i(k)$ and inputs $\mathcal{U}^i(k)$ (cf. Figure 3.3). The Kalman filter (KF) is the optimal Bayesian estimator in this setting, [136]; it recursively computes the exact posterior PDF $f(x^i(k)|\mathcal{Y}^i(k),\mathcal{U}^i(k))$. The KF recursion is

$$\hat{x}^i(k|k-1) = A_i\hat{x}^i(k-1) + B_iu^i(k-1) \tag{3.3}$$

$$P^i(k|k-1) = A_iP^i(k-1)A_i^{\mathrm{T}} + Q_i =: V_{\mathrm{o}}^i(P^i(k-1)) \tag{3.4}$$

$$L^i(k) = P^i(k|k-1)C_i^{\mathrm{T}}(C_iP^i(k|k-1)C_i^{\mathrm{T}} + R_i)^{-1} \tag{3.5}$$

$$\hat{x}^i(k) = \hat{x}^i(k|k-1) + L^i(k)(y^i(k) - C_i\hat{x}^i(k|k-1)) \tag{3.6}$$

$$P^i(k) = (I - L^i(k)C_i)P^i(k|k-1). \tag{3.7}$$

where $f(x^i(k)|\mathcal{Y}^i(k-1),\mathcal{U}^i(k)) = \mathcal{N}(x^i(k); \hat{x}^i(k|k-1), P^i(k|k-1))$, $f(x^i(k)|\mathcal{Y}^i(k), \mathcal{U}^i(k)) = \mathcal{N}(x^i(k); \hat{x}^i(k), P^i(k))$, and the short-hand notation $\hat{x}^i(k) = \hat{x}^i(k|k)$ and $P^i(k) = P^i(k|k)$ is used for the posterior variables. In (3.4), we introduced the short-hand notation V_{o}^i for the open-loop variance update for later reference. We

shall also need the M-step ahead prediction of the state ($M \geq 0$), whose PDF is given by [136, p. 111]

$$f(x^i(k+M)|\mathcal{Y}^i(k),\mathcal{U}^i(k+M)) = \mathcal{N}(x^i(k+M);\, \hat{x}^i(k+M|k), P^i(k+M|k)), \tag{3.8}$$

with mean and variance obtained by the open-loop KF iterations (3.3), (3.4), i.e., $\hat{x}^i(k+M|k) = A_i^M \hat{x}^i(k) + \sum_{m=1}^{M} A_i^{M-m} B u^i(k+m-1)$ and $P^i(k+M|k) = (V_{\mathrm{o}}^i \circ \cdots \circ V_{\mathrm{o}}^i)(P^i(k))$, where '$\circ$' denotes composition. Finally, the error of the KF is defined as

$$\hat{e}^i(k) := x^i(k) - \hat{x}^i(k). \tag{3.9}$$

3.2.3 Control

Because we are considering coordination of multiple systems, the i's control input may depend on the prediction of the other systems (cf. Figure 3.2). We thus consider a control policy

$$u^i(k-1) = F_i \hat{x}^i(k-1) + \sum_{j \in \mathbb{N}_N \setminus \{i\}} F_j \check{x}^j(k-1) \tag{3.10}$$

where the local KF estimate $\hat{x}^i(k)$ is combined with predictions $\check{x}^j(k)$ of the other agents (to be made precise below), and $\mathbb{N}_N$ denotes the set of all integers $\{1, \ldots, N\}$. For coordination schemes where not all agents need to be coupled, some F_j may be zero. Then, these states do not need to be predicted.

It will be convenient to introduce the auxiliary variable $\xi^i(k) = \sum_{j \in \mathbb{N}_N \setminus \{i\}} F_j \check{x}^j(k)$; (3.10) thus becomes

$$u^i(k-1) = F_i \hat{x}^i(k-1) + \xi^i(k-1). \tag{3.11}$$

3.2.4 Communication network

Communication between agents occurs over a bus network that connects *all* systems with each other. In particular, we assume that data (if transmitted) can be received by all agents that care about state information from the sending agent:

Assumption 3.1. Data transmitted by one agent can be received by all other agents.

Such bus-like networks are common, for example, in automation industry in form of *wired* fieldbus systems [137], but are also feasible for wireless networks as shown in the previous chapter. For the purpose of developing the triggers, we further abstract communication to be ideal:

Assumption 3.2. Communication between agents is without delay and message loss.

This assumption is dropped later in the multi-vehicle simulation.

3.2.5 State prediction

The sensor agent in Figure 3.3 sporadically communicates its local estimate $\hat{x}^i(k)$ to the remote estimator, which, at every step k, computes its own state estimate $\check{x}^i(k)$ from the available data via state prediction. We denote by $\gamma^i(k) \in \{0,1\}$ the decision taken by the sensor about whether an update is sent ($\gamma^i(k) = 1$) or not ($\gamma^i(k) = 0$). For later reference, we denote the set of all triggering decisions until k by $\Gamma^i(k) := \{\gamma^i(1), \gamma^i(2), \ldots, \gamma^i(k)\}$.

The state predictor on the remote agent (cf. Figure 3.3) uses the following recursion to compute $\check{x}^i(k)$, its remote estimate of $x^i(k)$:

$$\check{x}^i(k) = \begin{cases} A_i\check{x}^i(k-1) + B_i\check{u}(k-1) & \text{if } \gamma^i(k) = 0 \\ \hat{x}^i(k) & \text{if } \gamma^i(k) = 1; \end{cases} \tag{3.12}$$

that is, at times when no update is received from the sensor, the estimator predicts its previous estimate according to the process model (3.1) and prediction of the input (3.11) by

$$\check{u}^i(k-1) = F_i\check{x}^i(k-1) + \xi^i(k-1). \tag{3.13}$$

Implementing (3.13) thus requires the remote agent to run predictions of the form (3.12) for all other systems m that are relevant for computing $\xi^i(k-1)$. This is feasible as an agent can broadcast state updates (for $\gamma^i(k) = 1$) to all other systems via the bus network. We emphasize that $\xi^i(k-1)$, the part of the input $u^i(k-1)$ that corresponds to all other agents, is known exactly on the remote estimator, since updates are sent to all agents connected to the network synchronously. Hence, the difference between the actual input (3.11) and predicted input (3.13) stems from a difference in $\hat{x}^i(k-1)$ and $\check{x}^i(k-1)$.

With (3.13), the prediction (3.12) then becomes

$$\check{x}^i(k) = \begin{cases} \bar{A}_i\check{x}^i(k-1) + B_i\xi^i(k-1) & \text{if } \gamma^i(k) = 0 \\ \hat{x}^i(k) & \text{if } \gamma^i(k) = 1; \end{cases} \tag{3.14}$$

where $\bar{A}_i := A_i + B_iF_i$ denotes the closed-loop state transition matrix of agent i. The estimation error at the remote agent, we denote by

$$e^i(k) := x^i(k) - \check{x}^i(k). \tag{3.15}$$

A copy of the state predictor (3.14) is also implemented on the sensor agent to be used for the triggering decision (cf. Figure 3.3).

Finally, we comment how local estimation quality can possibly be further improved in certain applications.

Remark 3.1. In (3.14), agent j makes a pure state prediction about agent i's state in case of no communication from agent i ($\gamma^i(k) = 0$). If agent j has additional local sensor information about agent i's state, it may employ this by combining

the prediction step with a corresponding measurement update. This may help to improve estimation quality (e.g., obtain a lower error variance). In such a setting, the triggers developed herein can be interpreted as 'conservative' triggers that take only prediction into account.

Remark 3.2. Under the assumption of perfect communication, the event of not receiving an update ($\gamma^i(k) = 0$) may also contain information useful for state estimation (also known as *negative information* [65]). Here, we disregard this information in the interest of a straightforward estimator implementation (see [49] for a more detailed discussion).

3.2.6 Problem formulation

The main objective of this chapter is the development of principled ways for predicting future triggering decisions. In particular, we shall develop two concepts:

1. *predictive triggering:* at every step k and for a fixed horizon $M > 0$, $\gamma^i(k+M)$ is predicted, i.e., whether or not communication is needed at M steps in future; and

2. *self triggering:* the next trigger is predicted at the time of the last trigger.

In the next sections, we develop these triggers for agent i shown in Figure 3.3, which is representative for any one agent in Figure 3.1. Because we will thus discuss estimation, triggering, and prediction solely for agent i (cf. Figure 3.3), we drop the index 'i' to simplify notation. Agent indices are re-introduced in Section 3.7, when multiple agents are again considered.

For ease of reference, key variables from this and later sections are summarized in Table 3.1.

3.3 Triggering Framework

To develop a framework for making predictive triggering decisions, we extend the approach from [49], where triggering is formulated as a one-step optimal decision problem trading off estimation and communication cost. While this framework was used in [49] to re-derive existing event triggers (summarized in Section 3.3.1), we extend the framework herein to yield predictive and self triggering (Section 3.3.2 and 3.3.3).

3.3.1 Decision framework for event triggering

The sensor agent (cf. Figure 3.3) makes a decision between using the communication channel (and thus paying a communication cost $C(k)$) to improve the remote estimate, or to save communication, but pay a price in terms of a deteriorated estimation performance (captured by a suitable estimation cost $E(k)$). The communication

Table 3.1: Summary of main variables used in this chapter. The agent index 'i' is dropped for all variables in Section 3.3 to 3.5.

$\mathbb{N}_N$	Set of integers $\{1, \dots, N\}$
A_i, B_i, C_i, Q_i, R_i	Dynamic system parameters
F_i	Control gain corresponding to agent i's state
$x^i(k)$	State of agent i, eq. (3.1)
$\hat{x}^i(k)$	Kalman filter (KF) estimate (3.6)
$\check{x}^i(k)$	Remote state estimate (3.14)
$\hat{e}^i(k)$	KF estimation error (3.9)
$e^i(k)$	Remote estimation error (3.15)
$\gamma^i(k)$	Communication decision (1=communicate, 0=not)
$\Gamma^i(k)$	Set of communication decisions $\{\gamma^i(1), \dots, \gamma^i(k)\}$
$X\|_{\gamma(k)=0}$, $X\|_{\gamma(k)=1}$	Expression X evaluated for resp. $\gamma(k) = 0$, $\gamma(k) = 1$
$\mathcal{Y}^i(k)$	Set of all measurements on agent i until time k
$\mathcal{U}^i(k)$	Set of all inputs on agent i until time k
$\tilde{x}(k)$, $\tilde{e}(k)$, etc.	Collection of corresponding variables for all agents
$C(k)$	Communication cost ('i' dropped)
$E(k)$	Estimation cost ('i' dropped)
M	Prediction horizon ('i' dropped)
$\ell(k)$	Last triggering time ('i' dropped)
$\kappa(k)$	Time of last nonzero elem. in $\Gamma(k+M)$ ('i' dropped)
Δ	Number of steps from $\kappa(k-1)$ to $k+M$ (cf. Lem. 3.2)

cost $C(k)$ is application specific and may be associated with the use of bandwidth or energy, for example. We assume $C(k)$ is known for all times k. The estimation cost $E(k)$ is used to measure the discrepancy between the remote estimation error $e(k)$ without update ($\gamma(k) = 0$), which we write as $e(k)|_{\gamma(k)=0}$, and with update, $e(k)|_{\gamma(k)=1}$. Here, we choose

$$E(k) = e(k)^{\mathrm{T}} e(k)|_{\gamma(k)=0} - e(k)^{\mathrm{T}} e(k)|_{\gamma(k)=1} \tag{3.16}$$

comparing the difference in quadratic errors.

Formally, the triggering decision can then be written as

$$\min_{\gamma(k)\in\{0,1\}} \gamma(k)C(k) + (1-\gamma(k))E(k). \tag{3.17}$$

Ideally, one would like to know $e(k)|_{\gamma(k)=0}$ and $e(k)|_{\gamma(k)=1}$ exactly when computing the estimation cost in order to determine whether it is worth paying the cost for communication. However, $e(k)$ cannot be computed since the true state is generally unknown (otherwise we would not have to bother with state estimation in the first

place). As is proposed in [49], we consider instead the expectation of $E(k)$ conditioned on the data $\mathcal{D}(k)$ that is available by the decision making agent. Formally,

$$\min_{\gamma(k)\in\{0,1\}} \gamma(k)C(k) + (1-\gamma(k))\ \mathbb{E}[E(k)|\mathcal{D}(k)] \tag{3.18}$$

which directly yields the triggering law

$$\text{at time } k\text{:}\quad \gamma(k)=1 \ \Leftrightarrow\ \mathbb{E}[E(k)|\mathcal{D}(k)] \geq C(k). \tag{3.19}$$

In [49], this framework was used to re-derive common event-triggering mechanisms such as innovation-based triggers [60, 67], or variance-based triggers [61], depending on whether the current measurement $y(k)$ is included in $\mathcal{D}(k)$, or not.

Remark 3.3. The choice of quadratic errors in (3.16) is only one possibility for measuring the discrepancy between $e(k)|_{\gamma(k)=0}$ and $e(k)|_{\gamma(k)=1}$ and quantifying estimation cost. It is motivated from the objective of keeping the squared estimation error small, a common objective in estimation. The estimation cost in (3.16) is positive if the squared error $e(k)^{\mathrm{T}}e(k)|_{\gamma(k)=0}$ (i.e., without communication) is larger than $e(k)^{\mathrm{T}}e(k)|_{\gamma(k)=1}$ (with communication), which is to be expected on average. Moreover, the quadratic error is convenient for the following mathematical analysis. Finally, the scalar version of (3.16) was shown in [49] to yield common known event triggers. However, other choices than (3.16) are clearly conceivable, and the subsequent framework can be applied analogously.

3.3.2 Predictive triggers

This framework can directly be extended to derive a predictive trigger as formulated in Section 3.2.6, which makes a communication decision M steps in advance, where $M>0$ is fixed by the designer. Hence, we consider the future decision on $\gamma(k+M)$ and condition the future estimation cost $E(k+M)$ on $\mathcal{D}(k)=\{\mathcal{Y}(k),\mathcal{U}(k)\}$, the data available at the current time k. Introducing $\bar{E}(k+M|k) := \mathbb{E}[E(k+M)|\mathcal{Y}(k),\mathcal{U}(k)]$, the optimization problem (3.17) then becomes

$$\min_{\gamma(k+M)\in\{0,1\}} \gamma(k+M)C(k+M) + (1-\gamma(k+M))\bar{E}(k+M|k) \tag{3.20}$$

which yields the *predictive trigger* (PT):

$$\text{at time } k\text{:}\quad \gamma(k+M)=1 \ \Leftrightarrow\ \bar{E}(k+M|k) \geq C(k+M). \tag{3.21}$$

In Section 3.4, we solve $\bar{E}(k+M|k) = \mathbb{E}[E(k+M)|\mathcal{Y}(k),\mathcal{U}(k)]$ for the choice of error measure (3.16) to obtain an expression for the trigger (3.21) in terms of the problem parameters.

3.3.3 Self triggers

A self trigger computes the next triggering instant at the time when an update is sent. A self triggering law is thus obtained by solving (3.21) at time $k = \ell(k)$ for the smallest M such that $\gamma(k+M) = 1$. Here, $\ell(k) \leq k$ denotes the last triggering time; in the following, we drop 'k' when clear from context and simply write $\ell(k) = \ell$. Formally, the *self trigger* (ST) is then given by:

$$\begin{aligned} \text{at time } k{=}\ell\text{: find smallest } M{\geq}1 \text{ s.t. } \bar{E}(\ell+M|\ell) \geq C(\ell+M), \\ \text{set } \gamma(\ell+1){=}\ldots{=}\gamma(\ell+M-1){=}0, \gamma(\ell+M){=}1. \end{aligned} \tag{3.22}$$

While both the PT and the ST compute the next trigger ahead of time, they represent two different triggering concepts. The PT (3.21) is evaluated at every time step k with a given prediction horizon M, whereas the ST (3.22) needs to be evaluated at $k = \ell$ only and yields (potentially varying) M. That is, M is a fixed design parameter for the PT, and computed with the ST. Which of the two should be used depends on the application (e.g., whether continuous monitoring of the error signal is desirable). The two types of triggers will be compared in simulations and experiments in subsequent sections.

3.4 Predictive Trigger and Self Trigger

Using the triggering framework of the previous section, we derive concrete instances of the self and predictive trigger for the squared estimation cost (3.16).

To this end, we first determine the PDF of the estimation errors.

3.4.1 Error distributions

In this section, we compute the conditional error PDF $f(e(k+M)|\mathcal{Y}(k),\mathcal{U}(k))$ for the cases $\gamma(k+M) = 0$ and $\gamma(k+M) = 1$, which characterize the distribution of the estimation cost $E(k+M)$ in (3.16). These results are used in the next section to solve for the triggers (3.21) and (3.22).

Both triggers (3.21) and (3.22) predict the communication decisions M steps ahead of the current time k. Hence, in both cases, the set of triggering decisions $\Gamma(k+M)$ can be computed from the data $\mathcal{Y}(k)$, $\mathcal{U}(k)$. In the following, it will be convenient to denote the time index of the last nonzero element in $\Gamma(k+M)$ (i.e., the last planned triggering instant) by $\kappa(k)$; for example, for $\Gamma(10) = \{\ldots, \gamma(8) = 1, \gamma(9) = 1, \gamma(10) = 0\}$, $k = 6$, and $M = 4$, we have $\kappa(6) = 9$. It follows that $\kappa(k) \geq \ell(k)$, with equality $\kappa(k) = \ell(k)$ if no trigger is planned for the next M steps.

The following two lemmas state the sought error PDFs.

Lemma 3.1. *For $\gamma(k+M)=1$, the predicted error $e(k+M)$ conditioned on $\mathcal{Y}(k)$, $\mathcal{U}(k)$ is normally distributed with*[1]

$$\begin{aligned} f(e(k+M)|\mathcal{Y}(k),\mathcal{U}(k)) &= \mathcal{N}(e(k+M);\, \hat{e}^{\mathrm{c}}(k+M|k), P^{\mathrm{c}}(k+M|k)) \\ &= \mathcal{N}(e(k+M);\, 0, P(k+M)). \end{aligned} \tag{3.23}$$

Proof. See Appendix B.1. □

Lemma 3.2. *For $\gamma(k+M)=0$, the predicted error $e(k+M)$ conditioned on $\mathcal{Y}(k)$, $\mathcal{U}(k)$ is normally distributed*[1]

$$f(e(k+M)|\mathcal{Y}(k),\mathcal{U}(k)) = \mathcal{N}(e(k+M);\, \hat{e}^{\mathrm{nc}}(k+M|k), P^{\mathrm{nc}}(k+M|k)) \tag{3.24}$$

with mean and variance given as follows.

Case (i): $k > \kappa(k-1)$ (i.e., no trigger planned in prediction horizon)

$$\hat{e}^{\mathrm{nc}}(k+M|k) = \bar{A}^M \left(\hat{x}(k) - \bar{A}^{k-\ell}\hat{x}(\ell) - \sum_{m=1}^{k-\ell} \bar{A}^{k-\ell-m} B \xi(\ell+m-1) \right) \tag{3.25}$$

$$P^{\mathrm{nc}}(k+M|k) = P(k+M|k) + \Xi(k,M) \tag{3.26}$$

where

$$\Xi(k,M) := \sum_{m=1}^{M-1} G(M-m-1)L(k+m)\tilde{P}(k+m)L(k+m)^T G(M-m-1)^T, \tag{3.27}$$

$$\tilde{P}(k) := CAP(k)A^TC^T + CQC^T + R, \tag{3.28}$$

$$G(m) := AG(m-1) + BF\bar{A}^m, \quad G(0) := BF, \tag{3.29}$$

$L(k)$ is the KF gain (3.5), and $P(k+M|k)$ is the KF prediction variance in (3.8).

Case (ii): $k \le \kappa(k-1)$ (i.e., trigger planned in horizon)

$$\hat{e}^{\mathrm{nc}}(k+M|k) = 0 \tag{3.30}$$

$$P^{\mathrm{nc}}(k+M|k) = P(\kappa+\Delta|\kappa) + \Xi(\kappa,\Delta) \tag{3.31}$$

where κ is used as shorthand for $\kappa(k-1)$, and $\Delta := k+M-\kappa(k-1)$.

Proof. See Appendix B.2. □

A simpler formula for Lemma 3.2 can be given for the case of an autonomous system (3.1) without input:

[1] The superscripts 'c' and 'nc' denote the cases 'communication' ($\gamma=1$) and 'no communication' ($\gamma=0$).

Corollary 3.1. *For* (3.1) *with* $B_i u^i(k-1) = 0$, (3.24) *holds for case (i) with*

$$\hat{e}^{\mathrm{nc}}(k+M|k) = A^M \left(\hat{x}(k) - A^{k-\ell}\hat{x}(\ell)\right) \tag{3.32}$$
$$P^{\mathrm{nc}}(k+M|k) = P(k+M|k) \tag{3.33}$$

and for case (ii) with

$$\hat{e}^{\mathrm{nc}}(k+M|k) = 0 \tag{3.34}$$
$$P^{\mathrm{nc}}(k+M|k) = P(\kappa+\Delta|\kappa). \tag{3.35}$$

Proof. Taking $B = 0$ yields $\bar{A} = A$ and $\Xi(k, M) = 0$ and thus the result. □

We thus conclude that the extra term $\Xi(k, M)$ in the variance (3.26) stems from additional uncertainty about not exactly knowing future inputs.

3.4.2 Self trigger

The ST law (3.22) is stated for a general estimation error $\bar{E}(\ell+M|\ell)$. With the preceding lemmas, we can now solve for $\bar{E}(\ell+M|\ell)$ and obtain the concrete self triggering rule for the quadratic error (3.16).

Proposition 3.1. For the quadratic error (3.16), the self trigger (ST) (3.22) becomes:

$$\begin{aligned}&\text{find smallest } M \geq 1 \text{ s.t.}\\ &\mathrm{trace}(P(\ell+M|\ell) + \Xi(\ell, M) - P(\ell+M)) \geq C(\ell+M);\\ &\text{set } \gamma(\ell+1) = \ldots = \gamma(\ell+M-1) = 0, \gamma(\ell+M) = 1.\end{aligned} \tag{3.36}$$

Proof. Applying Lemma 3.1 and Lemma 3.2 (for $k = \ell = \kappa(k-1)$), we obtain

$$\begin{aligned}\bar{E}(\ell+M|\ell) &= \mathbb{E}\big[\, e(\ell+M)^{\mathrm{T}} e(\ell+M)|_{\gamma(\ell+M)=0} \,\big|\, \mathcal{Y}(\ell), \mathcal{U}(\ell)\,\big]\\ &\quad - \mathbb{E}\big[\, e(\ell+M)^{\mathrm{T}} e(\ell+M)|_{\gamma(\ell+M)=1} \,\big|\, \mathcal{Y}(\ell), \mathcal{U}(\ell)\,\big]\\ &= \|\hat{e}^{\mathrm{nc}}(\ell+M|\ell)\|^2 - \|\hat{e}^{\mathrm{c}}(\ell+M|\ell)\|^2 + \mathrm{trace}(P^{\mathrm{nc}}(\ell+M|\ell) - P^{\mathrm{c}}(\ell+M|\ell))\\ &= \mathrm{trace}(P(\ell+M|\ell) + \Xi(\ell, M) - P(\ell+M))\end{aligned} \tag{3.37}$$

where $\mathbb{E}[e^{\mathrm{T}} e] = \|\mathbb{E}[e]\|^2 + \mathrm{trace}(\mathrm{Var}[e])$ with $\|\cdot\|$ the Euclidean norm was used. □

The self triggering rule is intuitive: a communication is triggered when the uncertainty of the open-loop estimator (prediction variance $P(\ell+M|\ell) + \Xi(\ell, M)$) exceeds the closed-loop uncertainty (KF variance $P(\ell+M)$) by more than the cost of communication. The estimation mean does not play a role here, since it is zero in both cases for $k = \kappa$.

3.4.3 Predictive trigger

Similarly, we can employ lemmas 3.1 and 3.2 to compute the predictive trigger (3.21).

Proposition 3.2. For the quadratic error (3.16), the predictive trigger (PT) (3.21) becomes, for $k > \kappa(k-1)$,

$$\gamma(k+M) = 1 \;\Leftrightarrow\; \left\|\bar{A}^M(\hat{x}(k) - \bar{A}\check{x}(k-1) - B\xi(k-1))\right\|^2 + \operatorname{trace}\big(P(k+M|k) + \Xi(k,M) - P(k+M)\big) \geq C(k+M) \tag{3.38}$$

and, for $k \leq \kappa(k-1)$,

$$\gamma(k+M) = 1 \;\Leftrightarrow\; \operatorname{trace}\big(P(\kappa+\Delta|\kappa) + \Xi(\kappa,\Delta) - P(\kappa+\Delta)\big) \geq C(\kappa+\Delta). \tag{3.39}$$

with Δ as defined in Lemma 3.2.

Proof. For $k > \kappa(k-1)$ (i.e., the last scheduled trigger occurred in the past), we obtain from lemmas 3.1 and 3.2

$$\bar{E}(k+M|k) = \left\|\bar{A}^M(\hat{x}(k) - A\check{x}(k-1) - B\xi(k-1))\right\|^2 + \operatorname{trace}\big(P(k+M|k) + \Xi(k,M) - P(k+M)\big), \tag{3.40}$$

where we used $\bar{A}^{k-\ell}\hat{x}(\ell) + \sum_{m=1}^{k-\ell} \bar{A}^{k-\ell-m}B\xi(\ell+m-1) = A\check{x}(k-1) + B\xi(k-1)$, which follows from the definition of the remote estimator (3.14) with $\gamma(k) = 0$ for $k > \ell$.

Similarly, for $k \leq \kappa(k-1)$, we obtain $\bar{E}(k+M|k) = \operatorname{trace}\big(P(\kappa+\Delta|\kappa) + \Xi(\ell,\Delta) - P(\kappa+\Delta)\big)$. □

Similar to the ST (3.36), the second term in the PT (3.38) relates the M-step open-loop prediction variance $P(k+M|k) + \Xi(k,M)$ to the closed-loop variance $P(k+M)$. However, now the reference time is the current time k, rather than the last transmission ℓ, because the PT exploits data until k. In contrast to the ST, the PT also includes a mean term (first term in (3.38)). When conditioning on new measurements $\mathcal{Y}(k)$ ($k > \ell$), the remote estimator (which uses only data until ℓ) is biased; that is, the mean (3.25) is non-zero. The bias term captures the difference in the mean estimates of the remote estimator ($A\check{x}(k-1) + B\xi(k-1)$) and the KF ($\hat{x}(k)$), both predicted forward by M steps. This bias contributes to the estimation cost (3.38).

The rule (3.39) corresponds to the case where a trigger is already scheduled to happen at time κ in future (within the horizon M). Hence, it is clear that the estimation error will be reset at κ, and from that point onward, variance predictions are used in analogy to the ST (3.36) (ℓ replaced with κ, and the horizon M with Δ). This trigger is independent of the data $\mathcal{Y}(k)$, $\mathcal{U}(k)$ because the error at the future reset time κ is fully determined by the distribution (3.23), independent of $\mathcal{Y}(k)$, $\mathcal{U}(k)$.

3.4.4 Discussion

To obtain insight into the derived PT and ST, we next analyze and compare their structure. To focus on the essential triggering behavior and simplify the discussion, we consider the case without inputs ($B_i u^i(k-1) = 0$ in (3.1)). We also compare to an *event trigger* (ET), which is obtained from the PT (3.38) by setting $M = 0$:

$$\gamma(k) = 1 \;\Leftrightarrow\; \bar{E}(k|k) = \left\|\hat{x}(k) - A\check{x}(k-1)\right\|^2 \geq C(k). \tag{3.41}$$

The trigger directly compares the two options at the remote estimator, $\hat{x}(k)$ and $A\check{x}(k-1)$. To implement the ET, communication must be available instantaneously if needed.

The derived rules for ST, PT, and ET have the same threshold structure

$$\gamma(k+M) = 1 \;\Leftrightarrow\; \bar{E}(k+M|k) \geq C(k+M) \tag{3.42}$$

where the communication cost $C(k+M)$ corresponds to the triggering threshold. The triggers differ in the expected estimation cost $\bar{E}(k+M|k)$. To shed light on this difference, we introduce

$$\bar{E}^{\mathrm{mean}}(k,M) := \left\|A^M(\hat{x}(k) - A\check{x}(k-1))\right\|^2 \tag{3.43}$$

$$\bar{E}^{\mathrm{var}}(k,M) := \mathrm{trace}(P(k+M|k) - P(k+M)). \tag{3.44}$$

With this, the triggers ST (3.36), PT (3.38), (3.39), and ET (3.41) are given by (3.42) with

$$\bar{E}(k+0|k) = \bar{E}^{\mathrm{mean}}(k,0), M = 0 \qquad \text{(ET)} \tag{3.45}$$

$$\bar{E}(k+M|k) = \bar{E}^{\mathrm{mean}}(k,M) + \bar{E}^{\mathrm{var}}(k,M) \qquad \text{(PT), } k > \kappa \tag{3.46}$$

$$\bar{E}(k+M|k) = \bar{E}^{\mathrm{var}}(\kappa,\Delta) \qquad \text{(PT), } k \leq \kappa \tag{3.47}$$

$$\bar{E}(\ell+M|\ell) = \bar{E}^{\mathrm{var}}(\ell,M) \qquad \text{(ST).} \tag{3.48}$$

Hence, the trigger signals are generally a combination of the 'mean' signal (3.43) and the 'variance' signal (3.44). Noting that the mean signal (3.43) depends on real-time measurement data $\mathcal{Y}(k)$ (through $\hat{x}(k)$), while the variance signal (3.44) does not, we can characterize ET and PT as *online triggers*, while ST is an *offline trigger*. This reflects the intended design of the different triggers. ST is designed to predict the next trigger at the time ℓ of the last triggering, without seeing any data beyond ℓ. This allows the sensor to go to sleep in-between triggers, for example. ET and PT, on the other hand, continuously monitor the sensor data to make more informed transmit decisions (as shall be seen in the following comparisons).

While ET requires instantaneous communication, which is limiting for online allocation of communication resources, PT makes the transmit decision $M \geq 1$ steps ahead of time. ET compares the mean estimates only (cf. (3.45)), while PT results in a combination of mean and variance signal (cf. (3.46)). If a transmission is

already scheduled for $\kappa(k-1) \geq k$, PT resorts to the ST mechanism for predicting beyond $\kappa(k-1)$; that is, it relies on the variance signal only (cf. (3.47)).

While ST can be understood as an *open-loop* trigger ((3.48) can be computed without any measurement data), ET clearly is a *closed-loop* trigger requiring real-time data $\mathcal{Y}(k)$ for the decision on $\gamma(k)$. PT can be regarded as an intermediate scheme exploiting real-time data and variance-based predictions. Accordingly, the novel predictive triggering concept lies between the known concepts of event and self triggering.

The ST is similar to the variance-based triggers proposed in [61]. Therein, it was shown for a slightly different scenario (transmission of measurements instead of estimates) that event triggering decisions based on the variance are independent of any measurement data and can hence be computed off-line. Similarly, when assuming that all problem parameters A, C, Q, R in (3.1), (3.2) are known a priori, (3.36) can be pre-computed for all times. However, if some parameters only become available during operation (e.g., the sensor accuracy $R(k)$), the ST also becomes an online trigger.

For the case with inputs ($B_i u^i(k-1) \neq 0$ in (3.1)), the triggering behavior is qualitatively similar. The mean signal (3.43) will include the closed-loop dynamics $\bar{A}$ and the input $\xi(k-1)$ corresponding to other agents, and the variance signal (3.44) will include the additional term $\Xi(k, M)$ accounting for the additional uncertainty of not knowing the true input.

3.5 Illustrative Example

To illustrate the behavior of the obtained PT and ST, we present a numerical example. We study simulations of the stable, scalar, LTI system (3.1), (3.2) with:

Example 3.1. $A = 0.98$, $B = 0$ (no inputs), $C = 1$, $Q = 0.1$, $R = 0.1$, and $\bar{x}(0) = X_0 = 1$.

3.5.1 Self trigger

We first consider the self trigger (ST). Results of the numerical simulation of the event-triggered estimation system (cf. Figure 3.3) consisting of the local state estimator (3.3)–(3.7), the remote state estimator (3.14), and the ST (3.36) with constant cost $C(k) = C = 0.6$ are shown in Figure 3.4. The estimation errors of the local and remote estimator are compared in the first graph. As expected, the remote estimation error $e(k) = x(k) - \hat{x}(k)$ (orange) is larger than the local estimation error $\hat{e}(k) = x(k) - \hat{x}(k)$ (blue). Yet, the remote estimator only needs 14 % of the samples.

The triggering behavior is illustrated in the second graph showing the triggering signals $\bar{E}^{\text{mean}}$ (3.43), $\bar{E}^{\text{var}}$ (3.44), and $\bar{E} = \bar{E}^{\text{mean}} + \bar{E}^{\text{var}}$, and the bottom graph depicting the triggering decision γ. Obviously, the ST entirely depends on the variance signal $\bar{E}^{\text{var}}$ (orange, identical with $\bar{E}$ in black), while $\bar{E}^{\text{mean}} = 0$ (blue).

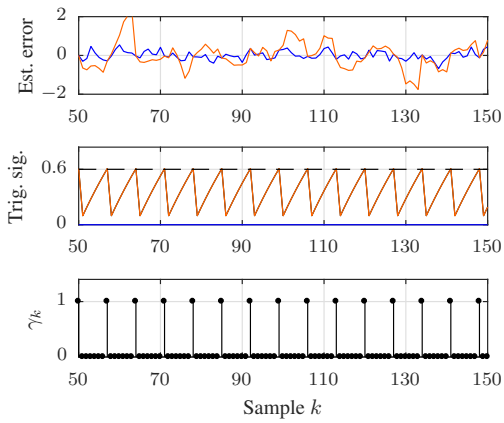


Figure 3.4: Example 3.1 with self trigger (ST). TOP: KF estimation error $\hat{e} = x - \hat{x}$ (**blue**) and remote error $e = x - \hat{x}$ (**orange**). MIDDLE: components of the triggering signal $\bar{E}^{\text{mean}}$ (3.43) (**blue**), $\bar{E}^{\text{var}}$ (3.44) (**orange**), the triggering signal $\bar{E} = \bar{E}^{\text{mean}} + \bar{E}^{\text{var}}$ (**black**, hidden), and the threshold $C(k) = 0.6$ (**dashed**). BOTTOM: triggering decisions γ.

This reflects the previous discussion about the ST being independent of online measurement data. The triggering behavior (the signal $\bar{E}$ and the decisions γ) is actually *periodic*, which can be deduced as follows: the variance $P(k)$ of the KF (3.3)–(3.7) converges exponentially to a steady-state solution $\bar{P}$, [136]; hence, the triggering law (3.36) asymptotically becomes $\text{trace}(V_{\text{o}}^{M}(\bar{P}) - \bar{P}) \geq C$ with $V_{\text{o}}(X) := AXA^{\text{T}} + Q$, and (3.22) thus has a unique solution M corresponding to the period seen in Figure 3.4.

Periodic transmit sequences are typical for variance-based triggering on time-invariant problems, which has also been found and formally proven for related scenarios in [61, 68].

3.5.2 Predictive trigger

The results of simulating Example 3.1, now with the PT (3.38), (3.39), and prediction horizon $M = 2$, are presented in Figure 3.5 for the cost $C(k) = C = 0.6$, and in Figure 3.6 for $C(k) = C = 0.25$. Albeit using the same trigger, the two simulations show fundamentally different triggering behavior: while the triggering signal $\bar{E}$ and the decisions γ in Figure 3.5 are irregular, they are periodic in Figure 3.6.

Apparently, the choice of the cost $C(k)$ determines the different behavior of the PT. For $C(k) = 0.6$, the triggering decision depends on both, the mean signal $\bar{E}^{\text{mean}}$ and the variance signal $\bar{E}^{\text{var}}$, as can be seen from Figure 3.5 (middle graph). Because $\bar{E}^{\text{mean}}$ is based on real-time measurements, which are themselves random variables (3.2), the triggering decision is a random variable. We also observe in Figure 3.5 that the variance signal $\bar{E}^{\text{var}}$ is alone not sufficient to trigger a communication.

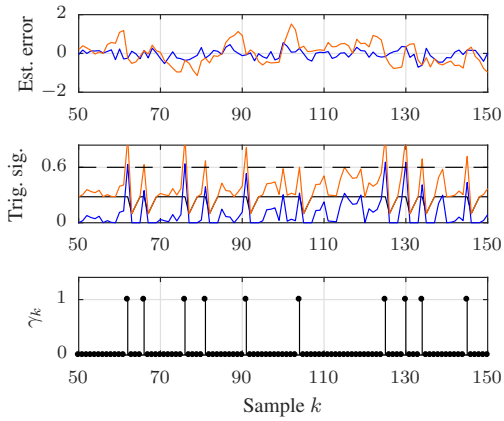


Figure 3.5: Example 3.1 with predictive trigger (PT) and $C(k) = 0.6$. Coloring of the signals is the same as in Figure 3.4. The triggering behavior is *stochastic*.

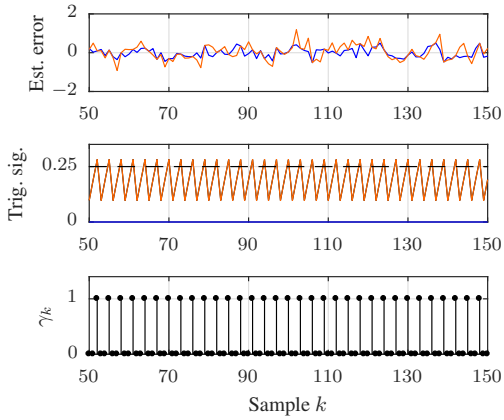


Figure 3.6: Example 3.1 with predictive trigger (PT) and $C(k) = 0.25$. Coloring of the signals is the same as in Figure 3.4. The triggering behavior is *periodic*.

However, when lowering the cost of communication $C(k)$ enough, the variance signal alone becomes sufficient to cause triggers. Essentially, triggering then happens according to (3.39) only, and (3.38) becomes irrelevant. Hence, the PT resorts to self triggering behavior for small enough communication cost $C(k)$. That is, the PT undergoes a phase transition for some value of $C(k)$ from stochastic/online triggering to deterministic/offline triggering behavior.

3.5.3 Estimation versus communication trade-off

Following the approach from [49], we evaluate the effectiveness of different triggers by comparing their trade-off curves of average estimation error $\mathcal{E}$ versus average communication $\mathcal{C}$ obtained from Monte Carlo simulations. In addition to the ST

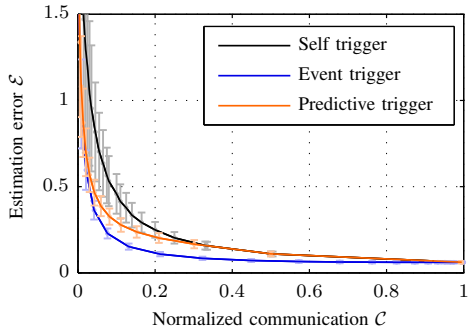


Figure 3.7: Trade-off between estimation error $\mathcal{E}$ and average communication $\mathcal{C}$ for different triggering concepts applied to Example 3.1. Each point represents the average from 50'000 Monte Carlo simulations, and the light error bars correspond to one standard deviation.

(3.36) and the PT (3.38), (3.39), $M = 2$, we also compare against the ET (3.41). The latter is expected to yield the best trade-off because it makes the triggering decision at the latest possible time (ET decides at time k about communication at time k).

The estimation error $\mathcal{E}$ is measured as the squared error $e(k)^2$ averaged over the simulation horizon (200 samples) and 50 000 simulation runs. The average communication $\mathcal{C}$ is normalized such that $\mathcal{C} = 1$ means $\gamma(k) = 1$ for all k, and $\mathcal{C} = 0$ means no communication (except for one enforced trigger at $k = 1$). By varying the constant communication cost $C(k) = C$ in a suitable range, an $\mathcal{E}$-vs-$\mathcal{C}$ curve is obtained, which represents the estimation/communication trade-off for a particular trigger. The results for Example 3.1 are shown in Figure 3.7.

Comparing the three different triggering schemes, we see that the ET is superior, as expected, because its curve is uniformly below the others Also expected, the ST is the least effective since no real-time information is available and triggers are purely based on variance predictions. The novel concept of predictive triggering can be understood as an intermediate solution between these two extremes. For small communication cost $C(k)$ (and thus relatively large communication $\mathcal{C}$), the PT behaves like the ST, as was discussed in the previous section and is confirmed in Figure 3.7 (orange and black curves essentially identical for large $\mathcal{C}$). When the triggering threshold $C(k)$ is relaxed (i.e., the cost increased), the PT also exploits real-time data for the triggering decision (through (3.43)), similar to the ET. Yet, the PT must predict the decision M steps in advance making its $\mathcal{E}$-vs-$\mathcal{C}$ trade-off generally less effective than the ET. In Figure 3.7, the curve for PT is thus between ET and ST and approaches either one of them for small and large communication $\mathcal{C}$.

3.6 Hardware Experiments: Remote Estimation & Feedback Control

Experimental results of applying the proposed PT and ST on an inverted pendulum platform are presented in this section. We show that trade-off curves in practice are similar to those in simulation (cf. Figure 3.7), and that the triggers are suitable for feedback control (i.e., stabilizing the pendulum).

3.6.1 Experimental setup

As an experimental platform we use the cyber-physical testbed already introduced in Section 2.5.1. Instead of a multi-hop network, the pendulum here is directly connected to a standard laptop running Matlab/Simulink. Thus, characteristics of the communication network to be investigated are implemented in the Simulink model. The round time of the network is assumed to be 10 ms. For the PT the prediction horizon is $M = 2$. Thus, the communication network has 20 ms to reconfigure, which is expected to be sufficient for fast protocols such as the one presented in the previous chapter.

For stabilization we use an LQR with $Q = 30I$ and $R = I$, with I the identity matrix, which leads to stable balancing with slight motion of the cart.

3.6.2 Remote estimation

The first set of experiments investigates the remote estimation scenario as in Figure 3.3. For this purpose, the pendulum is stabilized locally via the above LQR, which runs at 1 ms and directly acts on the encoder measurements and their derivatives obtained from finite differences. The closed-loop system thus serves as the dynamic process in Figure 3.3 (described by equation (3.1)), whose state is to be estimated and communicated via ET, PT, and ST to a remote location, which could represent another agent from Figure 3.1.

The local *State Estimator* in Figure 3.3 is implemented as the KF (3.3)–(3.7) with properly tuned matrices updated every 1 ms (at every sensor update). Triggering decisions are made at the round time of the network (10 ms). Accordingly, state predictions (3.14) are made every 10 ms (in *Prediction System i* in Figure 3.3).

Analogously to the numerical examples in Section 3.5, we investigate the estimation-versus-communication trade-off achieved by ET, PT, and ST. As can be seen in Figure 3.8, all three triggers lead to approximately the same curves. These results are qualitatively different from those of the numerical example in Figure 3.7, which showed notable differences between the triggers. Presumably, the reason for this lies in the low-noise environment of this experiment. The main source of disturbances is the encoder quantization, which is negligible. Therefore, the system is almost deterministic, and predictions are very accurate. Hence, in this setting, predicting future communication needs (PT, ST) does not involve any significant disadvantage compared to instantaneous decisions (ET).

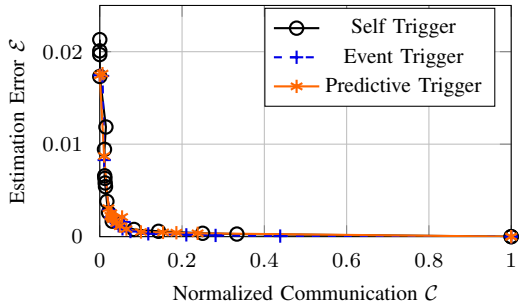


Figure 3.8: Trade-off between averaged communication and the estimation error for a pendulum experiment with low sensor noise. Each marker represents the mean of 10 experiments with the same communication cost. The variance is negligible and thus omitted.

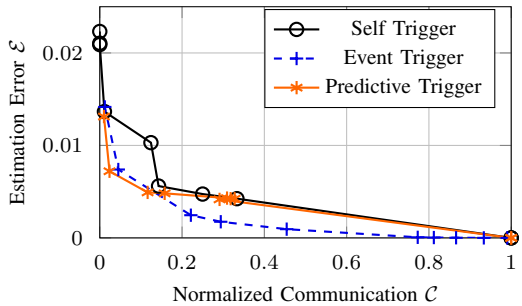


Figure 3.9: Same experiment as in Figure 3.8, but with noisy sensors.

To confirm these results, we added zero-mean Gaussian noise with variance 5×10^{-6} to the position and angle measurements. This emulates analog angle sensors instead of digital encoders, and is representative for many sensors in practice that involve stochastic noise. The results of this experiment are shown in Figure 3.9, which depict the same qualitative difference between the triggers as was observed in the numerical example in Figure 3.7.

3.6.3 Feedback control

The estimation errors obtained in Figure 3.8 are fairly small even with low communication. Thus, we expect the estimates obtained with PT and ST also to be suitable for feedback control, which we investigate here. In contrast to the setting in Section 3.6.2, the LQR controller does not use the local state measurements at the fast update interval of 1 ms, but the state predictions (3.14) instead. This corresponds to the controller being implemented on a remote agent, which is relevant for control of multiple CPSs as in Figure 3.1, where feedback loops are closed over a resource-limited network.

Figures 3.10 and 3.11 show the experimental results of using PT and ST for

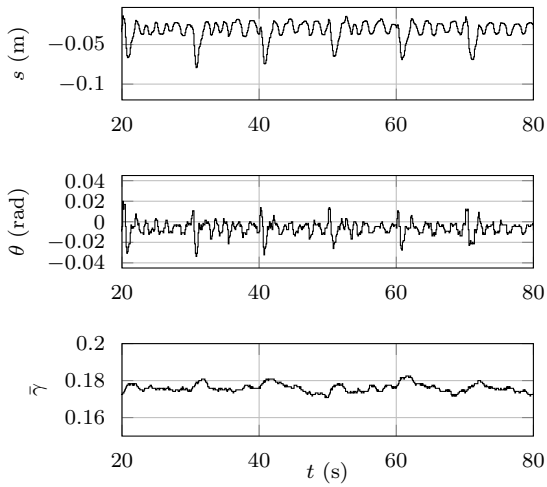


Figure 3.10: Closing the feedback loop with the PT. The graphs show, from top to bottom, the cart position s, the pendulum angle θ, and the obtained average communication $\bar{\gamma}$, computed as a moving average over 1200 samples. The communication cost was set to $C(k) = C = 0.009$.

feedback control. For these experiments, the weights of the LQR approach were chosen as those suggested by the manufacturer in [138], which leads to a slightly more robust controller. Both triggers are able to stabilize the pendulum well and save around 80 % of communication.

In addition to disturbances inherent in the system, the experiments also include impulsive disturbances on the input (impulse of 2 V amplitude and 500 ms duration every 10 s), which we added to study the triggers' behavior under deterministic disturbances. In addition to stochastic noise, such disturbances are relevant in many practical scenarios (e.g., a car braking, a wind gust on a drone). Under these disturbances, a particular advantage of PT over ST becomes apparent. The ST is an offline trigger which yields periodic communication (in this setting) that does not react to the external disturbances. The PT, on the other hand, takes the current error into account and is thus able to issue additional communication in case of disturbances. As a result, the maximum angle of the pendulum stays around 0.03 rad in magnitude for the PT, while it comes close to 0.04 rad for the ST.

3.7 Control with Multiple Agents

In the preceding sections, we addressed the problem posed in Section 3.2.6 for the case of two agents. In this section, we make precise how these results can be used for the scenario with multiple agents in Figure 3.1. Moreover, we sketch how the resulting closed-loop dynamics can be analyzed when remote estimates are used for feedback control.

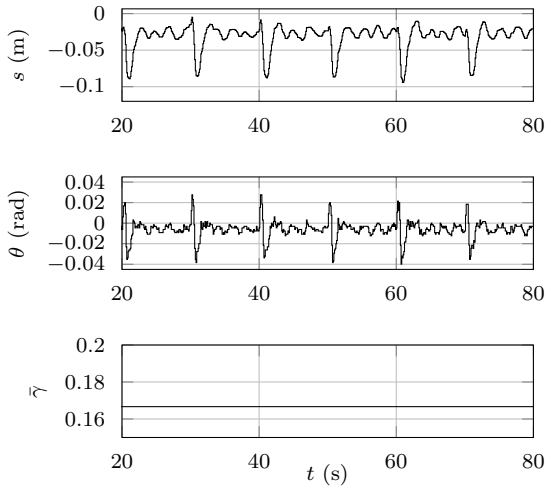


Figure 3.11: Closing the feedback loop with the ST. Same plots as in Figure 3.10.

Because we discuss multiple agents, we reintroduce the index 'i' to refer to an individual agent i from here onward.

3.7.1 Multiple agents

The derivations for a pair of agents as in Figure 3.2 in the previous sections equally apply to the scenario in Figure 3.1. Each agent implements the blocks from Figure 3.2: *State Estimation* is given by the KF (3.3)–(3.7), *Prediction* by (3.14), *Control* by (3.10), and the *Event Trigger* is replaced by either the ST (3.36) or the PT (3.38), (3.39). In particular, each agent makes predictions for those other agents whose state it requires for coordination. Whenever one agent transmits its local state estimate, it is broadcast over the network and received by all agents that care about this information, e.g., via many-to-all communication. In the considered scenario, the dynamics of the systems are decoupled according to (3.1), (3.2) (cf. Figure 3.1), but their action is coupled through the cooperative control (3.10).

In Section 3.8, a simulation study of a control problem with multiple agents is discussed.

3.7.2 Closed-loop analysis

While the main object of study in this chapter are predictive and self triggering for state estimation (cf. Figure 3.3), an important use of the algorithms is for feedback control as in Figures fig:iotControlSchematic and 3.2. The general suitability of the algorithms for feedback control has already been demonstrated in Section 3.6.3. As for feedback control, analyzing the closed-loop dynamics (e.g., for stability) is often of importance, we briefly outline here how this can be approached.

The closed-loop state dynamics of agent i are obtained from (3.1) and (3.10), and can be rewritten as

$$\begin{aligned} x^i(k) &= A_i x^i(k-1) + B_i F_i \hat{x}^i(k-1) + \sum_{j \in \mathbb{N}_N \setminus \{i\}} B_i F_j \check{x}^j(k-1) + v^i(k-1) \\ &= A_i x^i(k-1) + B_i F_i x^i(k-1) + \sum_{j \in \mathbb{N}_N \setminus \{i\}} B_i F_j x^j(k-1) \\ &\quad - B_i F_i \hat{e}^i(k-1) - \sum_{j \in \mathbb{N}_N \setminus \{i\}} B_i F_j e^j(k-1) + v^i(k-1) \end{aligned} \tag{3.49}$$

where $\hat{e}^i(k)$ is the KF estimation error (3.9) and $e^j(k)$ the remote estimation error (3.15). The combined closed-loop dynamics of N systems with concatenated state $\tilde{x}^{\mathrm{T}}(k) = [(x^1(k))^{\mathrm{T}}, (x^2(k))^{\mathrm{T}}, \ldots, (x^N(k))^{\mathrm{T}}]$ can then be written as

$$\tilde{x}(k) = (\tilde{A} + \tilde{B}\tilde{F})\tilde{x}(k-1) - \tilde{D}\tilde{\hat{e}}(k-1) - (\tilde{B}\tilde{F} - \tilde{D})\tilde{e}(k-1) + \tilde{v}(k-1) \tag{3.50}$$

where

$$\begin{aligned} \tilde{A} &:= \operatorname{diag}(A_1, \ldots, A_N), & \tilde{B}^{\mathrm{T}} &:= \begin{bmatrix} B_1^{\mathrm{T}} & \ldots & B_N^{\mathrm{T}} \end{bmatrix}, \\ \tilde{D} &:= \operatorname{diag}(B_1 F_1, \ldots, B_N F_N), & \tilde{F} &:= \begin{bmatrix} F_1 & \ldots & F_N \end{bmatrix}, \end{aligned}$$

diag denotes block-diagonal matrix, and $\tilde{\hat{e}}(k)$, $\tilde{e}(k)$, and $\tilde{v}(k)$ are the combined vectors of all $\hat{e}^i(k)$, $e^i(k)$, and $v^i(k)$ ($i \in \mathbb{N}_N$), respectively. The 'tilde' notation indicates variables that refer to the ensemble of all agents.

Equation (3.50) describes the closed-loop dynamics of N systems of Figure 3.1 that implement the control architecture in Figure 3.2; it can therefore be used to deduce closed-loop system properties. The evolution of the complete state $x(k)$ is governed by the transition matrix $\tilde{A} + \tilde{B}\tilde{F}$ and driven by three input terms: the KF error $\tilde{\hat{e}}(k-1)$, the remote error $\tilde{e}(k-1)$, and process noise $\tilde{v}(k-1)$. Under mild assumptions, the feedback matrix $\tilde{F}$ can be designed such that a stable transition matrix $\tilde{A} + \tilde{B}\tilde{F}$ results (i.e., all eigenvalues with magnitude less than 1), which implies that $\tilde{x}(k) = (\tilde{A} + \tilde{B}\tilde{F})\tilde{x}(k-1)$ is exponentially stable. Stability analysis then amounts to showing that the input terms are well behaved and bounded in a stochastic sense (e.g., bounded moments).[2] While $\tilde{v}(k-1)$ is Gaussian by assumption (cf. Section 3.2.1), $\tilde{\hat{e}}(k-1)$ being Gaussian follows from standard KF analysis [136] (cf. Section 3.2.2). Lemmas 3.2 and 3.1 can be instrumental to analyze the distribution of $\tilde{e}(k-1)$. However, the distribution of $\tilde{e}(k-1)$ depends on the chosen trigger, and its properties (e.g., bounded second moment) would have to be formally shown, which is beyond the goals of this chapter.

[2]For example, if, in $\tilde{x}(k) = (\tilde{A} + \tilde{B}\tilde{F})\tilde{x}(k-1) + \tilde{z}(k-1)$, the input $\tilde{z}(k)$ is uncorrelated and Gaussian with bounded variance, then stability of $\tilde{A} + \tilde{B}\tilde{F}$ implies bounded state variance (see, e.g., [136, Sec. 4.3]).

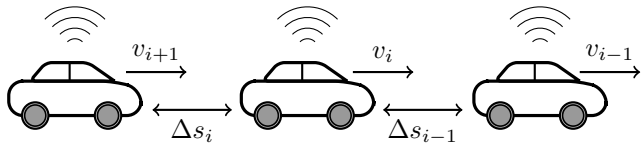


Figure 3.12: Schematic of vehicle platooning.

3.8 Simulation Study: Vehicle Platooning

To illustrate the scalability of the proposed triggers, we present a simulation study of vehicle platooning. Autonomous driving and in particular platooning of vehicles has already been motivated in Section 1.1. Platooning of autonomous vehicles has been extensively studied in literature, e.g., for heavy-duty freight transport [1, 139]. It has been shown, that platooning leads to remarkable improvements in terms of fuel consumption.

3.8.1 Model

We consider a chain of N vehicles (see Figure 3.12), which are modeled as unit point masses (cf. [64, 140]). The state of each vehicle is its absolute position s_i and velocity v_i, and its acceleration u_i is the control input. The control objectives are to maintain a desired distance between the vehicles and track a desired velocity for the platoon. For this study, we assume that every vehicle measures its absolute position.

The architecture of the vehicle platoon is as in Figure 3.1. To control the inter-vehicle distances, communication between the vehicles is required. We thus implement the control architecture given by Figure 3.2 with PT and ST to save communication. We assume 100 ms as the sample time for the inter-vehicle communication. Here, we consider the case where each vehicle transmits its local state information to all other vehicles. Alternative architecture, where communication is only possible with a subset of vehicles, are also conceivable in the considered scenario (see [139]), and the PT and ST can be used for only the required communication links appropriately.

For our chosen setup, where each vehicle is only able to measure its own absolute position, it is obvious that communication between vehicles is necessary to control the inter-vehicle distance. However, even if local sensor measurements are available, e.g., if every vehicle can measure the distance to the preceding vehicle via a radar sensor, communication is required to guarantee *string stability*. String stability indicates, whether oscillations are amplified upstream the traffic flow. In [141], it has been proven that if only local sensor measurements are used, string stability can only be guaranteed for velocity dependent spacing policies, i.e., the faster the cars drive the larger distances are required, and thus, the less fuel can be saved. Therefore, even in the presence of local measurements, communication between vehicles is crucial for fuel saving. In such a case, where additional local sensor measurements are available, predictive and self triggering can similarly be used, as also stated in

Remark 3.1.

To address the control objectives, we design an LQR for the linear state-space model that includes the vehicle velocities and their relative distances, i.e., $x_i(t) = [v_i(t),\ s_i(t) - s_{i-1}(t)]^{\mathrm{T}}$. The complete state $\tilde{x}$ is given by $x_1, x_2, \ldots, x_N$ except for no relative position for the last vehicle $i = N$ (cf. Figure 3.12). For this system, an LQR is designed with $Q = I$ and $R = 1000I$. The even-numbered diagonal entries of the Q matrix specify the inter-vehicle distance tracking, while the odd ones weight the desired velocity. To achieve tracking of desired velocity and inter-vehicle distance, the desired state $\tilde{x}_{\mathrm{des}}$ is introduced, and the LQR law $\tilde{u}(k) = \tilde{F}(\tilde{x}(k) - \tilde{x}_{\mathrm{des}}(k))$ implemented.

We emphasize that the feedback gain matrix $\tilde{F}$ is dense; that is, information about all states in the platoon are used to compute the optimal control input. Such controller can only be implemented in a distributed way, if complete state information is available on each agent via the architecture presented in Section 3.2.4 with all-to-all communication.

In the simulations below, position measurements are corrupted by independent noise, uniformly distributed in $[-0.1\,\mathrm{m}, 0.1\,\mathrm{m}]$. Likewise, the inputs are corrupted by uniform noise in $[-0.1\,\mathrm{m\,s^{-2}}, 0.1\,\mathrm{m\,s^{-2}}]$. Additionally, we assume 10 % Bernoulli message losses.

3.8.2 Platooning on changing surfaces

We investigate the performance versus communication trade-off achieved with PT and ST for platooning of 10 vehicles. Here, we are interested in the closed-loop performance that is achieved with the proposed architecture; hence, instead of the estimation error we use the sum of the element-wise absolute value of $\tilde{x} - \tilde{x}_{\mathrm{des}}$, averaged over the time steps and scaled by the state dimension, as performance metric $\tilde{J}$[3]. The platoon drives for 25 s, while keeping desired inter-vehicle distances of 10 m and velocity of $22.2\,\mathrm{m\,s^{-1}}$. After 200 m, the dynamics change due to different road conditions (e.g., continue driving on a wet road after leaving a tunnel), which is modeled by altering the vehicle dynamics accordingly (vehicles moving 50 % faster, and the effect of braking/accelerating is reduced by 50 %). Figure 3.13 shows the results from 100 Monte Carlo simulations.

Both triggers achieve significant communication savings at only a mild decrease of control performance. Similar to studies in previous sections, the PT performs better than the ST for low communication rates, because it can react to changing conditions. For high communication rates, PT and ST are identical. If the prediction horizon is extended, the performance of the PT gets closer to that of the ST, as can be obtained from the blue curve in Figure 3.13.

[3] Other performance metrics, such as the LQR cost, lead to similar insights, but have higher variance.

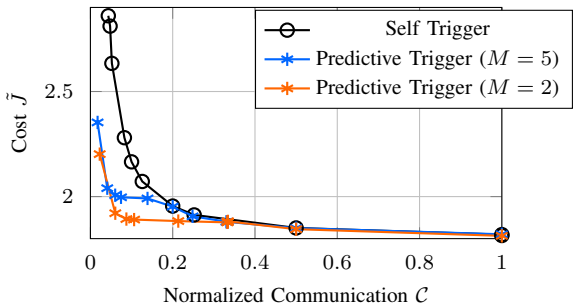


Figure 3.13: Trade-off between normalized communication and control cost for a 10 vehicles platoon. Every marker represents the mean of 100 Monte Carlo simulations. The variance is negligible and hence omitted. The plot shows the ST (black) as well as two curves for the PT, one with a prediction horizon of 2 (orange) and one with a prediction horizon of 5 (blue).

3.8.3 Braking

If vehicles drive in close proximity, the ability to react to sudden changes such as a braking maneuver of the preceding car, is critical. This is investigated here for three vehicles (simulation with more vehicles leads to the same insight).

Figure 3.14 shows simulation results, where all cars start with a velocity of $22.2\,\mathrm{m\,s^{-1}}$, but after 10 s, the first car brakes. The results in Figure 3.14 (left) show that even with very little communication, the PT is able to deal with this situation. The PT detects the need for more communication and is able to control inter-vehicle distances within safety bounds. As previously pointed out, the ST (Figure 3.14 right) cannot react online, which causes a crash in this example ($\Delta s_1 = 0$).

3.9 Conclusion

In control of multiple CPSs, the network is shared by many entities, thus, communication is a limited resource that must be taken into account when making control decisions for optimal system-level operation. This chapter sets a foundation for such resource-aware control. Distributed event-triggered state estimation provides a powerful architecture for sharing information between multiple systems and their cooperative control. The developed self trigger and predictive trigger allow one to anticipate future communication needs, which is fundamental for efficiently (re-)allocating network resources.

In order to leverage the potential of this chapter and realize actual resource savings on concrete systems, the integration of ST and PT herein with a suitable communication system is essential. While DETSE has successfully been implemented on wired CAN bus networks in prior works [60, 63], we target the integration with the protocol developed in Chapter 2, which provides many-to-all communication. Hence, it is ideally suited for scenarios such as in Figures 3.1 and 3.2, where multiple

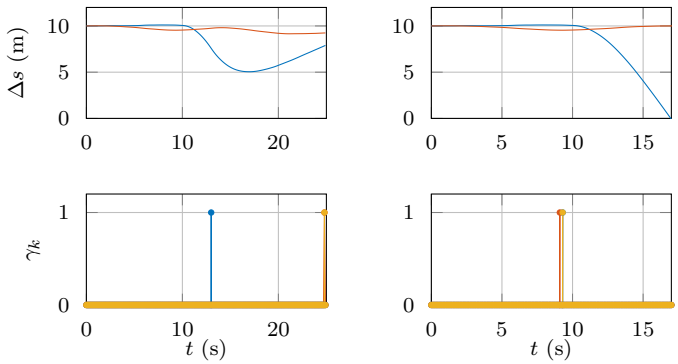


Figure 3.14: 3 vehicles platooning with a constant velocity of $22.2\,\mathrm{m\,s^{-1}}$. After 10 s the first car starts braking. The top plot shows the distances Δs_1 (blue) and Δs_2 (red); the bottom plot shows the communication instants (vehicle 1 in blue, vehicle 2 in red, and vehicle 3 in yellow). The left plots show the behavior for the PT (with communication cost $C(k) = C = 10$), the right plots for the ST (with communication cost $C(k) = C = 0.7$).

systems require information of each other for coordination. In particular, many-to-all communication allows for the effective realization of the predictors (3.14) on any agent that needs the corresponding state information. The concrete development and integration of such schemes is subject of ongoing research. While the focus of this chapter is on saving communication bandwidth, the proposed triggers can also be instrumental for saving other resources (e.g., computation or energy).

The predictive and self triggers are suitable for different application scenarios. The simulation and experimental studies herein clearly highlight the advantage of the predictive trigger: by continuously monitoring the triggering condition, it can react to unforeseeable events such as disturbances. The self trigger, on the other hand, is an offline trigger, which allows for setting devices to sleep. In contrast to commonly used event triggers, both proposed triggers can *predict* resource needs rather than making instantaneous decisions. Predictive triggering is a novel concept in-between the previously proposed concepts of self triggering and event triggering.

Concrete instances of the predictive and self trigger were derived herein for estimation of linear Gaussian systems. While the general idea of predicting triggers also extends to nonlinear estimation, properly formalizing this and deriving triggering laws for nonlinear problems is an interesting task for future work. Likewise, considering alternative optimization problems for different error choices in (3.16), as well as dynamic programming formulations in place of the one-step optimization in (3.17), may lead to interesting insights and alternative triggers. While the predictive and self triggers herein were shown to stabilize the inverted pendulum in the reported experiments, formally analyzing stability of the closed-loop system (e.g., along the lines outlined in Section 3.7.2) is another relevant open research question.

Chapter 4

Event-Triggered Pulse Control with Adaptation through Learning

The approaches presented so far rely on the availability of an accurate dynamics model. From here on, this assumption will be dropped. Instead of assuming a perfect model, we will evaluate the performance of the currently used model through a statistical analysis and trigger learning in case we detect an inaccurate model. We will, in addition, show that the learning approach is able to deal with load disturbances and introduce a novel event-triggered pulse control strategy that respects input saturations.

4.1 Introduction

In this chapter, we will present a new architecture for event-triggered pulse control that quantifies model accuracy and, if beneficial, automatically identifies system dynamics through learning. A block diagram of the approach is provided in Figure 4.1. We consider a plant with sensors and actuators, subject to process noise and disturbances (v and ϵ), and input saturations $u_{\max}$. Controller and actuator are connected over a communication network. Since communication is a scarce resource, periodic communication is not desirable and, therefore, we employ an event-triggered design (block 'State Trigger'). In case of an event, we apply a pulse with length t_{input} to reset the system to its equilibrium state. The pulse length naturally depends on the system dynamics. To obtain an accurate model of the system dynamics, we leverage system identification techniques to learn the model from data. As learning may be expensive, e.g., due to the involved computations, we only learn a new model if necessary, for instance, in case of a poor initial model or if the dynamics have changed. This decision process is quantified within a 'Learning Trigger'. Based on a statistical analysis of the time between events, the learning trigger decides whether the model of the system dynamics is accurate enough. If not, learning of a new model is triggered.

Contributions: The contributions of this chapter can be summarized as follows:

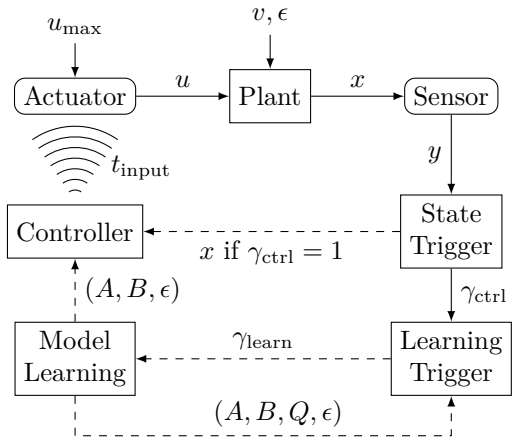


Figure 4.1: Block diagram of the proposed control design. Dashed lines represent connections that are only active in case of an event.

- A new architecture for event-triggered pulse control with learning;
- Development of a learning trigger for ETC, which allows to automatically identify system dynamics on necessity;
- Handling load disturbances by learning and compensating for them, thus replacing the integrator typically used in periodic control in a way suitable for wireless CPSs.

Outline: In the following section, we will formulate the problem setting. After that, we will detail our approach for event-triggered pulse control and the implementation of the learning trigger in Section 4.3. In Section 4.4, we will present a numerical study and conclude with a discussion in Section 4.5.

4.2 Problem Formulation

To study load disturbances, we enhance the LTI description (1.2) by a load disturbance $\epsilon \in \mathbb{R}^n$, i.e., we arrive at a system of the form

$$\mathrm{d}x(t) = Ax(t)\,\mathrm{d}t + Bu(t)\,\mathrm{d}t + \epsilon\,\mathrm{d}t + Q\,\mathrm{d}W(t). \tag{4.1}$$

We assume that we can measure the full state, thus, $y = x$ in Figure 4.1.

As depicted in Figure 4.1, control commands have to be communicated over a communication network. We thus employ an event-triggered design, with the block 'State Trigger' implemented by

$$\gamma_{\text{ctrl}} = 1 \iff \|x\|_2 \geq \delta, \tag{4.2}$$

where δ is a user-defined threshold and is essentially the deviation from the equilibrium that we are willing to tolerate. In case of an event, we apply a pulse to reset the system to its equilibrium, i.e.,

$$u(t) = \begin{cases} 0 & \text{if } \gamma_{\text{ctrl}} = 0 \\ \phi_{u_{\max}}\left(\hat{A}, \hat{B}, \hat{\epsilon}, x_{t_k}\right) & \text{if } \gamma_{\text{ctrl}} = 1, \end{cases} \tag{4.3}$$

where $\phi_{u_{\max}}$ is the pulse generating policy that will be made precise in Section 4.3.3, $(\hat{A}, \hat{B}, \hat{\epsilon})$ defines the model of the system dynamics we use to generate the pulse, and x_{t_k} the state of the system at the triggering instant. By applying a pulse with appropriate length, we can reset the system to its equilibrium state. This, however, requires that we have a model that accurately describes the true system dynamics. We obtain this model, and adapt it in case the dynamics change, via model-learning techniques. Model-learning may be expensive due to the involved computations or required exploration, therefore, we only want to learn in case the estimated dynamics $(\hat{A}, \hat{B}, \hat{\epsilon})$ deviate too much from the real dynamics (A, B, ϵ). Since the real dynamics are hidden, our decision needs to be based on some implicit feature, which will be the communication signal. Developing such a learning scheme for ETC is the main objective of this chapter.

4.3 Event-triggered Pulse Control with Adaptation through Learning

In this section, we present the control framework. We start with a derivation of the learning trigger, then show how we learn the system dynamics, and finally detail the derivation of the pulses.

4.3.1 Event-triggered Learning for Control

The learning trigger is based on the framework presented in [97] for ETSE. Here, we extend this framework to ETC. For the theoretical analysis, we assume a control strategy based on Dirac impulses, thus we have a control law of the form

$$u(t) = F\delta_{t_k}(t), \tag{4.4}$$

where F is the control gain and δ_{t_k} the Dirac impulse. In particular, the control input is zero apart from the triggering times t_k, where t_k corresponds to $\gamma_{\text{ctrl}} = 1$

in (4.2). To further investigate this, we write (4.1) in integrated form,

$$\begin{aligned} x\left(t_k\right) &= \int_{t_{k-1}}^{t_k} e^{A(t_k-t)} B u(t)\,\mathrm{d}t \\ &\quad + \underbrace{\int_{t_{k-1}}^{t_k} e^{A(t_k-t)} \epsilon\,\mathrm{d}t + \int_{t_{k-1}}^{t_k} e^{A(t_k-t)} Q\,\mathrm{d}W(t)}_{:=N(t_k)} \\ &= e^{A(t_k-t_k)} BF + N\left(t_k\right) \\ &= BF + N\left(t_k\right) \overset{!}{=} 0, \end{aligned} \tag{4.5}$$

where we assume that the process starts in $x(t_{k-1}) = 0$. If the matrix B is invertible, we can show that $F = B^{-1}N(t_k)$, where $N(t_k)$ is the measurement we get before applying the impulse, resets the system to zero. Implementing such a control law then also fulfills the prior assumption of $x(t_{k-1}) = 0$, as the system starts in zero after every triggering instant. In Section 4.3.3, we will drop the assumption of being able to apply Dirac impulses as inputs and instead apply pulses with the maximum input $u_{\max}$ for a given time.

Considering a control law as proposed in (4.4), we thus have a random process that always starts in zero. This is only true, if the input matrix B is known exactly. In that case, the sole cause of an error would be propagated noise and the load disturbance ϵ. Therefore, in case of no communication, we obtain

$$x(t) = \int_0^t e^{A(t-s)} \epsilon\,\mathrm{d}s + \int_0^t e^{A(t-s)} Q\,\mathrm{d}W(s). \tag{4.6}$$

We can now define a stopping time τ as the first moment the state crosses the threshold δ, which resets the error to zero,

$$\tau \coloneqq \inf\left\{t : \|x(t)\|_2 \geq \delta\right\}. \tag{4.7}$$

The stopping times defined in (4.7) coincide with the time between communication, hence, 'stopping times' and 'inter-communication times' will be used synonymously hereafter. We can now further define the expected value of these stopping times, $\mathbb{E}[\tau|x(0) = 0]$, which is the average communication rate of the system. This expected value can be obtained via Monte Carlo simulations (for a more detailed discussion, see [97]).

If we had a perfect model of the system dynamics, the average inter-communication times that we observe in the system should approach the expected value of the stopping time. If both values diverge, we have evidence that the model is inaccurate and can trigger learning of a new model. Precisely, we define the learning trigger in

Figure 4.1 as

$$\gamma_{\text{learn}} = 1 \iff \left| \frac{1}{N} \sum_{i=1}^{N} \tau_i - \mathbb{E}[\tau] \right| \geq \kappa. \tag{4.8}$$

In this equation, $\gamma_{\text{learn}} = 1$ indicates that a new model shall be learned, $\mathbb{E}[\tau]$ is approximated using Monte Carlo simulations, i.e., $\mathbb{E}[\tau] \approx \frac{1}{M} \sum_{i=1}^{M} \tau_i^{\text{sim}}$, and $\tau_1, \tau_2, \ldots, \tau_N$ define the last N empirically observed inter-communication times. Due to the randomness of the process, it can still happen that we trigger learning despite the model being perfect. Assuming that the stopping times are bounded by $\tau_{\max}$, the confidence level can be quantified using Hoeffding's inequality [142] and influenced through the design parameter η.

Theorem 4.1. *Let the parameters η, N, $M > N$, and $\tau_{\max}$ be given, $\tau_1, \ldots, \tau_N$ and $\tau_1^{\text{sim}}, \ldots, \tau_M^{\text{sim}}$ independent and identically distributed, and assume a perfect model. For*

$$\kappa = \tau_{\max} \sqrt{-\frac{2}{N} \ln \frac{\eta}{4}} \tag{4.9}$$

we obtain

$$\mathbb{P}\left[\left| \frac{1}{N} \sum_{i=1}^{N} \tau_i - \frac{1}{M} \sum_{i=1}^{M} \tau_i^{\text{sim}} \right| \geq \kappa \right] < \eta. \tag{4.10}$$

Proof. We compare stopping times obtained via Monte Carlo simulations with stopping times observed from the real process. In both cases, we have a random process that always starts in zero. This is the same setting as investigated in [97], thus, the theorem can be proven as shown therein. □

Boundedness of the stopping times can easily be ensured in practice by applying a control input the latest when $\tau_{\max}$ is reached. The parameter η then basically defines the tradeoff between accepting an inaccurate model or triggering learning despite the model being perfect.

Intuitively, η defines the probability that the error κ is observed, while empirical and expected stopping times are drawn from the same distribution (i.e., we have a perfect model). If this probability is below a predefined threshold, we learn a new model.

4.3.2 Model-Learning

For the derivation of the stopping times as well as for the final controller design we need knowledge of the full system dynamics (matrices A and B) and the load disturbance ϵ. To calculate the stopping times we additionally need knowledge of

the process noise variance Q. For this purpose, we rewrite system (4.1) in discrete time,

$$\begin{aligned} x(k+1) &= A_\mathrm{d} x(k) + B_\mathrm{d} u(k) + \epsilon + v(k) \\ &= \begin{pmatrix} A_\mathrm{d} & B_\mathrm{d} & \epsilon \end{pmatrix} \begin{pmatrix} x(k) \\ u(k) \\ 1 \end{pmatrix} + v(k). \end{aligned} \tag{4.11}$$

This way we can learn the system dynamics with standard least-squares techniques.

Having knowledge of the load disturbances, we can incorporate them in the control design in Section 4.3.3. This represents a suitable solution to replace the integral part of standard, periodic controllers.

Remark 4.1. Another problem that may be considered with this approach is the knowledge of the zero-level of the system. We are considering an equilibrium at $x(t) = 0$, but the measurements we typically get are only voltage signals from a sensor and what zero means for that system is not clear from the beginning. We can model this as a sensor bias, i.e., we would have the following system dynamics

$$x(k+1) = A_\mathrm{d} x(k) + B_\mathrm{d} u(k) + v(k) \tag{4.12a}$$

$$y(k) = x(k) + \xi, \tag{4.12b}$$

where ξ is the sensor bias. Rewriting this yields

$$\begin{aligned} y(k+1) &= x(k+1) + \xi \\ &= A_\mathrm{d} \left(y(k) - \eta \right) + B_\mathrm{d} u(k) + v(k) \\ &= \begin{pmatrix} A_\mathrm{d} & B_\mathrm{d} & (I - A_\mathrm{d})\, \xi \end{pmatrix} \begin{pmatrix} y(k) \\ u(k) \\ 1 \end{pmatrix} + v(k). \end{aligned} \tag{4.13}$$

That way we can identify the system dynamics and the sensor bias vial least-squares techniques.

Estimating both, a sensor bias and a load disturbance, is, however, not possible, as they are not distinguishable given the output data. Infinite combinations of bias and load disturbance could explain the sensor data equally well.

4.3.3 Implementation of Event-triggered Pulse Control with Input Saturation

If the state of the system exceeds the threshold δ we want to quickly reset it to zero. Application of Dirac impulses is not compatible with our assumption of an input saturation at the actuator.

Instead of applying Dirac impulses, we propose to apply the maximum input and vary the *duration* of the pulse. This has two main benefits: 1) The input cannot exceed the saturation and thus will drive the system state to its desired value, as it

will not be limited but applied as computed; 2) The system will be driven to zero as fast as possible, that way coming as close to the idealized Dirac input as possible. For the derivation of the length of the input we will restrict to first-order systems and later comment on extensions to higher-order systems.

To derive the length of an impulse we look at the system equation in integrated form. If no event is triggered, i.e., if the system is close to its desired state, we have $u(t) = 0$. At triggering times t_k we apply the maximum input,

$$x(t) = e^{at} x\left(t_k\right) + \int_{t_k}^{t} e^{a(t-s)} \left(b u_{\max} + \epsilon\right) \mathrm{d}s. \tag{4.14}$$

The input shall be applied for enough time such that the state becomes zero. So we can set (4.14) to zero and solve for t (setting $t_k = 0$),

$$0 \stackrel{!}{=} e^{at} x\left(0\right) + \int_{0}^{t} e^{a(t-s)} \left(b u_{\max} + \epsilon\right) \mathrm{d}s,$$

which leads to

$$t = \frac{1}{a} \ln\left(\frac{b u_{\max} + \epsilon}{a x\left(0\right) + b u_{\max} + \epsilon}\right). \tag{4.15}$$

In (4.14) and (4.15) we assumed the noise to be zero during the application of the pulse. For the Dirac impulse this holds, as the time of the application tends to zero. Here, we explicitly derive how long the input will be applied, hence, the system will during this time also be excited by noise and we will not be able to exactly drive it to zero. We explicitly take this into account when computing the stopping times. Instead of having a process that always starts in zero, we now have a process that starts in $x(0) \sim \mathcal{N}(0, \Sigma_0)$, with variance Σ_0.

For a system with state dimension larger than one, a single input will generally not be sufficient to drive the system state to zero. Instead, we would have to change between maximum and minimum input, what would lead to a bang-bang controller [143].

4.4 Numerical Study

For the numerical study, we will consider a collection of dynamical, first-order processes. For each system, we assume a remote controller that is colocated with the sensor, but needs to transmit its actuation commands over a communication network, where all controllers share the same network. We look at processes with different dynamics that we want to stabilize within the same range δ and with the same maximum input $u_{\max}$. However, the approach is also applicable for systems where δ and $u_{\max}$ are not identical.

Table 4.1: Comparison of the average inter-communication times for different investigated systems before and after learning.

System	Before	After	System	Before	After
1	44 ms	239 ms	6	128 ms	271 ms
2	136 ms	349 ms	7	56 ms	240 ms
3	107 ms	283 ms	8	55 ms	219 ms
4	135 ms	256 ms	9	89 ms	279 ms
5	163 ms	363 ms	10	34 ms	273 ms

We assume a continuous-time system that we discretize with a sample time of 1 ms. The sample time is not equal to the update interval of the communication system and is only limited by the maximum frequency of the timers in the processors used for controller and actuator. A fine discretization is necessary, as we will derive a continuous pulse length. The finer the discretization, the more accurate is the application of the pulse (and the earlier we notice if the system is outside the tolerable range).

We consider a continuous-time process of the form

$$\mathrm{d}x(t) = ax(t)\,\mathrm{d}t + b\,(u(t) + \epsilon)\,\mathrm{d}t + Q\,\mathrm{d}W(t). \tag{4.16}$$

The system has a maximum input of $u_{\max} = 1$ and we choose $\delta = 0.02$. We model the load disturbance to enter with the input, as for instance done in [144, p. 54]. The nominal dynamics are given with $a = 5$, $b = 3$, $\epsilon = 0.01$, and $Q = 10^{-4}$.

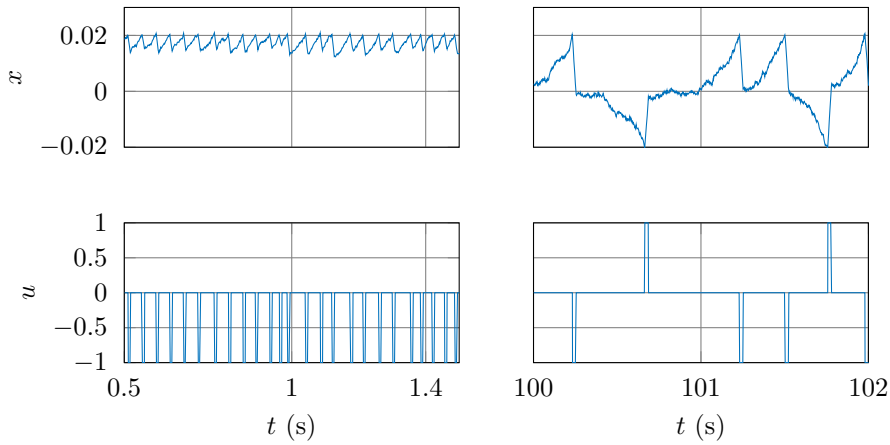
We demonstrate the applicability of the proposed algorithm by looking at ten specific systems, where parameters are uniformly sampled from intervals $a \in [1, 10]$, $b \in [1, 2]$, $\epsilon \in [0.01, 0.02]$, and $Q \in [10^{-4}, 10^{-3}]$.

As parameters of the learning trigger, we choose a confidence level $\eta = 0.05$, $N = 2000$, $M = 10\,000$, and $\tau_{\max} = 1\,\mathrm{s}$. According to Theorem 4.1, we then arrive at $\kappa \approx 0.066$. If learning is triggered, we use all data we have collected so far to learn a new model.

In Table 4.1, we compare the average inter-communication times of all three system before and after deriving new system matrices. For all of them we observe a significant increase in the inter-communication times after learning, i.e., communication is reduced. This demonstrates the potential of adapting dynamics models through learning.

In Figure 4.2, one specific system is investigated before (Figure 4.2a) and after (Figure 4.2b) learning new system matrices. Due to the error in the initial matrices, the system is not reset to zero with the pulses before learning and, thus, new control inputs have to be generated very frequently. After learning, the pulse length is such that the system is successfully reset, which also results in increased inter-communication times. This is especially emphasized as in Figure 4.2a, before learning, we show only 1 s, while in Figure 4.2b we show 2 s and still observe far less pulses.

To further investigate the learning performance of the approach, we look at the average and expected inter-communication times in Figure 4.3. The average inter-



(a) Before learning **(b)** After learning

Figure 4.2: Performance of one specific system before (left) and after (right) learning. It can be seen that before learning, the pulses are too short and the system is not reset to zero, while after learning the pulse length is appropriate. Further, communication is significantly reduced through learning.

communication times are computed with a moving average over 2000 stopping times, which we reset in case learning is triggered and after deriving new system matrices. For these simulations, we always collect data for 200 s in case learning is triggered and then only use these data points to learn the system dynamics. The system under investigation has parameters $(a, b, Q) = (5, 3, 10^{-4})$, without a load disturbance, and all other parameters as in the previous examples. In the beginning, we assume that we have an accurate model, hence, the observed inter-communication times approach the expected ones. After 2000 stopping times, the dynamics change, we then have $b = 2$ and a load disturbance $\epsilon = 0.03$. As expected, the inter-communication times decrease and learning is triggered. After learning, the empirical inter-communication times again approach the expected ones and we reduce communication. In a second change, after 6000 stopping times, the load disturbance ϵ is zero again and we have $(a, b) = (8, 1)$. Similar as before, this leads to a decrease of the inter-communication time and a learning experiment is triggered. Having learned new dynamics, the empirical inter-communication-times again approach the expected ones, i.e., average communication is reduced through learning.

The study reveals that the proposed architecture enables us to reduce inter-communication times through learning. We are able to learn system dynamics and subsequently reset the state of the system to zero in case it leaves its tolerable range. Through learning load disturbances, the architecture is a suitable replacement for integral control in event-triggered settings.

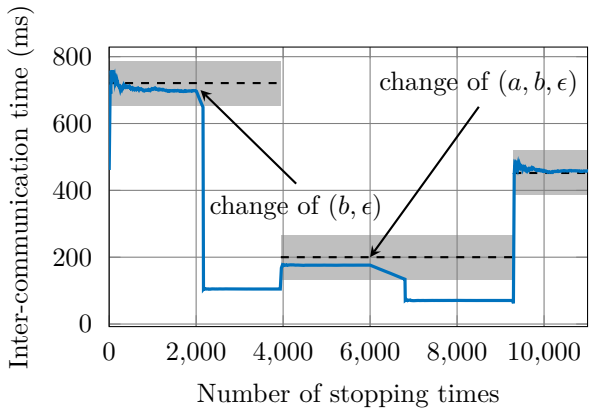


Figure 4.3: Average inter-communication times during one simulation. The solid line shows the empirically observed inter-communication times computed as an moving average over 2000 stopping times. The moving average is reset in case learning is triggered and when a new model has been learned. The dashed line indicates the expected inter-communication times with highlighted confidence interval in gray. The dynamics change after 2000 and after 7000 stopping times. In both cases, a decrease in the inter-communication time is observed and an increase after learning new matrices.

4.5 Conclusion

In wireless CPSs, communication is a scarce and limited resource. In this chapter, we presented a framework for event-triggered pulse control for wireless CPSs. Most common ETC approaches rely on the availability of an accurate dynamics model. Contrary to that, the proposed framework does not rely on this assumption, but uses model-learning instead. As learning is expensive (e.g., due to the involved computations), we only learn if necessary using the ETL framework. By observing the communication behavior we quantify the accuracy of the model and trigger learning of a new model only in case the accuracy is not sufficient.

Further, the presented control design respects input saturations at the actuator and can also handle load disturbances, essentially replacing the integral part of common periodic controllers.

A numerical study demonstrates the applicability of the approach and the benefit of learning the system dynamics. After learning, we observe a significant increase in the inter-communication time. However, the presented example is a first-order system. In future work, we seek to consider also higher order systems. Moreover, we assumed that we are able to perfectly measure the full state of the system. Incorporating Gaussian measurement noise is already possible with the presented approach. How to extend the ETL framework to partial state measurements is subject to ongoing research.

We employed a straightforward ETC scheme in this work in order to demonstrate

the benefits of ETL for control. Extensions of this idea to other ETC approaches is an interesting topic for further research.

In this work, we proposed for the first time to limit learning in ETC through ETL. We triggered learning experiments through comparing the expected and the observed time between communication. While this is an intuitive approach, in some cases this trigger does not detect disturbed models. A more robust behavior can be achieved by triggering on the full distribution, e.g., via a Kolmogorov-Smirnoff test [145]. Deriving similar theoretical guarantees as for the expected value trigger is also subject to ongoing work.

Chapter 5

Deep Reinforcement Learning for Event-Triggered Control

In the last chapter we outlined an approach for automatically learning system models and an event-triggered strategy based on these models. In this chapter we will, as an alternative, present an approach for end-to-end learning of ETC, without a manually designed control strategy. In contrast to existing approaches, we will learn both, the control policy and the communication strategy, simultaneously. We will further demonstrate that this approach straightforwardly extends to nonlinear systems.

5.1 Introduction

Most ETC approaches depend on a predesigned control strategy, including for instance a (fixed or dynamic) triggering threshold. In contrast to this, we show herein that *deep reinforcement learning* (DRL) algorithms can be leveraged in order to learn both control *and* communication law from scratch without the need for a dynamics model. We formulate resource-aware control as a RL problem, where the learning agent optimizes its actions (control input and communication decision) so as to maximize some expected reward over a time horizon. The reward function is composed of two terms, one capturing control performance, and one that gives rewards for time steps without communication. This way, the agent learns to control the system with good performance, but without communicating all the time.

More specifically, we consider the setup in Figure 5.1. While the learning agent is directly connected to the sensor and thus receives its measurements continuously, it must transmit actuator commands over a wireless network link. Thus, we seek to reduce communication of control inputs. We propose two approaches for learning ETC in this setting. In the first approach, we assume that a time-triggered feedback controller is given, and we learn only the communication policy. As an alternative, we learn both control and communication policy simultaneously. This can be regarded as *end-to-end learning*. In DRL, end-to-end learning (e.g., [86]) typically refers to

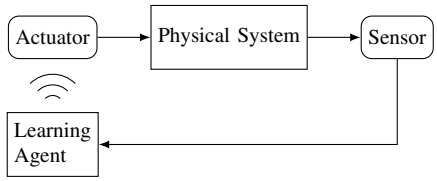


Figure 5.1: Learning of event-triggered control. The learning agent continuously receives sensor inputs, but has to transmit control signals over a resource-limited wireless network. The agent learns both control and communication; that is, (i) what actuator command to send, and (ii) when to send it.

learning the complete control policy from raw sensor data to actuator commands 'end to end,' without (artificially) separating into sub-tasks such as filtering, planning, and tracking. In the context of ETC, end-to-end thus emphasizes learning of both control and communication simultaneously, rather than separating the two. This is particularly interesting as the separation principle does not generally hold in ETC; that is, optimizing controller and communication structure separately, as often done in practice, does not necessarily yield the overall optimal event-triggered control law [146]. End-to-end DRL is a way to overcome this separation.

By means of numerical examples, we demonstrate that end-to-end learning of ETC is feasible. Moreover, we compare to some common model-based ETC approaches. The comparison reveals that, for linear settings with an accurate model available, model-based ETC typically cannot be outperformed by the proposed DRL approach—at least, at medium to high average communication rates. In some cases, however, DRL can find superior policies at very low communication rates, where model-based ETC yields unstable solutions. In contrast to common ETC methods, the proposed learning approach straightforwardly applies also to nonlinear control problems.

Contributions The contributions of this chapter can be summarized as follows:

- Proposal of DRL to learn event-triggered controllers from data;
- learning of communication policy only (with a given controller) with policy gradients [90];
- end-to-end learning of control *and* communication policy with deep deterministic policy gradient (DDPG) algorithm [89];
- demonstration of feasibility of DRL in numerical benchmark problems; and
- comparison to model-based ETC methods.

Outline The next section continues with a introduction to DRL with particular focus on approaches for continuous state-action spaces. The proposed approaches

for DRL of ETC are then introduced in Section 5.3. Section 5.4 presents numerical results, and the chapter concludes with a discussion in Section 5.5.

5.2 Background

We consider a discretized version of (1.1), i.e.,

$$x(k+1) = f\left(x(k), u(k)\right) + v(k), \tag{5.1a}$$

where we assume that the full state can be measured, but measurements are corrupted by Gaussian noise,

$$y(k) = x(k) + w(k). \tag{5.1b}$$

5.2.1 Deep Reinforcement Learning

We give a brief introduction to RL in general and present the two baseline algorithms we later focus on in Section 5.3.

The main goal in RL is to learn an optimal policy by trial and error while interacting with the environment. Mathematically, this can be formulated as a Markov decision process (MDP). In an MDP, we consider the setting where an agent interacts with the environment. At every time step, the agent selects an action $a(k)$, from the action space A, based on its current state $s(k)$, from the state space S, according to a policy $\pi\left(a(k)|s(k)\right)$.[1] The agent receives a reward $r(k)$ and the state transitions to the next state $s(k+1)$ according to the state transition probability $p(s', r|s = s(k), a = a(k))$. The goal of the RL agent is to maximize the expected discounted reward $\mathbb{E}[R(k)] = \mathbb{E}\left[\sum_{i=0}^{\mathrm{T}-1} \zeta^i r(k+i)\right]$, where $\zeta \in (0, 1]$ is the discount factor.[2] There are generally two types of RL methods: model-free and model-based. One model-free method to achieve the goal is to learn a value function $v_\pi(s) \stackrel{!}{=} \mathbb{E}\left[R(k)|s(k) = s\right]$, which denotes the expected return in case policy π is followed from state s onwards. The value function $v_\pi(s)$ follows the Bellman equation [147],

$$v_\pi(s) = \sum_a \pi\left(a|s\right) \sum_{s',r} p\left(s', r|s, a\right)\left[r + \zeta v_\pi(s')\right], \tag{5.2}$$

which can then be maximized to find the optimal state values.

Similarly, one can estimate a state-action value function (Q-function) $Q_\pi\left(s, a\right) \stackrel{!}{=} \mathbb{E}\left(R(k)|s(k) = s, a(k) = a\right)$ which determines the expected return for selecting action a in state s and following the policy π thereafter. In an MDP, the optimal

[1] If we want to learn a controller for a dynamical system, often $s(k) \equiv x(k)$ and $a(k) \equiv u(k)$ holds. However, this is not necessarily the case and, in particular, not in the setup we shall develop herein.

[2] To simplify the formulation, we consider the episodic case with $k \in [0, T-1]$.

action in the current state can be derived by maximizing the Q-function. If the transition probabilities p are available, this can, e.g., be done using (exact) dynamic programming (DP). In cases where the model is not known, we resort to RL (simulation-based approximate DP) methods. In such cases, the Q-function can be learned using the Q-learning algorithm presented in [148],

$$\begin{aligned} Q(s(k), a(k)) \leftarrow & Q(s(k), a(k)) \\ & + \alpha \left(r(k) + \zeta \max_{a(k)'} Q\left(a(k)', s(k)'\right) - Q(s(k), a(k)) \right). \end{aligned} \tag{5.3}$$

The Q-learning algorithm updates the Q-function using the collected experience $(s(k), a(k), r(k), s(k)')$. For a more detailed introduction to RL, see [149].

This basic RL approach has successfully been applied to low-dimensional tasks with discrete state and action space [149]. For controlling a dynamical system, we usually deal with a continuous state and action space, which might be of high dimension for complex systems. Continuous spaces could be discretized, but the discretization needs to be very fine for high-performance control. This, in turn, leads to very high-dimensional state and action spaces imposing unreasonable computational complexity and hampering convergence speed drastically.

To the rescue come parametrized function approximators. In machine learning, deep neural networks (DNNs) have widely been used to handle high-dimensional tasks. Recently, they have also been applied to RL, giving rise to the field of DRL. For instance, in deep Q-learning [150], the state-action function Q is approximated with a DNN, making it possible to solve complex tasks in high-dimensional continuous state spaces. However, this algorithm only works for discrete action spaces.

One possible solution to this problem is the actor-critic architecture [149]. The actor outputs continuous actions while the critic estimates the value function. Both can be implemented using DNNs. One such algorithm is deep deterministic policy gradient (DDPG) [89, 151], which we introduce in the following.

As an alternative, we look at policy search methods that directly learn a policy without a Q-function. Specifically, we will present the policy gradient algorithm [90] and the trust region policy optimization (TRPO) algorithm [152].

DDPG

The DDPG algorithm, and a variation of it, are presented in [89, 153]. For completeness, we restate the main derivations.

DDPG is an actor-critic algorithm with two networks. One is the actor network μ, parametrized by θ^μ that takes the state $s(k)$ as input and outputs an action $a(k)$. Additionally, we have the critic network Q, parametrized by θ^Q, which takes state and action as input and outputs a scalar estimate of the value function, the Q-value $Q\left(s(k), a(k)\right)$. The updates of the critic network are close to the original formulation of the Q-learning algorithm given in (5.3). Adapting (5.3) to the described neural

network setting leads to minimizing the loss function

$$L_Q\left(s(k), a(k)|\theta^Q\right) = \Bigg(Q\left(s(k), a(k)|\theta^Q\right) - \left(r(k) + \zeta \max_{a(k)'} Q\left(s(k)', a(k)'|\theta^Q\right)\right)\Bigg)^2. \tag{5.4}$$

For continuous action spaces, equation (5.4) is not tractable, as we would have to maximize over the next-state action $a(k)'$. Instead, we take the next-state action $a(k)' = \mu\left(s(k)'|\theta^\mu\right)$ of the actor network. Inserting this in equation (5.4) leads to

$$L_Q\left(s(k), a(k)|\theta^Q\right) = \left(Q\left(s(k), a(k)|\theta^Q\right) - \left(r(k) + \zeta Q\left(s(k)', \mu\left(s(k)'|\theta^\mu\right)|\theta^Q\right)\right)\right)^2. \tag{5.5}$$

Based on this loss function, the critic can learn the value function via gradient descent. Clearly, a crucial point is the quality of the actor's policy. The actor tries to minimize the difference between its current output a and the optimal policy a^*,

$$L_\mu\left(s(k)|\theta^\mu\right) = \left(a(k) - a(k)^*\right) = \left(\mu\left(s(k)|\theta^Q\right) - a(k)^*\right)^2. \tag{5.6}$$

The true optimal action $a(k)^*$ is of course unknown. As simply estimating it would require to solve a global optimization problem in continuous space, the critic network can instead provide a gradient that leads to higher estimated Q-values: $\nabla_{a(k)} Q\left(s(k), a(k)|\theta^Q\right)$. Computing this gradient is much faster. This was first introduced in [154]. The gradient implies a change in actions, which is used to update the actor network in this direction by backpropagation. In particular, for an observed state $s(k)$ and action $a(k)$, the parameters of the actor network are changed according to

$$\nabla_{\theta^\mu} J = \nabla_{a(k)} Q\left(s(k), a(k)|\theta^Q\right) \nabla_{\theta^\mu} \mu\left(s(k)|\theta^\mu\right) \tag{5.7}$$

approximating the minimization of (5.6).

Two general problems arise from this approach. For most optimization algorithms, it is usually assumed that samples are independent and identically distributed. This is obviously not the case if we sequentially explore an environment. To resolve this, a replay buffer of fixed size that stores tuples $(s(k), a(k), r(k), s(k+1))$ is used. Actor and critic are now updated by uniformly sampling mini-batches from this replay buffer.

The second problem is that the update of the Q-network uses the current Q-network to compute the target values (see (5.5)). This has proved to be unstable in many environments. Therefore, copies of actor and critic networks, $Q'\left(s(k), a(k)|\theta^{Q'}\right)$ and $\mu'\left(s(k)|\theta^{\mu'}\right)$ are created and used to calculate the target values. The copies are updated by slowly tracking the learned network,

$$\theta' = \kappa\theta + (1-\kappa)\,\theta', \tag{5.8}$$

with $\kappa \ll 1$. This typically leads to more robust learning.

In Section 5.3.1, we will show how this algorithm can be used to jointly learn controller and communication behavior.

Trust Region Policy Optimization (TRPO)

A second approach is to perform direct parameter search without a value function. This is referred to as direct policy search or policy gradient [90]. A parametrized policy π is adapted directly to maximize the expected reward. Since, without a model, analytical gradients of the reward function are not available, policy gradient methods use stochastic policies and adapt them to increase the likelihood of a high reward. Formally, let the policy $\pi(s(k);\theta^\pi)$ representing $p(a(k)|s(k))$ be parametrized by θ^π in a differentiable way. Now we aim to maximize the utility $J(\theta^\pi) = \mathbb{E}_{a(k)\sim\pi(s(k);\theta^\pi)} R(s(k), a(k))$. Policy gradient methods follow the gradient estimator of J for a given trajectory:

$$\nabla_{\theta^\pi} J(\theta^\pi) = \sum_{k=0}^{T-1} R(k) \nabla_{\theta^\pi} \log \pi(s(k);\theta^\pi)\,. \tag{5.9}$$

A recent advance of policy gradient methods is given by the Trust Region Policy Optimization [152] (TRPO) that uses a surrogate optimization objective and a trust region approach for updating the policy efficiently. In terms of theoretical guarantees, this algorithm ensures monotonic improvement of the policy performance, given the amount of training samples is large. We will use this method in Section 5.3.2 to learn the controller and triggering policy independently.

5.3 Approach

We present two approaches to learn resource-aware control. First, we consider learning communication structure and controller end-to-end. The policy should then output both, the communication decision and the control input,

$$(\gamma(k), u(k)) = \pi_{\text{combined}}\left(s(k)\right) = \pi_{\text{combined}}\left(x(k), u(k-1)\right), \tag{5.10}$$

where $\gamma(k)$ is a binary variable with $\gamma(k) = 0$ indicating no communication. Alternatively, we start with a control strategy for the system without communication constraints, either learned or designed. The goal is then to learn the communication structure, i.e., a policy

$$\gamma(k) = \pi_{\text{comm}}\left(s(k)\right) = \pi_{\text{comm}}\left(x(k), u(k), u(k-1)\right). \tag{5.11}$$

This strategy requires us to separate the design of controller and communication structure.

For both settings, the state of the RL agent includes the current state $x(k)$ of the system and the last control input $u(k-1)$. This is necessary, as in case of no

communication, $u(k-1)$ will be applied again, so knowledge of the last control input is needed for the problem to form an MDP. In (5.11), the state is further augmented and also includes the current control input $u(k)$. The RL agent learns a communication policy, i.e., it needs to decide, whether $u(k)$ or $u(k-1)$ will be applied. Therefore it needs knowledge of both. The action $a(k)$ of the RL agent consists of the communication decision for the separated policy, and of communication decision and control input in the combined case.

In RL, the reward function typically depends on the states and actions of the system. We additionally consider communication, thus we arrive at a reward function of the form

$$r(k) = -x(k)^{\mathrm{T}} Q x(k) - u(k)^{\mathrm{T}} R u(k) - \lambda \gamma(k), \tag{5.12}$$

where λ is a hyper-parameter. During training, the agent receives negative rewards for bad performance and for communication. In an episodic reinforcement learning task, where agents' interaction with the environment is divided into episodes, an additional constant positive reward is often given to the agent to prevent undesired early termination of the episodes, e.g., the pole dropping for the cart-pole system.

5.3.1 Joint Learning of Communication and Control

To learn resource-aware controllers, we consider both the discrete action space (the decision whether to communicate) and the continuous action space (the control input that should be applied). This is related to the idea of reinforcement learning in parameterized action space [153, 155].

This framework considers a parameterized action space Markov decision process (PAMDP), which involves a set of discrete actions $A_{\mathrm{d}} = \{d_1, d_2, \ldots, d_k\}$. Each discrete action $d \in A_{\mathrm{d}}$ is associated with m_{d} continuous parameters $\left\{p_1^{\mathrm{d}}, p_2^{\mathrm{d}}, \ldots, p_{m_{\mathrm{d}}}^{\mathrm{d}}\right\} \in \mathbb{R}^{m_{\mathrm{d}}}$. An action is represented by a tuple $\left(d, p_1^{\mathrm{d}}, \ldots, p_{m_{\mathrm{d}}}^{\mathrm{d}}\right)$. This leads to the following action space: $A = \cup_{\mathrm{d} \in \mathrm{A_d}} \left(d, p_1^{\mathrm{d}}, \ldots, p_{m_{\mathrm{d}}}^{\mathrm{d}}\right)$.

In our case, there are two discrete actions, d_1 and d_2, where d_1 corresponds to the decision to communicate the control input ($\gamma(k) = 1$). Accordingly, the action d_1 has $m_{\mathrm{d}_1} = 1$ continuous parameter $p_1^{\mathrm{d}_1} = u$, which is the control input. Action d_2 does not have any continuous parameter, as we apply the last control input again.

As stated in Section 5.2.1, we consider the DDPG algorithm[3], where both actor and critic network are implemented using DNNs. The architecture is depicted in Figure 5.2. The actor network outputs continuous values for all actions in the action space, i.e., we have

$$a(k) = (d_1(k), d_2(k), u(k)) = \pi_{\mathrm{combined}}\left(x(k), u(k-1)\right). \tag{5.13}$$

This is different from (5.10), as we do not receive a discrete parameter $\gamma(k)$, but continuous values for all parameters $a(k)$. To obtain a discrete decision, we determine

[3] Our implementation is based on the non-parameterized DDPG framework provided in [156].

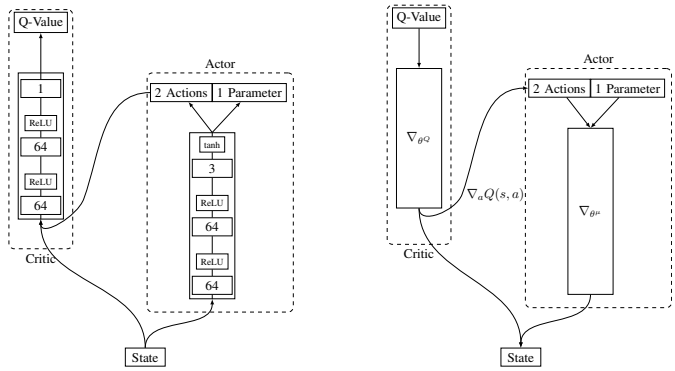


Figure 5.2: Visualization of the actor-critic network structure (adapted from [153]). On the left, the general network architecture, showing the units and activation function of each layer. Each block represents one layer of the network with the number describing the number of neurons. The smaller blocks indicate the activation functions. On the right, the update of the actor using back-propagation.

the communication decision by

$$\gamma(k) = \begin{cases} 1 & \text{if } d_1(k) > d_2(k) \\ 0 & \text{otherwise.} \end{cases} \tag{5.14}$$

The continuous parameter (the control input $u(k)$) is directly obtained as an output of the actor. The output of the actor and the current state then serve as input for the critic, which estimates the Q-function value. This structure has been applied to a gaming environment in [153].

During training, exploration is done in an ϵ-greedy fashion. With probability ϵ, we select a random discrete action (whether to communicate). Besides the ϵ-greedy exploration we add exploration noise in form of an Ornstein Uhlenbeck process to the output of the actor as has been successfully demonstrated in [89]. Pseudo-code of this approach is presented in Algorithm 1.

5.3.2 Learning Communication only

An alternative to the aforementioned end-to-end approach is to separately learn the communication strategy and the stabilizing controller. In this approach, a control policy is first fully trained using a high-performing RL algorithm, e.g., TRPO [152]. Instead of hand-engineering the communication strategy, we propose to use policy gradients [90] to learn this communication structure. In essence, the trained controller computes the control input in every time-step, whereas another learning agent controls whether to send this control input to the system, thus implementing (5.11).

Algorithm 1 Jointly learn communication and controller (adapted from [89]).

Initialize ϵ
Randomly initialize critic DNN $Q\left(s(k), a(k)|\theta^Q\right)$ and actor DNN $\mu\left(s(k)|\theta^\mu\right)$ with weights θ^Q and θ^μ.
Initialize target networks Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathcal{R}$
for episode = 1 to N **do**
  Receive initial observation state $s(1)$
  for k = 1 to M **do**
    Generate uniformly distributed $\phi \in [0, 1]$
    if $\phi < \epsilon$ **then**
      Generate $\xi \sim B\left(2, 0.5\right)$ from Bernoulli distribution
      if $\xi == 1$ **then**
        Choose discrete action d_1
      else
        Choose discrete action d_2
      end if
    else
      Select $a(k) = \mu\left(s(k)|\theta^Q\right)$ and apply exploration noise to the actor output
      Get communication decision $\gamma(k)$ using (5.14)
    end if
    Execute action $a(k)$, receive reward $r(k)$ and state $s(k+1)$
    Store transition $(s(k), a(k), r(k), s(k+1))$ in $\mathcal{R}$
    Sample random mini-batch from $\mathcal{R}$
    Update critic by minimizing loss function (5.5)
    Update actor policy using sampled policy gradient (5.7)
    Update target networks according to equation (5.8)
  end for
end for

The general scheme is related to hierarchical reinforcement learning [157] and gated recurrent neural networks [158]. We discuss preliminary experimental results of this alternative approach, in relatively challenging tasks, in Section 5.4.4.

5.4 Validation

In this section, we validate the proposed DRL approaches through several numerical simulations. For the algorithm introduced in Section 5.3.1, which jointly learns communication behavior and controller, we show the general applicability as a proof of concept on the inverted pendulum, compare to several model-based ETC algorithms on the same platform, and show its general applicability for nonlinear tasks. In Section 5.4.4, we demonstrate learning resource-aware locomotion tasks

using the algorithm presented in Section 5.3.2.[4]

The numerical simulations presented in this section were carried out in environments adapted from the OpenAI Gym[5]. The OpenAI Gym provides simulation models of different classical control tasks, such as the inverted pendulum and the cart-pole system, as well as physics simulation systems, Atari games, and many more. For our approaches, we augment the reward functions provided in the OpenAI Gym according to (5.12). The simulations are carried out on a cluster utilizing parallel runs for the training and testing processes with randomized seeds.

5.4.1 Proof of Concept

As a proof of concept, we apply the learning algorithm presented in Section 5.3.1 to the inverted pendulum. The inverted pendulum consists of a pendulum attached to a motor with the goal to keep the pendulum close to its upright position at $\theta = 0\,\mathrm{rad}$. We assume process and measurement noise as in (5.1) and the initial state also a Gaussian distributed random variable with $x(0) \sim \mathcal{N}(x(0); 0, \Sigma_0)$. The standard deviation of noise and initial position was chosen to be 10^{-4}.

The simulation environment provides upper and lower bounds of $\pm 2\,\mathrm{N\,m}$ on the input torque that may be applied to the pendulum. One discrete time step lasts 50 ms.

We train the controller using the joint learning approach detailed in Section 5.3.1. The hyper-parameter λ in (5.12) is tuned by a grid search of 25 values between 0.01 and 100. Different hyper-parameter values correspond to different communication rates and controller performances. For each hyper-parameter setting and task, we carry out 5 randomized training processes using different random seeds, each consists of one million training iterations. During performance evaluation, we carry out 100 randomized test episodes for each of the 5 trained agents for each hyper-parameter setting with each episode lasting 500 discrete time steps.

Results of one such test episode can be seen in Figure 5.3. The plot is a representative example for the results obtained from the different agents and test episodes. The pendulum system remains stable with the angle staying well within $\pm 0.1\,\mathrm{rad}$, while significantly saving communication. Here we observe a saving rate of around 90 %. Further it can be seen that the learning approach does not converge to a triggering law with fixed threshold. The threshold at which communication of a new control input is triggered is dynamically changing throughout the experiment.

In general, stability is very important in control tasks. However, most works in reinforcement learning aim at achieving optimal control instead of stability. For policy iteration methods (which we use herein), guarantees on monotonic improvement of the policy can be given, as discussed for instance in [152, 159]. However, the resulting controller derived in this work is approximated by a deep neural network, which is highly nonlinear, thus analyzing the stability of the system is not straightforward.

[4] Code of representative examples and video of resource-aware locomotion are available at `https://sites.google.com/view/drlcom/`.

[5] `https://gym.openai.com/`

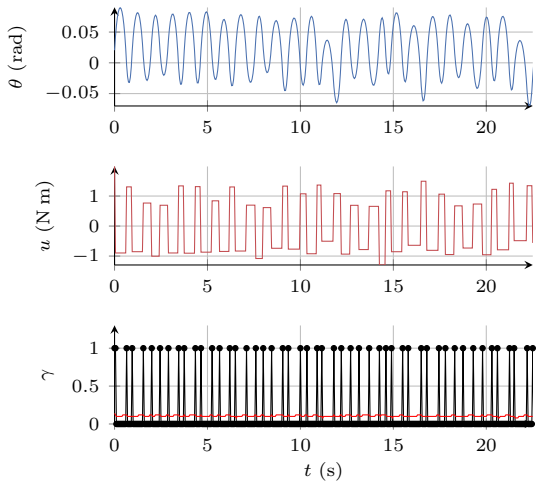


Figure 5.3: Stabilization of the inverted pendulum with an event-triggered controller learned with the method presented in Section 5.3.1. The plots show, from top to bottom, the angle of the pendulum θ, the control input u, and the communication (decision γ in black, average communication in red). The average communication here and in following plots is computed as a moving average over 50 samples.

Further, finding optimal control policies does not necessarily imply stability, but the connection is more subtle (cf. [160, 161]). For the time being, this renders the effort of analyzing stability intractable. While stability of DRL is an important topic for research, this example is a proof of concept that joint control and communication policies can be found with DRL.

5.4.2 Comparison

We compare the performance of the learning approach to common model-based ETC designs on the inverted pendulum. For balancing, the inverted pendulum can be approximated as a linear system and methods from linear control theory may be used. We consider the intuitive ETC algorithm, where we only communicate, if the state deviates too much from its desired position. The state of the inverted pendulum consists of its angle θ and its angular velocity $\dot{\theta}$, the desired value for both is zero. Hence, we apply the following control law

$$u(k) = \begin{cases} Kx(k) & \text{if } \|x(k)\|_2 > \delta \\ u(k-1) & \text{otherwise,} \end{cases} \tag{5.15}$$

where the matrix K is designed with an LQR approach using Q and R as in the reward function of the learning algorithm. Additionally, we compare to the approaches introduced in [162] and [163]. In both cases, we use the formulation as a periodic event-triggered control algorithm provided in [164], which is $\gamma(k) = 1 \iff$

$\|K\hat{x}(k) - Kx(k)\| > \delta \|Kx(k)\|$ for [162], and $\gamma(k) = 1 \iff \|\hat{x}(k) - x(k)\| > \delta \|x(k)\|$ for [163]. The algorithms only give a communication threshold, but require a stabilizing controller K. For both, we used the same LQR as for (5.15).

The algorithms have different triggering laws but are all based on a fixed threshold δ. For comparison, we vary this threshold. After every experiment, we compute the quadratic cost and the average communication. These simulations revealed that communication savings up to around 60 % for [163], 70 % for [164], and 80 % for (5.15) are possible. When running similar simulations with the DDPG approach from Section 5.3.1, we noted that the model-based approaches clearly outperform the learning approach. However, communication savings of 90 %, as observed in Figure 5.3, cannot be achieved with these model-based approaches as they become unstable before. The learning agent, in contrast, is still able to come up with a good policy.

5.4.3 Swing-up

As previously stated, the presented DRL approach can also be applied to more challenging, e.g., nonlinear, systems. In this section, we take on such a setting where the aforementioned ETC designs do not apply. The training and evaluation procedures in this section follow the same paradigm detailed in Section 5.4.1.

In Figure 5.4, the inverted pendulum is presented again, but with the initial angle well beyond the linear region. As can be seen, the agent is able to learn a resource-aware swing-up policy and then stabilize the pendulum around $\theta = 0\,\text{rad}$ while saving around 80 % communication.

We also trained the learning agent on the cart-pole system, where it was similarly able to learn a stable policy while saving around 90 % of communication (with an underlying sample time of 25 ms). The result of an experiment is shown in Figure 5.5.

5.4.4 Simulated locomotion

So far, we have addressed canonical tasks in optimal control using the proposed end-to-end approach. In this section, we move the focus to advanced tasks, i.e., locomotion. We applied the proposed parameterized DDPG approach of Section 5.3.1 to resource-aware locomotion, but only with moderate success. This is possibly due to the lack of reward hand-engineering and is left for future work considerations. During our experiments, we discovered that learning controller and communication behavior separately, as explained in Section 5.3.2, allows us to address even challenging tasks such as robotic locomotion. In this approach, we first train the agent with full communication using TRPO, typically using iteration numbers on the 10^6 order-of-magnitude. After the TRPO agent is trained, we train the communication strategy using a policy gradient approach with augmented reward as in (5.12) until we observe desired behaviors trading off performance and communication saving. Our experimental environment is based on the Mujoco physics simulation engine[165].

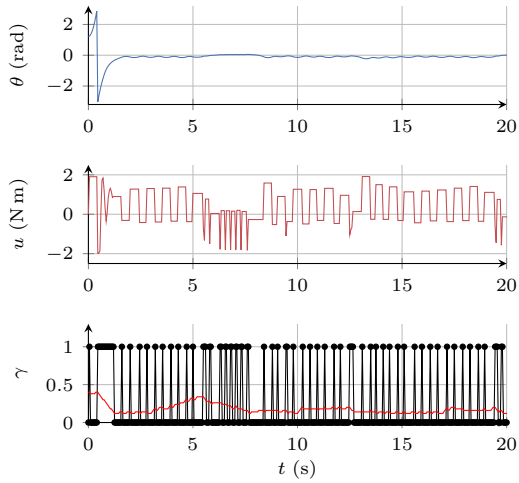


Figure 5.4: Resource-aware swing-up of the inverted pendulum, showing the angle θ (top) and the communication (bottom, discrete decision in black, average communication in red). The jump observed in the beginning is due to the pendulum crossing π and thus immediately switching from π to $-\pi$.

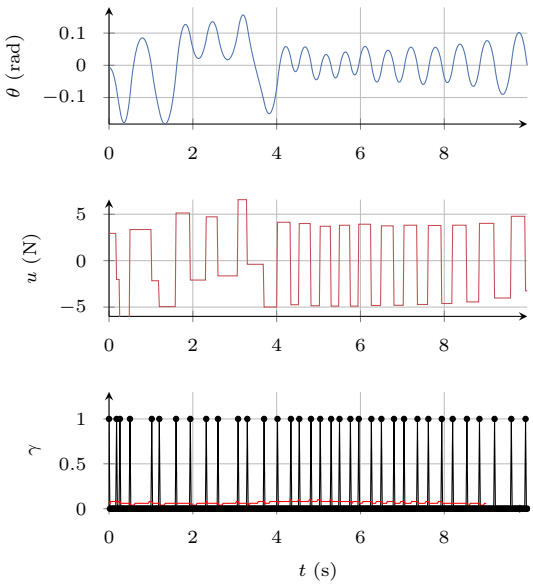


Figure 5.5: Resource-aware control of the cart-pole system.

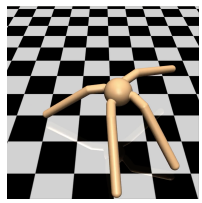


Figure 5.6: The ant robot.

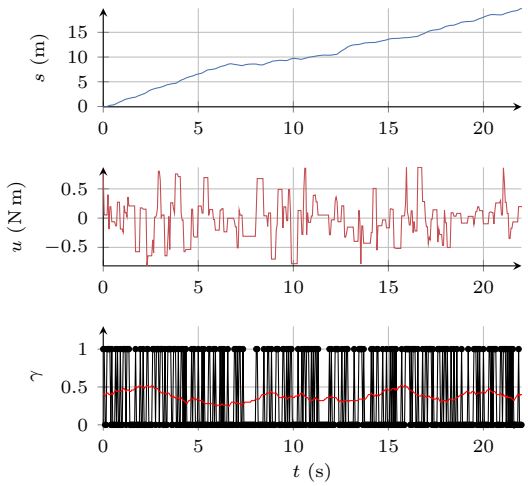


Figure 5.7: Simulation of the Ant robot (see Figure 5.6) learning to walk while saving communication, showing from top to bottom the position s of the center of mass, the input u, which is the torque applied to the hip motor, communication instants in black and the average communication in red.

As an example we trained an ant (quadruped) robot (Figure 5.6) in a simulated 3D locomotion task. It is a relatively challenging task considering the high-dimensional state space (111 states with 13 for position, 14 for velocity, 84 for external force) and under-actuation (8 actuators). To make matters worse, it can be easily toppled and is then subsequently not able to stand up. The underlying sampling time is 50 ms.

As shown in Figure 5.7, the ant learns to walk saving around 60 % of communication. We did observe that, during some of the runs, the resource-aware ant falls and causes worse performance. However, this happens only around 10 % of the time. As our method is task agnostic and not specifically engineered for locomotion tasks, we consider the performances and communication savings non-trivial.

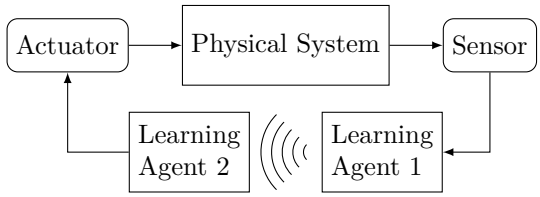


Figure 5.8: Learning resource-aware control with two learning agents. The first agent continuously receives measurements and can transmit information to the second agent over a wireless channel with limited bandwidth. The second agent can continuously apply control commands.

5.5 Discussion

Most existing approaches in ETC rely on the availability of accurate dynamics models for the design of control law and event trigger. In contrast, we proposed the use of DRL for simultaneously learning control *and* communication policies from simulation data without the need of an analytical dynamics model. For scenarios where an accurate linear model is available, the numerical comparisons herein have shown that common model-based ETC approaches are superior to the learning approach. This is to be expected because the model-based design fully exploits the model structure. For some cases, however, the DRL approach succeeded in finding stabilizing controllers at very low average communication rates, which the model-based designs were unable to obtain. What is more, the key advantage of the learning-based approach lies in its versatility and generality. As the examples herein have shown, the same algorithm can be used to also learn control and communication policies for nonlinear problems, including complex ones like locomotion. In the presented example, significant communication savings of around 60 % were obtained.

One limitation of our current approaches is the zero-order hold employed at the actuator. Instead of zero-order-hold, some model-based approaches perform predictions based on the dynamics model in case of no communication, and thus achieve better performance. This could also be done if learning agents are used and would lead to a two agent problem as depicted in Figure 5.8. The first agent continuously receives measurement updates and decides when to transmit data to the second agent. The second agent can continuously apply control inputs, which includes the possibility of making predictions based on a learned model. Investigating such more general learning architectures is an interesting and challenging topic for future work. Whether theoretical guarantees such as on stability and robustness can also be obtained for the learned controllers is another topic worthwhile to be investigated.

Chapter 6

Summary and Future Work

This chapter summarizes the results of this thesis and highlights future research directions.

6.1 Summary

Cyber-physical systems (CPSs) are expected to tightly interact with each other and the physical world while leveraging available data for learning. This enables emerging application, such as autonomous driving or smart factories. As a stepping stone towards that vision, this thesis presented results on fast and resource-efficient control of wireless CPSs. Typical wireless imperfections, such as packet losses, delays, and bandwidth constraints, were addressed by a tight integration of communication protocol and control strategy. Classical control concepts were then enhanced by machine learning techniques to get rid of the assumption that accurate dynamics of the systems are available and that these dynamics are time-invariant.

Chapter 2 presented for the first time feedback control over low-power wireless multi-hop networks with update intervals of 20-50 ms. A network protocol was developed that tamed network imperfections to the extent possible. Remaining imperfections (constant delay and rare, i.i.d. packet losses) were then taken into account in the control design to achieve stable closed-loop behavior. Stability was formally proven and demonstrated in experiments on a cyber-physical tested. Moreover, the network provides a many-to-all communication structure, allowing to straightforwardly implement distributed control. As an example for distributed control, synchronization of multiple physical systems over a low-power wireless multi-hop network was presented.

Chapter 3 took on the challenge of limited bandwidth inherent in wireless communication channels. Typical event-triggered control (ETC) and state estimation methods take instantaneous decisions about whether to transmit information or not. That way, the communication system cannot react and reallocate resources, thus, resource savings are not possible. In contrast to that, novel triggering strategies were introduced that predict communication requirements in advance. This allows to

reallocate resources in case of no communication and thus to actually save resources.

In Chapter 4, the assumption of having an accurate system model available was dropped. An event-triggered pulse control strategy was introduced and, based on a statistical analysis of the inter-communication times, model quality was evaluated. In case of a bad model, learning of a new model was triggered. Apart from the model, possible load disturbances were learned as well, thus, the approach represents a replacement of the integrator typically used in periodic control.

Independently optimizing control and communication strategy does not necessarily yield the optimal overall ETC strategy. Moreover, model-based approaches often only work for linear dynamics. Chapter 5 presented an end-to-end learning approach that simultaneously learns communication and control policy from scratch. The approach straightforwardly generalizes to nonlinear systems and communication savings of around 60 % were demonstrated on a simulated robotic system.

6.2 Future Work

To arrive at CPSs that autonomously act in the real world, make efficient use of available resources, and acquire new skills or improve their behavior through learning, there are more challenges to be overcome.

In the thesis, we showed how fast physical systems can be controlled over wireless networks. However, this was done in a time-triggered fashion, i.e., communication was running at the fastest periodic rates the network was able to support. We also presented frameworks to schedule communication in an event-triggered way, but without implementing them in a real wireless setting. In future work, we seek to combine event-triggered approaches with the wireless communication system. As a first step towards this direction, we will enhance the flexibility of the wireless control system by allowing for switching between different schedules, e.g., switching between synchronization and stabilization tasks. Next steps involve the integration with the concepts presented in Chapter 2. By deciding about future communication demands in advance, we want to show that we can save not only energy, but also resources. This will complement the integration at design time by an integration at runtime. However, the concepts presented in Chapter 2 still make deterministic decisions about communication, i.e., the communication system must be able to support simultaneous communication of all agents. As this might not be possible, we plan to instead derive a priority measure that is sent to the network manager. That way, the network manager can assign available slots to systems that are in need of communication.

We also presented a framework to learn dynamics models on demand based on a statistical analysis in Chapter 4. Currently, old models are overwritten whenever a new model is learned. However, in reality, dynamics might change due to repeating reasons, such as different loads of a heavy-duty vehicle or changing road conditions. Thus, in future work we want to store old models to be able to reuse them in case they become relevant again. This might be possible by relating models to available

information about the environment. Building knowledge about causal relationships between changes in the environment and changes in the dynamics model is then a key challenge.

We discussed the constrained embedded devices as one limitation of wireless CPSs. As learning is computationally expensive, the integration of such concepts with the wireless control system is challenging. We therefore seek to further exploit the communication capabilities of wireless CPSs through outsourcing heavy computations to cloud services. If the cloud is accessible by all agents, they can also use, for instance, models learned by agents with similar dynamics. This further fosters the collaboration aspect of CPSs. By outsourcing heavy computations to cloud services, we plan to integrate the model-learning framework with the wireless control system. The learned models can then be used to make better informed decisions about future communication demands.

End-to-end learning of communication and control strategy represents an alternative to model-based designs. Building on the methods shown in Chapter 5, we plan to investigate other hierarchical reinforcement learning [157] and temporal abstraction methods such as the option framework [166, 167]. Temporal abstraction naturally also opens the possibility to make statements about future communication needs. This will then again lead to interesting comparisons between model-based strategies as in Chapter 3 and model-free reinforcement learning techniques. How these methods can be used in setups with resource-constrained embedded devices and what kind of guarantees are possible is another open problem for future research.

Appendix A

Control Details of Chapter 2

Here, we present details of the control design from Chapter 2. In particular, we present the proof of Theorem 2.1, a stability analysis including noise, implementation details of the controllers we used for the stabilization experiments, and outline the approach to multi-agent synchronization.

A.1 Proof of Theorem 2.1

For clarity, we reintroduce time index k for θ and ϕ here. Following a similar approach as in [168], 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 $\mathrm{Var}[\delta_\theta(k)] = \sigma_{p_1}^2$, and similarly for $\delta_\phi(k)$. Employing this transformation, $\tilde{A}(k)$ in (2.8) 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 2.1. Thus, all properties of (2.6) are satisfied, and Lemma 2.1 yields the result.

A.2 Stability Analysis with Noise

Dropping the assumption of $v(k) = w(k) = 0$, the system description (2.6) becomes

$$z(k+1) = \tilde{A}(k)z(k) + \tilde{E}\epsilon(k). \tag{A.1}$$

For a system that is constantly perturbed by Gaussian noise the state correlation will not vanish, but we can guarantee it to be bounded. We thus define MSS for systems as (A.1) (cf. [169, 170]).

Definition A.1. Let $M(k) := \mathbb{E}[z(k)z^{\mathrm{T}}(k)]$ denote the state correlation matrix. The system (A.1) is *mean square stable (MSS)* if $\lim_{k\to\infty} M(k) < \infty$ for any initial $z(0)$.

Lemma A.1 ([127])**.** *The state correlation matrix for system* (A.1) *satisfies the difference equation*

$$M(k+1) = \tilde{A}_0 M(k) \tilde{A}_0^{\mathrm{T}} + \tilde{E}_0 W \tilde{E}_0^{\mathrm{T}} + \sum_{i=1}^{L} \sigma_i^2 \left(\tilde{A}_i M(k) \tilde{A}_i^{\mathrm{T}} + \tilde{E}_i W \tilde{E}_i^{\mathrm{T}} \right). \tag{A.2}$$

The noise covariance W and the matrices $\tilde{E}_0$ and $\tilde{E}_i$ are constant, thus to simplify notation we can write $\tilde{E}_0 W \tilde{E}_0^{\mathrm{T}} + \sum_{i=1}^{L} \sigma_i^2 \tilde{E}_i W \tilde{E}_i^{\mathrm{T}} = \hat{W}$. We further define:

Definition A.2. The difference equation (A.2) can be written as

$$M(k+1) = f\left(M(k)\right) + \hat{W}, \tag{A.3}$$

with

$$f(X) = \tilde{A}_0 X \tilde{A}_0^{\mathrm{T}} + \sum_{i=1}^{L} \sigma_i^2 \tilde{A}_i X \tilde{A}_i^{\mathrm{T}}. \tag{A.4}$$

We will now show that mean-square stability of system (2.6) implies mean-square stability of system (A.1).

Lemma A.2. *Mean-square stability of system* (2.6) *implies mean-square stability of* (A.1)*.*

Proof. We first write (A.3) in explicit form,

$$M(k) = f^k\left(M(k)\right) + \sum_{i=0}^{k-1} f^i\left(\hat{W}\right), \tag{A.5}$$

where f^k denotes the repeated composition of f with itself. Now taking the limit yields

$$\begin{aligned}
\lim_{k\to\infty} M(k) &= \underbrace{\lim_{k\to\infty} f^k\left(M(k)\right)}_{=0, \text{ as Lemma 2.1 is fulfilled}} + \lim_{k\to\infty} \sum_{i=0}^{k-1} f^i\left(\hat{W}\right) \\
&= \lim_{k\to\infty} \sum_{i=0}^{k-1} f^i\left(\hat{W}\right) \\
&= \lim_{k\to\infty} \sum_{i=0}^{k-1} f^i\left(\bar{W} - f\left(\bar{W}\right)\right) \\
&= \lim_{k\to\infty} \sum_{i=0}^{k-1} f^i\left(\bar{W}\right) - \lim_{k\to\infty} \sum_{i=1}^{k} f^i\left(\bar{W}\right) \\
&= f^0\left(\bar{W}\right) - \lim_{k\to\infty} f^k\left(\bar{W}\right) \\
&= \bar{W},
\end{aligned} \tag{A.6}$$

with $\bar{W}$ the unique solution to

$$\hat{W} = \bar{W} - f\left(\bar{W}\right). \tag{A.7}$$

□

That is, MSS of the noise-free system implies MSS of the system perturbed by Gaussian noise.

To show that Lemma A.2 can be applied to our system, we rewrite (2.8) to include noise

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

Theorem A.1. *The system* (A.8) *is MSS if, and only if, there exists a* $P > 0$ *such that* (2.7) *holds with* $\tilde{A}_0$, $\tilde{A}_1$, $\tilde{A}_2$, $\sigma_{p_1}^2$, *and* $\sigma_{p_2}^2$ *as in Theorem 2.1 and*

$$\tilde{E}_0 = \begin{pmatrix} 1 & 0 \\ 0 & \mu_\theta \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \tilde{E}_1 = \begin{pmatrix} 0 & 0 \\ 0 & -\mu_\theta \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \tilde{E}_2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Proof. We employ the same transformation for θ and ϕ as in Theorem 2.1, what in the same way fulfills the requirements on $p_i(k)$. $v(k)$ and $w(k)$ are i.i.d., zero-mean Gaussian random variables with finite variance (Σ_{proc} respectively Σ_{meas}), what fulfills the requirements on $\epsilon(k)$. Thus, Lemma A.2 yields the stability result. □

A.3 Stabilizing Controllers

For the stability experiments of Section 2.5.2, we employ the design outlined in Section 2.4.2. 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 [138]. The nominal controller is designed for an update interval $T_U = 40\,\text{ms}$ via pole placement, and we chose 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 adjusted 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 2.5, estimation and filtering occurred at intervals between 10 ms and 20 ms, depending on the experiment.

A.4 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 2.5.3.

We consider the architecture in Figure 2.5, 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 linear quadratic optimal control (LQR) [3] to design the synchronizing controllers. We choose the quadratic cost function

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

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 [3]. The optimal stabilizing controller that minimizes (A.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 collocated 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 states of other agents are received. While the analysis in Section 2.4.3 can be generalized to the synchronization setting, a formal stability proof is beyond the scope of this chapter. In general, stability is less critical here because of shorter update intervals in the local feedback loop.

For the synchronization experiments in Section 2.5.3, we chose Q_i in (A.9) for all pendulums as suggested by the manufacturer in [138] 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 zero.

Appendix B

Proofs of Chapter 3

B.1 Proof of Lemma 3.1

Because $\check{x}(k) = \hat{x}(k)$ for $\gamma(k) = 1$ from (3.14), the remote error $e(k)$ is identical to the KF error $\hat{e}(k) = x(k) - \hat{x}(k)$. From KF theory [136, p. 41], it is known that the conditional and unconditional error distributions are identical, namely

$$f(\hat{e}(k)) = f(\hat{e}(k)|\mathcal{Y}(k),\mathcal{U}(k)) = \mathcal{N}(\hat{e}(k); 0, P(k)). \tag{B.1}$$

That is, the error distribution is independent of any measurement data. Therefore, we also have $f(e(k+M)|\mathcal{Y}(k),\mathcal{U}(k)) = f(\hat{e}(k+M)|\mathcal{Y}(k),\mathcal{U}(k)) = f(\hat{e}(k+M))$ (see [171, Proof of Lem. 2] for a formal argument), from which the claim follows with (B.1).

B.2 Proof of Lemma 3.2

We first establish, for any $M \geq 0$,

$$\begin{aligned}\hat{x}(k+M) = \bar{A}^M \hat{x}(k) &+ \sum_{m=1}^{M} \bar{A}^{M-m} B\xi(k+m-1) \\ &+ \sum_{m=1}^{M} \bar{A}^{M-m} L(k+m) z(k+m) \end{aligned} \tag{B.2}$$

$$\begin{aligned}\hat{x}(k+M|k) = \bar{A}^M \hat{x}(k) &+ \sum_{m=1}^{M} \bar{A}^{M-m} B\xi(k+m-1) \\ &+ \sum_{m=1}^{M-1} G(M-m-1) L(k+m) z(k+m) \end{aligned} \tag{B.3}$$

with $z(k) := y(k) - C\hat{x}(k|k-1)$ the KF innovation, $L(k)$ the KF gain, and $G(m)$ as in (3.29), through proof by induction. For $M = 0$, (B.2) and (B.3) hold trivially with

$\hat{x}(k) = \hat{x}(k)$ and $\hat{x}(k|k) = \hat{x}(k)$, respectively. Induction assumption (IA): assume (B.2) and (B.3) hold for M. Show they are then also true for $M+1$. We have from the KF iterations:

$$\begin{aligned}
\hat{x}(k+M+1) &= A\hat{x}(k+M) + Bu(k+M) + L(k+M+1)z(k+M+1) \\
&= \bar{A}\hat{x}(k+M) + B\xi(k+M) + L(k+M+1)z(k+M+1) \\
&\quad \text{(by (3.11))} \\
&= \bar{A}^{M+1}\hat{x}(k) + \sum_{m=1}^{M+1} \bar{A}^{M+1-m} B\xi(k+m-1) \\
&\quad + \sum_{m=1}^{M+1} \bar{A}^{M+1-m} L(k+m)z(k+m) \quad \text{(from IA (B.2))}
\end{aligned}$$

and

$$\begin{aligned}
\hat{x}(k+M+1|k) &= A\hat{x}(k+M|k) + Bu(k+M) \\
&= A\hat{x}(k+M|k) + BF\hat{x}(k+M) + B\xi(k+M) \\
&= (A+BF)\Big(\bar{A}^{M}\hat{x}(k) + \sum_{m=1}^{M} \bar{A}^{M-m} B\xi(k+m-1)\Big) + B\xi(k+M) \\
&\quad + A\Big(\sum_{m=1}^{M-1} G(M-m-1)L(k+M)z(k+M)\Big) \\
&\quad + BF\Big(\sum_{m=1}^{M} \bar{A}^{M-m} L(k+M)z(k+M)\Big) \quad \text{(from IA (B.2), (B.3))} \\
&= \bar{A}^{M+1}\hat{x}(k) + \sum_{m=1}^{M+1} \bar{A}^{M+1-m} B\xi(k+m-1) \\
&\quad + \sum_{m=1}^{M} G(M-m)L(k+M)z(k+M) \quad \text{(by def. of } G(m)).
\end{aligned}$$

Hence, (B.2) and (B.3) are true for $M+1$, which completes the induction.

Next, we analyze the error $e(k+M)$ for the case $\gamma(k+M) = 0$ (no communication). To ease the presentation, we introduce the auxiliary variable $e^{\mathrm{nc}}(k) := e(k)|_{\gamma(k)=0}$.

Case (i): First, we note that $k > \kappa(k-1)$ implies $\kappa(k-1) = \ell(k)$ because $\kappa(k-1)$, the last nonzero element of $\Gamma(k+m-1)$, is in the past, and the identity thus follows from the definition of $\ell(k)$. It follows further that all triggering decisions following $\gamma(\ell) = 1$ are 0 until $\gamma(k+m-1)$ (otherwise $\gamma(\ell)$ would not be the last element in $\Gamma(k+m-1)$). Hence, we have the communication pattern $\gamma(\ell) = 1$ and $\gamma(\ell+1) = \gamma(\ell+2) = \cdots = \gamma(k+m-1) = 0$.

Let $\tilde{\Delta} := M + k - \ell$. From

$$e^{\mathrm{nc}}(k+M) = x(k+M) - \bar{A}^{\tilde{\Delta}}\hat{x}(\ell) - \sum_{m=1}^{\tilde{\Delta}} \bar{A}^{\tilde{\Delta}-m} B\xi(\ell+m-1)$$

it follows that the conditional distribution (3.24) is Gaussian. It thus suffices to consider mean and variance in the following.

For the conditional mean, we have

$$\begin{aligned}
&\mathbb{E}[e^{\mathrm{nc}}(k+M)|\mathcal{Y}(k),\mathcal{U}(k)] \\
&= \mathbb{E}[x(k+M)|\mathcal{Y}(k),\mathcal{U}(k)] - \bar{A}^{\tilde{\Delta}}\hat{x}(\ell) - \sum_{m=1}^{\tilde{\Delta}} \bar{A}^{\tilde{\Delta}-m} B\xi(\ell+m-1), \qquad \text{(B.4)}
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}[x(k+M)|\mathcal{Y}(k),\mathcal{U}(k)] &= \mathbb{E}\left[\mathbb{E}[x(k+M)|\mathcal{Y}(k),\mathcal{U}(k+M)]\big|\mathcal{Y}(k),\mathcal{U}(k)\right] \\
&= \mathbb{E}[\hat{x}(k+M|k)|\mathcal{Y}(k),\mathcal{U}(k)] \\
&= \bar{A}^M\hat{x}(k) + \sum_{m=1}^{M} \bar{A}^{M-m} B\xi(k+m-1) \qquad \text{(B.5)}
\end{aligned}$$

where we used the tower property of conditional expectation, (3.8), and (B.3) with the fact that the KF innovation sequence $z(k)$ is zero-mean and uncorrelated. Using (B.5) with (B.4), we obtain

$$\begin{aligned}
\mathbb{E}[e^{\mathrm{nc}}(k+M)|\mathcal{Y}(k),\mathcal{U}(k)] &= \bar{A}^M(\hat{x}(k) - \bar{A}^{k-\ell}\hat{x}(\ell)) + \sum_{m=1}^{M} \bar{A}^{M-m} B\xi(k+m-1) \\
&\quad - \sum_{m=1}^{k-\ell} \bar{A}^{\tilde{\Delta}-m} B\xi(\ell+m-1) - \sum_{m=k-\ell+1}^{M+k-\ell} \bar{A}^{M+k-\ell-m} B\xi(\ell+m-1) \qquad \text{(B.6)} \\
&= \bar{A}^M\Big(\hat{x}(k) - \bar{A}^{k-\ell}\hat{x}(\ell) - \sum_{m=1}^{k-\ell} \bar{A}^{k-\ell-m} B\xi(\ell+m-1)\Big) \qquad \text{(B.7)}
\end{aligned}$$

which proves (3.25). The first and third sum in (B.6) can be seen to be identical by substituting m with $m + k - \ell$.

Employing the tower property for the conditional variance, we get

$$\begin{aligned}
&\mathrm{Var}[e^{\mathrm{nc}}(k+M)|\mathcal{Y}(k),\mathcal{U}(k)] \\
&\quad = \mathbb{E}\left[\mathrm{Var}[e^{\mathrm{nc}}(k+M)|\mathcal{Y}(k),\mathcal{U}(k+M)]\big|\mathcal{Y}(k),\mathcal{U}(k)\right] \\
&\qquad + \mathrm{Var}\left[\mathbb{E}[e^{\mathrm{nc}}(k+M)|\mathcal{Y}(k),\mathcal{U}(k+M)]\big|\mathcal{Y}(k),\mathcal{U}(k)\right] \\
&\quad = \mathbb{E}[P(k+M|k)|\mathcal{Y}(k),\mathcal{U}(k)] + \mathrm{Var}[\hat{x}(k+M|k)|\mathcal{Y}(k),\mathcal{U}(k)] \\
&\quad = P(k+M|k) + \mathrm{Var}[\hat{x}(k+M|k)|\mathcal{Y}(k),\mathcal{U}(k)].
\end{aligned}$$

Furthermore, $\mathrm{Var}[\hat{x}(k+M|k)|\mathcal{Y}(k),\mathcal{U}(k)] = \Xi(k,M)$ follows from (B.3), $z(k)$ being uncorrelated, and

$$\begin{aligned}&\mathrm{Var}[z(k+M)|\mathcal{Y}(k),\mathcal{U}(k)]\\&= \mathrm{Var}[CA\hat{e}(k+m-1) + Cv(k+m-1) + w(k+M)|\mathcal{Y}(k),\mathcal{U}(k)]\\&= \tilde{P}(k+M)\end{aligned}$$

as defined in (3.28). This completes the proof for *Case (i)*.

Case (ii): We use $\kappa = \kappa(k-1)$ to simplify notation. By definition of κ, we have $\kappa \leq M+k-1$, and hence $k \leq \kappa \leq M+k-1$. That is, a triggering will happen now or before the end of the horizon $M+k$. At the triggering instant κ, we have from (3.14), $e(\kappa) = x(\kappa) - \hat{x}(\kappa)$. Hence, the distribution of the error at time κ is known irrespective of past and future data. Following the same arguments as in the proof of Lemma 3.1, we have $f(e(\kappa)|\mathcal{Y}(k),\mathcal{U}(k)) = f(e(\kappa)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)) = \mathcal{N}(e(\kappa);\, 0, P(\kappa))$.

From the definition of κ, we know that there is no further communication happening until $M+k-1$. Thus, we can iterate (3.14) with $\gamma = 0$. Using the same reasoning as in *Case (i)*, we have

$$e^{\mathrm{nc}}(k+M) = e^{\mathrm{nc}}(\kappa+\Delta) = x(\kappa+\Delta) - \bar{A}^{\Delta}\hat{x}(\kappa) - \sum_{m=1}^{\Delta} \bar{A}^{\Delta-m} B\xi(\kappa+m-1)$$

and thus

$$\begin{aligned}&\mathbb{E}[e^{\mathrm{nc}}(\kappa+\Delta)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)]\\&= \mathbb{E}[x(\kappa+\Delta)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)] - \bar{A}^{\Delta}\hat{x}(\kappa) - \sum_{m=1}^{\Delta} \bar{A}^{\Delta-m} B\xi(\kappa+m-1)\\&= \mathbb{E}[\hat{x}(\kappa+\Delta|\kappa)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)] - \bar{A}^{\Delta}\hat{x}(\kappa) - \sum_{m=1}^{\Delta} \bar{A}^{\Delta-m} B\xi(\kappa+m-1) = 0\end{aligned}$$

where the last equality follows from (B.3) and $z(k)$ being zero-mean. Similarly, for the variance, we obtain

$$\begin{aligned}\mathrm{Var}[e^{\mathrm{nc}}(\kappa+\Delta)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)] &= \mathbb{E}[P(\kappa+\Delta|\kappa)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)]\\&\quad + \mathrm{Var}[\hat{x}(\kappa+\Delta|\kappa)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)]\\&= P(\kappa+\Delta|\kappa) + \mathrm{Var}[\hat{x}(\kappa+\Delta|\kappa)|\mathcal{Y}(\kappa),\mathcal{U}(\kappa)]\\&= P(\kappa+\Delta|\kappa) + \Xi(\kappa,\Delta).\end{aligned}$$

Bibliography

[1] Bart Besselink, Valerio Turri, Sebastian H. Van De Hoef, Kuo Yun Liang, Assad Alam, Jonas Martensson, and Karl H. Johansson. Cyber-physical control of road freight transport. *Proceedings of the IEEE*, 104(5), 2016.

[2] David B Ender. Process control performance: Not as good as you think. *Control Engineering*, 40(10):180–190, 1993.

[3] Brian D. O. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, 2007.

[4] Xian-Ming Zhang, Qing-Long Han, and Xinghuo Yu. Survey on recent advances in networked control systems. *IEEE Transactions on Industrial Informatics*, 12(5):1740–1752, 2016.

[5] Kyoung-Dae Kim and Panganamala R Kumar. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 100(Special Centennial Issue):1287–1308, 2012.

[6] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Design Automation Conference (DAC), 47th ACM/IEEE*, pages 731–736, 2010.

[7] Jianhua Shi, Jiafu Wan, Hehua Yan, and Hui Suo. A survey of cyber-physical systems. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pages 1–6, 2011.

[8] Ning Lu, Nan Cheng, Ning Zhang, Xuemin Shen, and Jon W Mark. Connected vehicles: Solutions and challenges. *IEEE Internet of Things Journal*, 1(4):289–299, 2014.

[9] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015.

[10] László Monostori. Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia Cirp*, 17:9–13, 2014.

[11] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, Baek-Gyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alexander Roederer, et al. Challenges and research directions in medical cyber–physical systems. *Proceedings of the IEEE*, 100(1):75–90, 2012.

[12] Yin Zhang, Meikang Qiu, Chun-Wei Tsai, Mohammad Mehedi Hassan, and Atif Alamri. Health-CPS: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95, 2017.

[13] Joo P Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1), 2007.

[14] Lixian Zhang, Huijun Gao, and Okyay Kaynak. Network-induced constraints in networked control systems - a survey. *IEEE Transactions on Industrial Informatics*, 9(1), 2013.

[15] Rogelio Luck and Asok Ray. An observer-based compensator for distributed delays. *Automatica*, 26(5), 1990.

[16] Bruno Sinopoli, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, Michael I. Jordan, and Shankar S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9), 2004.

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

[18] Junlin Xiong and James Lam. Stabilization of linear systems over networks with bounded packet loss. *Automatica*, 43(1), 2007.

[19] Gregory C. Walsh, Hong Ye, and Linda G. Bushnell. Stability analysis of networked control systems. *IEEE Transactions on Control Systems Technology*, 10(3), 2002.

[20] Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzén. How does control timing affect performance? Analysis and simulation of timing using jitterbug and truetime. *IEEE Control Systems Magazine*, 2003.

[21] Nicolas W. Bauer, S. J.L.M.Bas Van Loon, Nathan Van De Wouw, and W. P.M.H.Maurice Heemels. 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.

[22] Anton Cervin. *Integrated Control and Real-Time Scheduling.* PhD thesis, Department of Automatic Control, Lund Institute of Technology (LTH), 2003.

[23] Goran Frehse, Arne Hamann, Sophie Quinton, and Matthias Woehrle. Formal analysis of timing effects on closed-loop properties of control software. In *IEEE Real-Time Systems Symposium (RTSS)*, 2014.

[24] John A. Stankovic, Tarek F. Abdelzaher, Chenyang Lu, Lui Sha, and Jennifer C. Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7), 2003.

[25] Chenyang Lu, Brian M. Blum, Tarek F. Abdelzaher, John A. Stankovic, and Tian He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2002.

[26] Tian He, John A. Stankovic, Chenyang Lu, and Tarek Abdelzaher. SPEED: a stateless protocol for real-time communication in sensor networks. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2003.

[27] Octav Chipara, Chenyang Lu, and Gruia-Catalin Roman. Real-time query scheduling for wireless sensor networks. In *IEEE International Real-Time Systems Symposium (RTSS)*, 2007.

[28] Octav Chipara, Chengjie Wu, Chenyang Lu, and William Griswold. Interference-aware real-time flow scheduling for wireless sensor networks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2011.

[29] Tony O'donovan, Wolf-Bastian Pöttner, Utz Roedig, Jorge Sá Silva, Ricardo Silva, Cormac J. Sreenan, Vasos Vassiliou, Thiemo Voigt, Lars Wolf, Zinon Zinonos, James Brown, Felix Büsching, Alberto Cardoso, José Cecílio, Jose Do Ó, Pedro Furtado, Paulo Gil, and Anja Jugel. The GINSENG system for wireless monitoring and control. *ACM Transactions on Sensor Networks*, 10(1), 2013.

[30] S. M. Shahriar Nirjon, John A. Stankovic, and Kamin Whitehouse. IAA: Interference aware anticipatory algorithm for scheduling and routing periodic real-time streams in wireless sensor networks. In *IEEE International Conference on Networked Sensing Systems (INSS)*, 2010.

[31] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. Real-time scheduling for WirelessHART networks. In *IEEE Real-Time Systems Symposium (RTSS)*, 2010.

[32] Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen. End-to-end delay analysis for fixed priority scheduling in WirelessHART networks. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2011.

[33] Haibo Zhang, Pablo Soldati, and Mikael Johansson. Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks. In *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2009.

[34] Marco Zimmerling, Luca Mottola, Pratyush Kumar, Federico Ferrari, and Lothar Thiele. Adaptive real-time communication for wireless cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 1(2), 2017.

[35] Romain Jacob, Marco Zimmerling, Pengcheng Huang, Jan Beutel, and Lothar Thiele. End-to-end real-time guarantees in wireless cyber-physical systems. In *IEEE Real-Time Systems Symposium (RTSS)*, 2016.

[36] Johan Akerberg, Mikael Gidlund, and Mats Bjorkman. Future research challenges in wireless sensor and actuator networks targeting industrial automation. In *IEEE International Conference on Industrial Informatics (INDIN)*, 2011.

[37] Bo Li, Yehan Ma, Tyler Westenbroek, Chengjie Wu, Humberto Gonzalez, and Chenyang Lu. Wireless routing and control: A cyber-physical case study. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2016.

[38] Yehan Ma, Dolvara Gunatilaka, Bo Li, Humberto Gonzalez, and Chenyang Lu. Holistic cyber-physical management for dependable wireless control systems. *ACM Transactions on Cyber-Physical Systems*, 3(1):3, 2018.

[39] Matteo Ceriotti, Michele Corra, Leandro D'Orazio, Roberto Doriguzzi, Daniele Facchin, Sc Tefan Guna, Gian Paolo Jesi, Renato Lo Cigno, Luca Mottola, Amy L. Murphy, Massimo Pescalli, Gian Pietro Picco, Denis Pregnolato, and Carloalberto Torghele. Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2011.

[40] Abusayeed Saifullah, Sriram Sankar, Jie Liu, Chenyang Lu, Ranveer Chandra, and Bodhi Priyantha. Capnet: A real-time wireless management network for data center power capping. In *IEEE Real-Time Systems Symposium (RTSS)*, 2014.

[41] Nicholas J. Ploplys, Paul A. Kawka, and Andrew G. Alleyne. Closed-loop control over wireless networks. *IEEE Control Systems Magazine*, 24(3), 2004.

[42] Hong Ye, Gregory C. Walsh, and Linda G. Bushnell. Real-time mixed-traffic wireless networks. *IEEE Transactions on Industrial Electronics*, 48(5), 2001.

[43] Johan Eker, Anton Cervin, and Andreas Hörjel. Distributed wireless control using bluetooth. In *IFAC Conference on New Technologies for Computer Control*, 2001.

[44] J. P. Lynch, Y. Wang, R. A. Swartz, K. C. Lu, and C. H. Loh. Implementation of a closed-loop structural control system using wireless sensor networks. *Structural Control and Health Monitoring*, 2007.

[45] Michael Lemmon. Event-triggered feedback in control, estimation, and optimization. In Alberto Bemporad, Maurice Heemels, and Mikael Johansson, editors, *Networked Control Systems*, volume 406 of *Lecture Notes in Control and Information Sciences*, pages 293–358. Springer, 2010.

[46] WPMH Heemels, Karl Henrik Johansson, and Paulo Tabuada. An introduction to event-triggered and self-triggered control. In *Decision and Control (CDC), IEEE 51st Annual Conference on*, pages 3270–3285, 2012.

[47] Jan Lunze and Lars Grüne. Introduction to networked control systems. In *Control Theory of Digitally Networked Dynamic Systems*. Springer, Heidelberg, 2014.

[48] Marek Miskowicz. *Event-Based Control and Signal Processing*. CRC Press, 2016.

[49] Sebastian Trimpe and Marco Campi. On the choice of the event trigger in event-based estimation. In *International Conference on Event-based Control, Communication, and Signal Processing*, pages 1–8, 2015.

[50] Joris Sijs, Benjamin Noack, Mircea Lazar, and Uwe D Hanebeck. Time-periodic state estimation with event-based measurement updates. In *Event-Based Control and Signal Processing*. CRC Press, 2016.

[51] Dawei Shi, Ling Shi, and Tongwen Chen. *Event-Based State Estimation*. Springer, 2016.

[52] Karl J Åström. Event based control. In *Analysis and design of nonlinear control systems*, pages 127–147. Springer, 2008.

[53] Toivo Henningsson, Erik Johannesson, and Anton Cervin. Sporadic event-based control of first-order linear stochastic systems. *Automatica*, 44(11):2890–2895, 2008.

[54] Xiangyu Meng and Tongwen Chen. Optimal sampling and performance comparison of periodic and event based impulse control. *IEEE Transactions on Automatic Control*, 57(12):3252–3259, 2012.

[55] Anton Cervin and Karl Johan Åström. On limit cycles in event-based control systems. In *Decision and Control (CDC), 46th IEEE Conference on*, pages 3190–3195, 2007.

[56] Karl-Erik Årzén. A simple event-based PID controller. *IFAC Proceedings Volumes*, 32(2):8687–8692, 1999.

[57] José Sánchez, Antonio Visioli, and Sebastián Dormido. Event-based PID control. In *PID Control in the Third Millennium*, pages 495–526. Springer, 2012.

[58] Sylvain Durand and Nicolas Marchand. Further results on event-based PID controller. In *Control Conference (ECC), European*, pages 1979–1984, 2009.

[59] Maben Rabi and Karl H Johansson. Event-triggered strategies for industrial control over wireless networks. In *Proceedings of the 4th annual international conference on wireless internet*, page 34. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[60] Sebastian Trimpe and Raffaello D'Andrea. An experimental demonstration of a distributed and event-based state estimation algorithm. In *18th IFAC World Congress*, pages 8811–8818, 2011.

[61] S. Trimpe and R. D'Andrea. Event-based state estimation with variance-based triggering. *IEEE Transaction on Automatic Control*, 59(12):3266–3281, 2014.

[62] Miguel Martínez-Rey, Felipe Espinosa, Alfredo Gardel, and Carlos Santos. On-board event-based state estimation for trajectory approaching and tracking of a vehicle. *Sensors*, 15(6):14569–14590, 2015.

[63] Sebastian Trimpe. Event-based state estimation: An emulation-based approach. *IET Control Theory & Applications*, 11:1684–1693, July 2017.

[64] Michael Muehlebach and Sebastian Trimpe. Distributed event-based state estimation for networked systems: An LMI-approach. *IEEE Transactions on Automatic Control*, 63(1):269–276, January 2018.

[65] Joris Sijs, Benjamin Noack, and Uwe D Hanebeck. Event-based state estimation with negative information. In *16th International Conference on Information Fusion*, pages 2192–2199, 2013.

[66] Dawei Shi, Tongwen Chen, and Ling Shi. An event-triggered approach to state estimation with multiple point- and set-valued measurements. *Automatica*, 50(6):1641–1648, 2014.

[67] Junfeng Wu, Qing-Shan Jia, K.H. Johansson, and Ling Shi. Event-based sensor data scheduling: Trade-off between communication rate and estimation quality. *IEEE Transactions on Automatic Control*, 58(4):1041–1046, 2013.

[68] Alex S. Leong, Subhrakanti Dey, and Daniel E. Quevedo. Sensor scheduling in variance based event triggered estimation with packet drops. *IEEE Transactions on Automatic Control*, 62(4):1880–1895, 2017.

[69] Jan Willem Marck and Joris Sijs. Relevant sampling applied to event-based state-estimation. In *International Conference on Sensor Technologies and Applications*, pages 618–624, July 2010.

[70] Manel Velasco, Josep Fuertes, and Pau Marti. The self triggered task model for real-time control systems. In *Work-in-Progress Session of the 24th IEEE Real-Time Systems Symposium*, 2003.

[71] Xiaofeng Wang and M.D. Lemmon. Self-triggered feedback control systems with finite-gain $\mathcal{L}_2$ stability. *IEEE Transactions on Automatic Control*, 54(3):452–467, 2009.

[72] Manuel Mazo, Adolfo Anta, and Paulo Tabuada. An ISS self-triggered implementation of linear controllers. *Automatica*, 46(8):1310–1314, 2010.

[73] Adolfo Anta and Paulo Tabuada. To sample or not to sample: Self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030–2042, 2010.

[74] Nacim Meslem and Christophe Prieur. State estimation based on self-triggered measurements. In *19th IFAC World Congress*, pages 86–91, 2014.

[75] Vincent Andrieu, Madiha Nadri, Ulysse Serres, and Jean-Claude Vivalda. Self-triggered continuous-discrete observer with updated sampling period. *Automatica*, 62:106–113, 2015.

[76] Florian David Brunner, TMP Gommans, WPMH Heemels, and Frank Allgöwer. Resource-aware set-valued estimation for discrete-time linear systems. In *Decision and Control (CDC), IEEE 54th Annual Conference on*, pages 5480–5486, 2015.

[77] Markus Kögel and Rolf Findeisen. Robust output feedback predictive control with self-triggered measurements. In *Decision and Control (CDC), IEEE 54th Annual Conference on*, pages 5487–5493, 2015.

[78] João Almeida, Carlos Silvestre, and Antonio M Pascoal. Observer based self-triggered control of linear plants with unknown disturbances. In *American Control Conference (ACC)*, pages 5688–5693, 2012.

[79] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[80] Stefan Schaal and Christopher G Atkeson. Learning control in robotics. *IEEE Robotics & Automation Magazine*, 17(2):20–29, 2010.

[81] Alonso Marco, Philipp Hennig, Stefan Schaal, and Sebastian Trimpe. On the design of LQR kernels for efficient controller learning. In *Decision and Control (CDC), IEEE 56th Annual Conference on*, pages 5193–5200, 2017.

[82] Andreas Doerr, Duy Nguyen-Tuong, Alonso Marco, Stefan Schaal, and Sebastian Trimpe. Model-based policy search for automatic tuning of multivariate PID controllers. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 5295–5301, 2017.

[83] Alon Ascoli, Dominik Baumann, Ronald Tetzlaff, Leon O Chua, and Manfred Hild. Memristor-enhanced humanoid robot control system–Part I: Theory behind the novel memcomputing paradigm. *International Journal of Circuit Theory and Applications*, 46(1):155–183, 2018.

[84] Dominik Baumann, Alon Ascoli, Ronald Tetzlaff, Leon O. Chua, and Manfred Hild. Memristor-enhanced humanoid robot control system–Part II: Circuit theoretic model and performance analysis. *International Journal of Circuit Theory and Applications*, 46(1):184–220, 2018.

[85] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[86] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[87] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016.

[88] Pramod P Khargonekar and Munther A Dahleh. Advancing systems and control research in the era of ML and AI. *Annual Reviews in Control*, 2018.

[89] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[90] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[91] Kyriakos G. Vamvoudakis and Henrique Ferraz. Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. *Automatica*, 87:412 – 420, 2018.

[92] Xiangnan Zhong, Zhen Ni, Haibo He, Xin Xu, and Dongbin Zhao. Event-triggered reinforcement learning approach for unknown nonlinear continuous-time system. In *Neural Networks (IJCNN), International Joint Conference on*, pages 3677–3684, 2014.

[93] Avimanyu Sahoo, Hao Xu, and Sarangapani Jagannathan. Neural network-based event-triggered state feedback control of nonlinear continuous-time systems. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3):497–509, 2016.

[94] Xiong Yang, Haibo He, and Derong Liu. Event-triggered optimal neuro-controller design with reinforcement learning for unknown nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1–13, 2017.

[95] Vignesh Narayanan and Sarangapani Jagannathan. Event-triggered distributed control of nonlinear interconnected systems using online reinforcement learning with exploration. *IEEE Transactions on Cybernetics*, 2017.

[96] Burak Demirel, Arunselvan Ramaswamy, Daniel E Quevedo, and Holger Karl. Deepcas: A deep reinforcement learning algorithm for control-aware scheduling. *IEEE Control Systems Letters*, pages 737–742, 2018.

[97] Friedrich Solowjow, Dominik Baumann, Jochen Garcke, and Sebastian Trimpe. Event-triggered learning for resource-efficient networked control. In *American Control Conference (ACC)*, 2018.

[98] Patricia Derler, Edward A. Lee, and Alberto Sangiovanni Vincentelli. Modeling cyber-physical systems. *Proceedings of the IEEE*, 100(1), 2012.

[99] Peter Corke, Tim Wark, Raja Jurdak, Wen Hu, Philip Valencia, and Darren Moore. Environmental wireless sensor networks. *Proceedings of the IEEE*, 98(11), 2010.

[100] Nikolaus Correll, Prabal Dutta, Richard Han, and Kristofer Pister. New directions: Wireless robotic materials. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2017.

[101] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys and Tutorials*, 18(4), 2016.

[102] Chenyang Lu, Abusayeed Saifullah, Bo Li, Mo Sha, Humberto Gonzalez, Dolvara Gunatilaka, Chengjie Wu, Lanshun Nie, and Yixin Chen. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proceedings of the IEEE*, 104, 2016.

[103] Thomas Watteyne, Vlado Handziski, Xavier Vilajosana, Simon Duquennoy, Oliver Hahm, Emmanuel Baccelli, and Adam Wolisz. Industrial wireless IP-based cyber-physical systems. *Proceedings of the IEEE*, 104(5), 2016.

[104] Karl Johan Åström and Björn Wittenmark. *Computer-Controlled Systems: Theory and Design*. Prentice Hall, 1996.

[105] James A. Preiss, Wolfgang Honig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[106] Jannik Abbenseth, Felipe Garcia Lopez, Christian Henkel, and Stefan Dörr. Cloud-based cooperative navigation for mobile service robots in dynamic industrial environments. In *Symposium on Applied Computing (SAC)*, 2017.

[107] Karl Johan Åström and Björn Wittenmark. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[108] Aitor Hernandez, Joao Faria, José Araújo, Pangun Park, Henrik Sandberg, and Karl H. Johansson. Inverted pendulum control over an IEEE 802.15.4 wireless sensor and actuator network. In *European Conference on Wireless Sensor Networks (EWSN)*, 2011.

[109] Jose Araujo, Manuel Mazo, Adolfo Anta, Paulo Tabuada, and Karl H. Johansson. System architectures, protocols and algorithms for aperiodic wireless control systems. *IEEE Transactions on Industrial Informatics*, 10(1), 2014.

[110] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[111] Craig B. Schindler, Thomas Watteyne, Xavier Vilajosana, and Kristofer S.J. Pister. Implementation and characterization of a multi-hop 6TiSCH network for experimental feedback control of an inverted pendulum. In *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2017.

[112] Sebastian Trimpe and Raffaello D'Andrea. The balancing cube: A dynamic sculpture as test bed for distributed estimation and control. *IEEE Control Systems Magazine*, 32(6), 2012.

[113] Fredrik Österlind and Adam Dunkels. Approaching the maximum 802.15.4 multi-hop throughput. In *ACM Workshop on Embedded Networked Sensors (HotEmNets)*, 2008.

[114] Björn Wittenmark, Johan Nilsson, and Martin Törngren. Timing problems in real-time control systems. In *American Control Conference (ACC)*, 1995.

[115] TC Yang, H Yu, MR Fei, and LX Li. Networked control systems: A historical review and current research topics. *Measurement & Control*, 38(1), 2005.

[116] Kannan Srinivasan, Maria a Kazandjieva, Saatvik Agarwal, and Philip Levis. The β -factor : Measuring wireless link burstiness. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2008.

[117] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. Orchestra: Robust mesh networks through autonomously scheduled TSCH. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2015.

[118] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. Efficient network flooding and time synchronization with glossy. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2011.

[119] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. Low-power wireless bus. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2012.

[120] Timofei Istomin, Amy L. Murphy, Gian Pietro Picco, and Usman Raza. Data prediction + synchronous transmissions = ultra-low power wireless sensor networks. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2016.

[121] Felix Sutton, Marco Zimmerling, Reto Da Forno, Roman Lim, Tonio Gsell, Georgia Giannopoulou, Federico Ferrari, Jan Beutel, and Lothar Thiele. Bolt: A stateful processor interconnect. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2015.

[122] Romain Jacob, Licong Zhang, Marco Zimmerling, Jan Beutel, Samarjit Chakraborty, and Lothar Thiele. TTW: A time-triggered-wireless design for CPS. In *Design, Automation & Test in Europe (DATE)*, 2018.

[123] Marco Zimmerling, Federico Ferrari, Luca Mottola, and Lothar Thiele. On modeling low-power wireless protocols based on synchronous packet transmissions. In *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2013.

[124] Jens Karschau, Marco Zimmerling, and Benjamin M Friedrich. Renormalization group theory for percolation in time-varying networks. *Scientific reports*, 8(1):8011, 2018.

[125] C Lenzen, P Sommer, and R Wattenhofer. Pulsesync: An efficient and scalable clock synchronization protocol. *IEEE/ACM Transactions on Networking*, 23(3), 2015.

[126] Yehan Ma, Dolvara Gunatilaka, Bo Li, Humberto Gonzalez, and Chenyang Lu. Holistic cyber-physical management for dependable wireless control systems. *ACM Transactions on Cyber-Physical Systems*, 3(1), 2018.

[127] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Vendataramanan Balakrishnan. *Linear Matrix Inequalities in System & Control Theory*. Society for Industrial & Applied Mathematics, 1994.

[128] Y. Labit, D. Peaucelle, and D. Henrion. SEDUMI INTERFACE 1.02: A tool for solving LMI problems with SEDUMI. In *IEEE International Symposium on Computer Aided Control System Design (CACSD)*, 2002.

[129] Jan Lunze. *Feedback control of large scale systems.* Prentice-Hall, 1992.

[130] Pulkit Grover. Information structures, the Witsenhausen counterexample, and communicating using actions. *Encyclopedia of Systems and Control*, 2014.

[131] Jan Lunze. Synchronization of heterogeneous agents. *IEEE Transactions on Automatic Control*, 57(11), 2012.

[132] Dominik Baumann, Fabian Mager, Harsoveet Singh, Marco Zimmerling, and Sebastian Trimpe. Evaluating low-power wireless cyber-physical systems. In *IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, pages 13–18, 2018.

[133] Olfa Boubaker. The inverted pendulum: A fundamental benchmark in control theory and robotics. In *International Conference on Education and e-Learning Innovations (ICEELI)*, 2012.

[134] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. An empirical study of low-power wireless. *ACM Transactions on Sensor Networks*, 6(2), 2010.

[135] Karl Johan Åström and Bo Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. In *Proceedings of the 14th IFAC World Congress*, pages 301–306, 1999.

[136] Brian DO Anderson and John B Moore. *Optimal Filtering.* Dover Publications, Mineola, New York, 2005.

[137] Jean-Pierre Thomesse. Fieldbus technology in industrial automation. *Proceedings of the IEEE*, 93(6), 2005.

[138] Quanser Inc. IP02 - self-erecting single inverted pendulum (SESIP) - linear experiment #6: PV and LQR control - instructor manual, 2012.

[139] Assad Alam, Jonas Mårtensson, and Karl H. Johansson. Experimental evaluation of decentralized cooperative cruise control for heavy-duty vehicle platooning. *Control Engineering Practice*, 38:11 – 25, 2015.

[140] W Levine and Michael Athans. On the optimal error regulation of a string of moving vehicles. *IEEE Transactions on Automatic Control*, 11(3):355–361, Jul 1966.

[141] Gerrit JL Naus, Rene PA Vugts, Jeroen Ploeg, Marinus JG van de Molengraft, and Maarten Steinbuch. String-stable CACC design and experimental validation: A frequency-domain approach. *IEEE Transactions on Vehicular Technology*, 59(9):4268–4279, 2010.

[142] Ulrike Von Luxburg and Bernhard Schölkopf. Statistical learning theory: Models, concepts, and results. In *Handbook of the History of Logic*, volume 10, pages 651–706. Elsevier, 2011.

[143] Richard Bellman, Irving Glicksberg, and Oliver Gross. On the "bang-bang" control problem. *Quarterly of Applied Mathematics*, 14(1):11–18, 1956.

[144] Karl Johan Åström and Tore Hägglund. *PID controllers: Theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.

[145] Frank J Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[146] Chithrupa Ramesh, Henrik Sandberg, Lei Bao, and Karl Henrik Johansson. On the dual effect in state-based scheduling of networked control systems. In *American Control Conference (ACC)*, pages 2216–2221, 2011.

[147] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[148] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[149] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[150] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[151] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[152] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, Lille, France, Jul 2015.

[153] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

[154] Roland Hafner and Martin Riedmiller. Reinforcement learning in feedback control. *Machine Learning*, 84(1):137–169, Jul 2011.

[155] Warwick Masson, Pravesh Ranchod, and George Konidaris. Reinforcement learning with parameterized actions. In *AAAI Conference on Artificial Intelligence*, pages 1934–1940, 2016.

[156] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. `https://github.com/openai/baselines`, 2017.

[157] Richard S Sutton. TD models: Modeling the world at a mixture of time scales. In *Machine Learning Proceedings 1995*, pages 531–539. Elsevier, 1995.

[158] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[159] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, volume 2, pages 267–274, 2002.

[160] Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960.

[161] Dimitri P Bertsekas. Stable optimal control and semicontractive dynamic programming. *SIAM Journal on Control and Optimization*, 56(1):231–252, 2018.

[162] MCF Donkers and WPMH Heemels. Output-based event-triggered control with guaranteed $\mathcal{L}_\infty$-gain and improved and decentralized event-triggering. *IEEE Transactions on Automatic Control*, 57(6):1362–1376, 2012.

[163] Paulo Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[164] WPM Heemels, MCF Donkers, and Andrew R Teel. Periodic event-triggered control for linear systems. *IEEE Transactions on Automatic Control*, 58(4):847–861, 2013.

[165] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 5026–5033, 2012.

[166] Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2000.

[167] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, 2017.

[168] Matt Rich and Nicola Elia. Optimal mean-square performance for MIMO networked systems. In *American Control Conference (ACC)*, 2015.

[169] James S Freudenberg, Richard H Middleton, and Victor Solo. Stabilization and disturbance attenuation over a gaussian communication channel. *IEEE Transactions on Automatic Control*, 55(3):795–799, 2010.

[170] Ali A Zaidi, Tobias J Oechtering, Serdar Yüksel, and Mikael Skoglund. Stabilization of linear systems over gaussian networks. *IEEE Transactions on Automatic Control*, 59(9):2369–2384, 2014.

[171] Sebastian Trimpe. Predictive and self triggering for event-based state estimation. In *Decision and Control (CDC), IEEE 55th Conference on*, pages 3098–3105, 2016.