Technische Universität Dresden

Doctoral Thesis

Design and Real-World Evaluation of Dependable Wireless Cyber-Physical Systems

Author:
Fabian Mager

Advisor:
Prof. Dr. Marco Zimmerling

A thesis submitted in fulfillment of the requirements for the degree of Doktoringenieur (Dr.-Ing.)

in the

Networked Embedded Systems Lab
Fakultät Informatik

December 22, 2022

To my familiy.

Abstract

The ongoing effort for an efficient, sustainable, and automated interaction between humans, machines, and our environment will make cyber-physical systems (CPS) an integral part of the industry and our daily lives. At their core, CPS integrate computing elements, communication networks, and physical processes that are monitored and controlled through sensors and actuators. New and innovative applications become possible by extending or replacing static and expensive cable-based communication infrastructures with wireless technology. The flexibility of wireless CPS is a key enabler for many envisioned scenarios, such as intelligent factories, smart farming, personalized healthcare systems, autonomous search and rescue, and smart cities.

High dependability, efficiency, and adaptivity requirements complement the demand for wireless and low-cost solutions in such applications. For instance, industrial and medical systems should work reliably and predictably with performance guarantees, even if parts of the system fail. Because emerging CPS will feature mobile and battery-driven devices that can execute various tasks, the systems must also quickly adapt to frequently changing conditions. Moreover, as applications become ever more sophisticated, featuring compact embedded devices that are deployed densely and at scale, efficient designs are indispensable to achieve desired operational lifetimes and satisfy high bandwidth demands.

Meeting these partly conflicting requirements, however, is challenging due to imperfections of wireless communication and resource constraints along several dimensions, for example, computing, memory, and power constraints of the devices. More precisely, frequent and correlated message losses paired with very limited bandwidth and varying delays for the message exchange significantly complicate the control design. In addition, since communication ranges are limited, messages must be relayed over multiple hops to cover larger distances, such as an entire factory. Although the resulting mesh networks are more robust against interference, efficient communication is a major challenge as wireless imperfections get amplified, and significant coordination effort is needed, especially if the networks are dynamic.

CPS combine various research disciplines, which are often investigated in isolation, ignoring their complex interaction. However, to address this interaction and build trust in the proposed solutions, evaluating CPS using real physical systems and wireless networks paired with formal guarantees of a system's end-to-end behavior is necessary. Existing works that take this step can only satisfy a few of the abovementioned requirements. Most notably, multi-hop communication has only been used to control

slow physical processes while providing no guarantees. One of the reasons is that the current communication protocols are not suited for dynamic multi-hop networks.

This thesis closes the gap between existing works and the diverse needs of emerging wireless CPS. The contributions address different research directions and are split into two parts. In the first part, we specifically address the shortcomings of existing communication protocols and make the following contributions to provide a solid networking foundation:

- We present Mixer, a communication primitive for the reliable many-to-all message exchange in dynamic wireless multi-hop networks. Mixer runs on resource-constrained low-power embedded devices and combines synchronous transmissions and network coding for a highly scalable and topology-agnostic message exchange. As a result, it supports mobile nodes and can serve any possible traffic patterns, for example, to efficiently realize distributed control, as required by emerging CPS applications.
- We present Butler, a lightweight and distributed synchronization mechanism with formally guaranteed correctness properties to improve the dependability of synchronous transmissions-based protocols. These protocols require precise time synchronization provided by a specific node. Upon failure of this node, the entire network cannot communicate. Butler removes this single point of failure by quickly synchronizing all nodes in the network without affecting the protocols' performance.

In the second part, we focus on the challenges of integrating communication and various control concepts using classical time-triggered and modern event-based approaches. Based on the design, implementation, and evaluation of the proposed solutions using real systems and networks, we make the following contributions, which in many ways push the boundaries of previous approaches:

- We are the first to demonstrate and evaluate fast feedback control over low-power wireless multi-hop networks. Essential for this achievement is a novel co-design and integration of communication and control. Our wireless embedded platform tames the imperfections impairing control, for example, message loss and varying delays, and considers the resulting key properties in the control design. Furthermore, the careful orchestration of control and communication tasks enables real-time operation and makes our system amenable to an end-to-end analysis. Due to this, we can provably guarantee closed-loop stability for physical processes with linear time-invariant dynamics.
- We propose control-guided communication, a novel co-design for distributed self-triggered control over wireless multi-hop networks. Self-triggered control can save energy by transmitting data only when needed. However, there are no solutions that bring those savings to multi-hop networks and that can reallocate freed-up resources, for example, to other agents. Our control system informs the communication system of its transmission demands ahead of time so that communication resources can be

allocated accordingly. Thus, we can transfer the energy savings from the control to the communication side and achieve an end-to-end benefit.

- We present a novel co-design of distributed control and wireless communication that resolves overload situations in which the communication demand exceeds the available bandwidth. As systems scale up, featuring more agents and higher bandwidth demands, the available bandwidth will be quickly exceeded, resulting in overload. While event-triggered control and self-triggered control approaches reduce the communication demand on average, they cannot prevent that potentially all agents want to communicate simultaneously. We address this limitation by dynamically allocating the available bandwidth to the agents with the highest need. Thus, we can formally prove that our co-design guarantees closed-loop stability for physical systems with stochastic linear time-invariant dynamics.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor Marco Zimmerling. This thesis was only possible due to his continuous support in all aspects of the work. I would also like to thank him especially for always being understanding of personal circumstances. This dedication is evident in the exceptional way he supervises and mentors doctoral students and his aspiration for high-quality research. Therefore, I am very grateful to him for giving me the opportunity as his first doctoral student and for being part of the beginnings of his research group.

Furthermore, I would like to thank all my former and current colleagues for the wonderful time as well as their support over all the years. I would especially like to thank my long-time office mate Carsten Herrmann, who has always been a source of inspiration and motivation with his extensive and profound knowledge about almost everything I have asked. In addition, I would like to thank Kai Geißdörfer for his support, hardware excursions, coffee breaks, and the dinners we had together. My family and I always appreciated his calm and friendly manner. Thanks also to Conny Okuma for helping with all the paperwork and orders.

Essential for the results of this thesis was also the very successful collaboration with our project partners Dominik Baumann and Sebastian Trimpe. I am grateful for the pleasant and productive working atmosphere, and I always enjoyed our project meetings and discussions. Looking back, I happily recall the countless hours Dominik and I spent either physically or virtually in the lab together. Our collaboration could hardly have been better. Even if debugging was painful at times, we never lost our confidence.

Furthermore, I was fortunate to work with very talented and supportive individuals from the TEC group at ETH Zurich. Special thanks go to Lothar Thiele, Andreas Biri, and Romain Jacob, who were always there with advice and support. I would also like to thank Reto Da Forno for his help with the FlockLab testbed.

Many thanks also to my former colleagues and friends Frank Busse, Martin Nowack, and Robert Krahn, with whom I had the pleasure of gaining my first experiences in the scientific environment. I look back on this time with pleasure.

Last but not least, my deepest gratitude goes to my family for all their sacrifice and support during this journey.

Finally, the works presented in this thesis were supported by the German Research Foundation (DFG) through SPP 1914 project EcoCPS (grants ZI 1635/1-1), the Emmy Noether project NextIoT (grant ZI 1635/2-1), and the Center for Advancing Electronics Dresden (grant EXC 1056).

Contents

List of Abbreviations

AP application processor

BLE Bluetooth Low Energy

CP communication processor

CPS cyber-physical systems

CRC cyclic redundancy check

DPP dual-processor platform

ETC event-triggered control

i.i.d. independent and identically distributed

LTI linear time-invariant

LWB Low-Power Wireless Bus

LQR linear quadratic regulator

MSB mean square bounded

PHY physical layer

RLNC random linear network coding

SFD start of frame delimiter

ST synchronous transmissions

STC self-triggered control

TTW Time-Triggered Wireless

WSN wireless sensor network

List of Figures

List of Tables

1

Introduction

1.1 Motivation

Technological advances lead to an increasing digitalization of our world. The accompanying interconnection between humans, machines, and our environment transforms industries, businesses, and our daily lives. Cyber-physical systems (CPS) play an essential role in this transformation, as the integration of processes in our physical world with the computing and communication capabilities of the cyber world is at the core of these systems [39]. Figure 1.1 shows the main components of CPS. Sensors monitor relevant parameters of the physical process, while actuators can directly interact with the process and influence its state. Based on the sensed process state and a fundamental understanding of the physical process, appropriate commands for the actuators are computed by a controller. The interdependencies between the computations and the process state lead to a feedback loop whose characteristics and requirements depend mainly on the dynamics of the physical process and the specific control task. Because sensors, actuators, and controllers are often not co-located due to the dimensions of the physical process (e.g., in a refinery), information must be exchanged via a communication network (wired or wireless). In general, CPS are complex systems, and their design requires interdisciplinary knowledge and engineering principles from different fields, such as control, communication, and embedded systems [39, 144].

There are many different research fields and applications in which CPS play an important role. For example, nurtured by growing demands for individualized, high-quality products, and to maximize profit margins, smart factories will use CPS to realize highly automated and adaptive manufacturing flows [17, 66, 184]. Moreover, considerable challenges in the agricultural sector are caused by, for example, environmental damages, climate change, and an ever-increasing world population, demanding more efficient and sustainable farming methods [176]. In this context, CPS will monitor

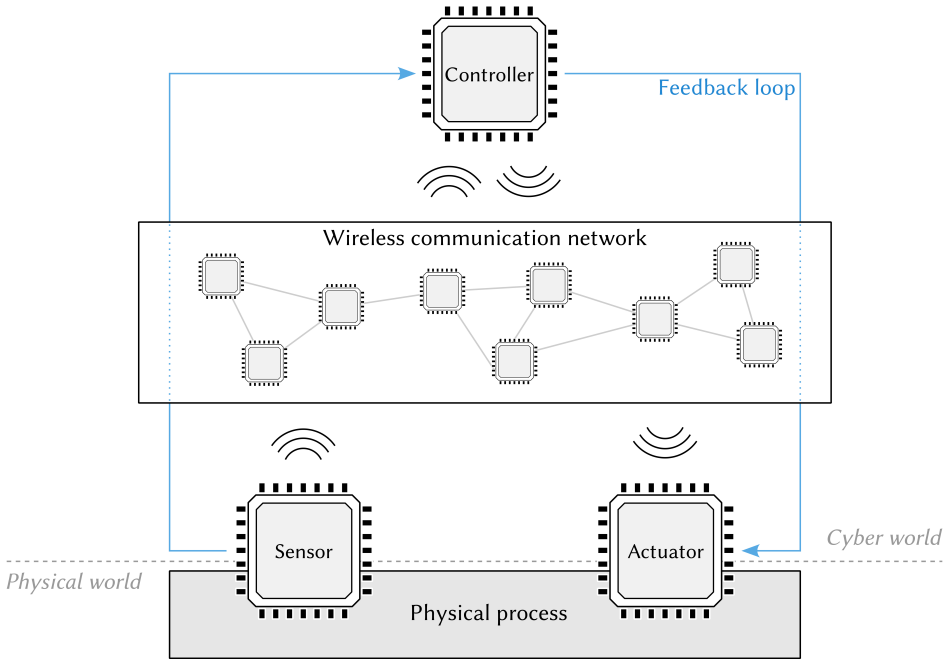


FIGURE 1.1: Wireless CPS. Embedded devices are deeply integrated into the physical environment to monitor its state (sensor). Based on that, control commands are computed (controller) and applied through actuators, closing the feedback loop. A communication network interconnects spatially distributed devices and distributes all information over large distances. Thereby, wireless networks achieve unprecedented flexibility at much lower costs.

the environment and control the precise and efficient use of water, fertilizers, and pesticides, ultimately reducing the footprint of the agricultural sector while increasing crop yield. Other examples where CPS serve as an important building block include personalized healthcare [167], next-generation power grids [46], intelligent transportation systems [19], autonomous search and rescue [61], and the optimization of various processes in entire cities [82].

Wireless cyber-physical systems. Closing feedback loops using wireless technology (see Figure 1.1) has the potential to advance the field of CPS, improving existing and enabling new applications. Wireless CPS can significantly reduce costs and provide higher mobility, better scalability, and unparalleled flexibility [4, 190].

In comparison, installation and maintenance of a cable-based infrastructure are costly and error-prone, especially due to a large number of connections [26, 122]. Once installed, the communication infrastructure is static and scales poorly toward larger deployments in general. The high costs often require a trade-off between the number of sensors and actuators best suited for the specific physical process and an economically viable solution. In contrast, embedded devices that communicate wirelessly are inexpensive and flexible, can be deployed densely and at scale, easily extend existing systems, and can reach into tiny spaces. Truly untethered devices become possible using batteries, energy harvesting [139], or wireless power transfer techniques [21] and provide the highest flexibility. This flexibility opens up new possibilities as previously

unreachable information can now be unlocked, for example, at moving and rotating parts, over impassable terrain, or in challenging environmental conditions such as heat and abrasive substances [45, 88, 147].

1.2 Application Requirements

In addition to the need for low-cost and embedded devices, CPS applications also require high dependability, efficiency, and adaptivity, particularly in medical and industrial settings [17]. We will discuss these requirements in more detail in the following.

Dependability. A dependable system that can be trusted to safely and correctly fulfill its desired task is essential to most applications. Dependability includes various aspects, such as reliability, availability, predictability, and fault tolerance.

A prerequisite for the correct operation of CPS is the reliable communication of sensor data and control commands, as the latter depend on recent information about the state of the physical process. Missing or delayed information can lead to reduced control performance or compromise the stability of the feedback system. However, since CPS are distributed systems and potentially consist of a large number of devices, failures will be no exception. For example, broken sensors, software bugs, or empty batteries can render parts of the system non-functional at any time. Therefore, the impact of such individual failures should affect the rest of the system as little as possible [45]. A reliable and fault-tolerant operation plays an important role in industry, especially in safety-critical systems where already a few message losses may lead to severe outages, substantial financial losses, or even accidents involving humans [4, 190].

In addition to reliability, predictability is another crucial aspect. The physical process and its characteristics determine how fast a control loop must be closed, imposing real-time constraints on the CPS. For example, typical update intervals in industrial applications range from tens or hundreds of milliseconds to a few seconds [4]. Hence, the timing of communication and computations must be predictable and deterministic with as little variation as possible [101, 135]. Such properties are the foundation for formal guarantees on a system's end-to-end behavior and to build trust in the proposed solution, for example, by analyzing safety, stability, or control performance [17].

Efficiency. As CPS applications become ever more sophisticated, they will use more agents, have higher bandwidth demands, and potentially close multiple feedback loops over the same network [2]. Thus, the efficient use of available resources, particularly network bandwidth, is indispensable. For instance, it is expected that the traffic volumes of machine-to-machine communication for monitoring and control will increase by up to 50 % per year over the next ten years [38].

Efficiency is also a primary concern when using completely untethered devices that rely only on stored and potentially harvested energy. Replacing batteries or other maintenance work is costly, especially when there are many devices or the system is

deployed in remote or hard-to-reach locations. Thus, an energy-efficient operation is mandatory to achieve desired operational lifetimes and reduce maintenance costs. For example, industrial sensors, such as vibration, temperature, and pressure sensors, have lifetime expectations of several years [4].

Adaptivity. The flexibility of wireless CPS is paving the way for new and highly dynamic applications. Mobile and heterogeneous agents, such as different autonomous robots in a smart factory [17], can move around freely or where needed, complementing or replacing the static infrastructure. The networks that emerge due to this flexibility are continuously changing and potentially need to extend over long distances, for example, to cover entire factories or large open areas [61, 176]. Therefore, CPS must adapt quickly at run-time to changing conditions within the system itself or in the environment.

Furthermore, the systems need to adapt to different and possibly varying communication demands, including traffic load and communication patterns, depending on the application and control design. For instance, while distributed control approaches scale better and are more robust [190], they have different communication requirements than centralized control. That is, instead of forwarding all sensor data toward a distinct central point and then distributing the control commands to the actuators, information must flow efficiently from and to multiple locations. Examples include collaborative robots [86] and swarming drones [30, 140], which frequently require mutual information to coordinate their actions and avoid crashing into each other. Moreover, some distributed control approaches are only tractable if each agent has complete information about the system's state [22].

1.3 Challenges

The development of wireless CPS faces significant challenges in meeting the various application requirements. We address the main challenges in the following.

Imperfections of wireless communication. Wireless networks transport information, for example, via radio waves, without using a continuous, guided, and typically protected medium such as cables. Thus, the antennas can generally broadcast information in all directions, a prerequisite to high flexibility. However, the downside is that wireless communication is much more susceptible to interference, fading effects, and environmental changes, decreasing communication reliability. For example, different signals or multi-path reflections of the same signal can overlap and interfere, leading to message loss, a common phenomenon in wireless networks. Unfortunately, these losses often have a significant correlation in space and time [162, 186], interrupting feedback loops unpredictably and making theoretical analyses and proofs of the system's behavior hard, if not impossible.

In addition, since transmitted signals get attenuated over distance and when passing through obstacles, the communication range and bandwidth are very limited depending

on the wireless technology and the frequencies used. The constrained bandwidth affects how quickly feedback loops can be closed and lowers the number of messages carrying sensor readings and control signals compared to wired solutions. Furthermore, the lower throughput and the message losses result in longer and varying delays for the message exchange, making it difficult to meet real-time requirements. Because of these wireless imperfections, closing feedback loops over a wireless network comes with significant challenges [107, 175].

Dynamics of multi-hop networks. The networks of emerging CPS can quickly exceed the communication range of individual devices, necessitating complex and dynamic mesh topologies. On the one hand, such topologies are more robust and fault-tolerant because each device has multiple options for forwarding messages instead of a single link, which represents a single point of failure [92]. On the other hand, effective and efficient communication in multi-hop networks requires significant coordination between the devices.

Other than in single-hop networks, where all devices can directly exchange messages with each other, exchanging messages over multiple hops introduces various challenges. Since messages can take different paths toward their destination, there are varying delays, out-of-order deliveries, and duplicates. Well-known problems such as hidden or exposed nodes hinder communication when coordination is lacking. In addition, the theoretical throughput in multi-hop networks is bounded to at most half of the throughput of single-hop networks [133].

Mobile devices, failures, and a changing environment add another dimension: time-varying wireless communication links. The required coordination of nodes becomes an ongoing effort because communication links will continuously appear, disappear, and change in quality. The latter can also be observed when the devices are static [163].

Overall, the challenges associated with dynamic multi-hop networks add to the wireless imperfections, making communication less reliable, unpredictable, and elaborate to coordinate, eventually impairing the control performance [191].

Highly constrained resources of embedded devices. Embedded devices that can be deeply integrated into the physical environment and are expected to have long and maintenance-free lifetimes are subject to resource constraints along different dimensions. For example, reduced computing capabilities, small memories, and power constraints limit the algorithmic complexity of programs and affect the control and communication performance. However, these constraints conflict with the ever-increasing demands for higher performance, dependability, and scalability of CPS. The design of such systems thus becomes a non-trivial balancing act between achievable throughput, communication range, and energy cost.

1.4 State of the Art

In this section, we discuss the state of the art in wireless CPS and describe the gaps with respect to the envisioned application scenarios. We begin with an examination of the current state of communication technology.

Communication technology. Today's practically relevant wireless communication protocols used in industry are, for example, WirelessHART, ISA100.11a, ZigBee, Bluetooth, and Wi-Fi [17, 184]. Inspired by communication protocols in wired networks, these wireless protocols gather information about the quality of wireless links and the overall network topology to derive routing information and coordinate the interaction between the nodes. The performance and efficiency of these link-based routing protocols depend on the correctness and actuality of the topology information, which requires regular maintenance to keep up with changes in the network, for example, to repair broken routes. However, when the network is dynamic and communication links frequently change, the maintenance overhead drastically increases and occupies valuable communication bandwidth. Consequently, routing-based protocols become overloaded and are unable to support the dynamic infrastructures of emerging CPS. Unfortunately, this counteracts one of the key selling points of wireless communication: its flexibility.

Wireless communication protocols typically use techniques such as retransmissions, multi-path routing, and different medium access strategies to increase the reliability of the message exchange. However, while effective, these techniques further amplify the problem of varying and unpredictable communication delays, making it difficult to provide real-time and control performance guarantees.

In addition, realizing feedback or distributed control, for example, coordinating a swarm of drones [140], requires that messages are exchanged deterministically between all or a subset of devices (i.e., many-to-many) - essentially "in all directions" [4]. However, the traffic patterns of most wireless communication protocols lack support for efficient many-to-many communication between multiple sensors, controllers, and actuators. For instance, in WirelessHART and ISA100.11a, the traffic can flow from all devices toward a gateway or vice versa. The lack of proper communication support results in poor control performance or the infeasibility of closed-loop stability [187].

Moreover, state-of-the-art wireless protocols often have centralized designs featuring a distinct network manager. This manager coordinates the message exchange by collecting topology information and distributing communication schedules. Although centralized designs are generally more straightforward and easier to analyze, the central instance is always a single point of failure and becomes a bottleneck as systems are scaled up [101, 190]. Hence, due to wireless imperfections and inefficient communication support, distributed and scalable solutions that can meet the high demands of CPS are missing today.

Wireless cyber-physical systems. The strong interconnection between communication and control nurtured research on control under network-induced constraints, with relevant works being surveyed in [69] and [191]. These works analyze the impact of, for example, limited bandwidth, varying delays, and packet loss on the control system and develop new control designs. However, wireless control has mainly been analyzed theoretically in the past, and only a few works provide empirical results, for example, in simulated case studies using WirelessHART [96, 111]. Despite progress over the years, wireless CPS are rare in industry, except for non-critical tasks with relaxed requirements [17]. The main reason is the lack of trust in wireless solutions, particularly concerning dependability. Thus, real-world experiments on realistic CPS testbeds must complement a rigorous theoretical analysis [32, 101].

Table 1.1 lists selected works evaluated on physical platforms and real wireless networks. The various works include: structural control of buildings, bridges, or tunnels [109]; adaptive lighting in road tunnels [27]; power management in data centers [150]; control of water levels in double-tank systems [9, 10]; control of underwater and ground robots [68, 151, 152], and the well-known academic example of stabilizing an inverted pendulum [14, 44, 67, 136, 165].

One of the key observations is that there is a considerable gap between the requirements of emerging CPS and the combination of properties that current approaches can provide, particularly the support for multi-hop communication makes a difference. So far, multi-hop communication has only been shown for slow processes with update intervals on the order of several seconds. In contrast, a stability analysis that considers the entire system relies on small single-hop networks. Moreover, a distributed implementation, where nodes can make local decisions based on their own information and information received from other nodes, does not exist due to the constrained traffic patterns of existing wireless communication protocols. Also, the possibility of reallocating or saving unused resources is not self-evident, although it is essential for efficiency and scalability. Advances in control design, for example, event-triggered control (ETC) and self-triggered control (STC) methods [63, 124], show that significant savings are possible without sacrificing control performance. For example, Araújo et al. [9, 10] use a STC approach to reduce the generated control traffic. However, they also note that these savings do not directly carry over to the communication system because communication resources must be arbitrated and allocated ahead of time. Therefore, gaining a true end-to-end benefit is an ongoing and non-trivial integration challenge.

1.5 Contributions and Road Map

The contributions of this thesis are split into two parts. In the first part, we focus on the networking foundation and address the shortcomings of state-of-the-art low-power wireless communication protocols to satisfy higher application demands. We then integrate communication and control in the second part and build real wireless CPS to validate our theoretical findings regarding stability and performance guarantees.

TABLE 1.1: Selected works on wireless CPS that have been evaluated on physical platforms and real wireless networks [17]. There is a large gap between the requirements of emerging CPS applications (regarding dependability, adaptivity, and efficiency) and the capabilities of state-of-the-art approaches. In particular, multi-hop communication poses a significant challenge in combination with the other requirements.

	Dependability		**Adaptivity**			**Efficiency**	
Work	Stability analysis	Short update intervals	Multi-hop	Mode changes	Distributed implementation	Reallocation	Resource savings
[9, 10]	✓	✗	✗	✗	✗	✓	✓
[44]	✓	✓	✗	✗	✗	✗	✗
[136]	✓	✓	✗	✗	✗	✗	✗
[14]	✓	✓	✗	✗	✗	✗	✗
[68]	✓	✓	✗	✗	✗	✗	✓
[151, 152]	✓	✓	✗	✗	✗	✗	✗
[109, 165]	✗	✓	✗	✗	✗	✗	✗
[67]	✗	✓	✗	✗	✗	✗	✗
[27]	✗	✗	✓	✗	✗	✗	✗
[150]	✗	✗	✓	✗	✗	✓	✓

Thereby, we make use of classical time-triggered control designs as well as modern event-based approaches.

Each chapter features the design, implementation, and evaluation of the proposed solution based on real hardware and wireless networks, which distinguishes our work from many prior works. Building real systems requires, in addition to the engineering effort, dealing with the complex interaction between communication and control. However, such solutions are necessary to build trust and acceptance, ultimately leading to a broader application. In the following, we give an overview of our contributions, providing a road map for the thesis.

Part I: Networking Foundation

Efficient and scalable many-to-all communication (Chapter 2). State-of-the-art communication protocols have various shortcomings when it comes to supporting CPS. The communication is typically coordinated based on up-to-date information about the network state, such as the network topology. However, mobile agents and environment dynamics frequently change the topology and result in increased coordination overhead. As systems scale up, using more agents and covering larger areas, the overhead explodes and eventually affects the control performance. Moreover, centralized designs and constrained traffic patterns, for example, one-to-all, poorly support distributed control applications.

To address this problem, we have developed Mixer, a many-to-all communication primitive for dynamic wireless multi-hop networks consisting of low-cost embedded devices. Many-to-all communication is universal as it allows the realization of any possible traffic pattern. Mixer's novel combination of random linear network coding (RLNC) and synchronous transmissions (ST) enables the efficient and topology-agnostic distribution of information in dynamic networks. Thereby, it approaches the order-optimal scaling behavior regarding the distributed information. Our evaluation shows that Mixer improves latency, energy efficiency, and reliability compared with state-of-the-art communication protocols.

This chapter is based on the following publication:

Carsten Herrmann[⋆], Fabian Mager[⋆], and Marco Zimmerling. "Mixer: Efficient Many-to-All Broadcast in Dynamic Wireless Mesh Networks." In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 145–158, Shenzhen, China, 2018. doi:10.1145/3274783.3274849.

[⋆] Both authors contributed equally to this work.

Highly available and fault-tolerant wireless communication (Chapter 3). Dependability is one of the most critical requirements for CPS and requires, among other things, the ability to cope with faults. For instance, individual device faults should have no or only a minor impact on the message exchange between functioning devices. However, Mixer and all other ST-based protocols, despite their exceptional

performance and several other benefits for wireless CPS, suffer from a single point of failure. The reason is that a particular node starts the communication, providing a time reference to the other nodes to meet the strict timing constraints of ST. A failure of this node leads to a communication outage. This fundamentally impairs the availability of the communication service in the presence of node failures and in case the network splits into partitions.

We have addressed this problem with Butler, a distributed and lightweight synchronization mechanism. Butler is used on demand and reliably synchronizes the nodes in the network at low overhead. As a result, after executing Butler, all nodes can start communication, which removes the single point of failure. Butler dramatically increases the availability of ST-based protocols without affecting their performance.

This chapter is based on the following publication:

Fabian Mager, Andreas Biri, Lothar Thiele, and Marco Zimmerling. "Butler: Increasing the Availability of Low-Power Wireless Communication Protocols." In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 108–119, Linz, Austria, 2022. https://dl.acm.org/doi/10.5555/3578948.3578958.

Part II: Wireless Cyber-Physical Systems

Feasibility of fast feedback control with stability guarantees (Chapter 4). Existing solutions that were evaluated in practice (see Table 1.1) are, for the most part, limited to small single-hop networks. Those that support multi-hop communication do not achieve the efficiency and dependability to control fast physical processes with update intervals ranging from tens to a few hundred milliseconds [4]. In addition, due to the many challenges of multi-hop communication, an analysis of the system's properties becomes complicated and prevents any guarantees in existing works.

To close this gap, we have developed a wireless embedded system based on a tight co-design of communication and control. Our approach mitigates the wireless imperfections to the extent possible, considers the resulting communication properties when designing the control system, and schedules the different run-time tasks to satisfy real-time constraints. Thus, our solution is the first to provide fast feedback control and coordination across real low-power wireless multi-hop networks. Moreover, we analyze the end-to-end properties of our system, provide formal guarantees on its closed-loop stability, and validate these guarantees on a real CPS testbed.

This chapter is based on the following publication:

Fabian Mager*, Dominik Baumann*, Romain Jacob, Lothar Thiele, Sebastian Trimpe, and Marco Zimmerling. "Feedback Control Goes Wireless: Guaranteed Stability over Low-power Multi-hop Networks." In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 97–108, Montreal, Quebec, Canada, 2019. doi:10.1145/3302509.3311046. **(Best paper award)**

* Both authors contributed equally to this work.

Efficient arbitration and allocation of communication resources (Chapter 5). As CPS applications become ever more sophisticated and require, for example, more agents, shorter update intervals, and higher-volume data streams, the generated traffic will inevitably exceed the available network bandwidth. ETC and STC approaches account for this by transmitting data only when needed, improving scalability and enabling energy savings. However, an end-to-end solution that transfers those benefits to multi-hop networks and that can reallocate freed-up bandwidth to additional agents or other data sources is still missing.

Therefore, we have proposed control-guided communication, a novel co-design for distributed STC over wireless multi-hop networks. Our control system predicts transmission demands and informs the communication system ahead of time so that the latter can allocate its resources most efficiently.

This chapter is based on the following publication:

Dominik Baumann[⋆], Fabian Mager[⋆], Marco Zimmerling, and Sebastian Trimpe. "Control-Guided Communication: Efficient Resource Arbitration and Allocation in Multi-Hop Wireless Control Systems." *IEEE Control Systems Letters*, 4(1):127–132, 2020. doi:10.1109/LCSYS.2019.2922188.

[⋆] Both authors contributed equally to this work.

Stability guarantees under network overload (Chapter 6). Existing approaches using ETC or STC, such as our contribution in Chapter 5, can significantly reduce the generated control traffic on average. However, they cannot prevent overload situations where all agents want to communicate simultaneously. These situations lead to unpredictable message loss and make it impossible to provide stability guarantees or performance bounds.

To solve this overload problem, we have presented a co-design of communication and distributed control using a predictive triggering approach. The agents predict their communication needs, based on the control objective and their states, in terms of priorities. An efficient distribution of these priorities in the network enables the dynamic arbitration of the available bandwidth to the agents with the highest need to communicate their current state. We prove that our co-design guarantees closed-loop stability for heterogeneous physical systems with stochastic linear time-invariant (LTI) dynamics.

This chapter is based on the following publication:

Fabian Mager[⋆], Dominik Baumann[⋆], Carsten Herrmann, Sebastian Trimpe, and Marco Zimmerling. "Scaling Beyond Bandwidth Limitations: Wireless Control With Stability Guarantees Under Overload." *ACM Transactions on Cyber-Physical Systems*, 6(3):20:1–20:30, 2022. doi:10.1145/3502299

[⋆] Both authors contributed equally to this work.

Author Contribution Statement

This thesis is largely based on the results of the EcoCPS project, which is part of the Priority Program 1914 on cyber-physical networking of the German Research Foundation (DFG).

In the first part of this thesis, dealing with the networking foundation, all authors were actively involved in formulating the research problems, developing the solutions, evaluating the results, and writing the paper. The order of authors reflects their contribution, with the first author being the main contributor unless otherwise noted.

Within EcoCPS, we have also worked closely together with Dr. Dominik Baumann and Prof. Sebastian Trimpe from the Institute for Data Science in Mechanical Engineering at RWTH Aachen. Their expertise in control systems has complemented our expertise in wireless communication and embedded hardware/software design. Accordingly, the contributions in the second part of this thesis, addressing the co-design of control and communication, are joint efforts. Together we have formulated the research goals, developed the overarching system design, particularly the interfaces between communication and control, and built CPS testbeds to validate our approaches on real physical systems and networks. Concerning the implementation and analysis of the different CPS components, the author of this thesis primarily contributed to the communication-related parts and the embedded engineering. By contrast, the equally contributing author focused on the control-related parts.

Part I
Networking Foundation

2

Mixer: Efficient Many-to-All Broadcast in Dynamic Wireless Mesh Networks

Preface

In the first part of this thesis, we focus on the networking foundation for wireless CPS. State-of-the-art wireless communication protocols do not satisfy the diverse needs of emerging CPS applications. There are many reasons for this, including that the protocols are not designed for dynamic networks, scale poorly toward larger deployments and higher traffic demands, or fail to meet the timing and reliability requirements of feedback control applications.

To address these shortcomings, we present Mixer in this chapter, a many-to-all broadcast primitive for dynamic wireless mesh networks. Many-to-all communication has the advantage that it is widely applicable and can serve any traffic pattern, an essential requirement for efficient distributed control and adaptive CPS. To achieve an efficient operation in real networks, we design Mixer in response to the theory of RLNC and the characteristics of physical layer (PHY) capture. Thus, Mixer approaches the order-optimal scaling in the number of messages to be exchanged. Our evaluation shows that Mixer outperforms the state of the art across all metrics by up to 3.8× and provides a reliability greater than 99.99 %, even at a node moving speed of 60 km/h.

2.1 Introduction

Many-to-all broadcast is the process of disseminating information from one, multiple, or all nodes to every node in a network. It is a universal communication primitive as it can serve any possible traffic pattern (point-to-point, one-to-many, all-to-all, ...), and it is fundamental for a growing number of applications and network services involving multiple sources and multiple destinations.

For example, closing distributed feedback loops in CPS relies on wireless communication among sensors (sources), actuators (destinations), and controllers (acting as sources and destinations). To enable coordination in autonomous systems, such as collaborative agents [86], robotic materials [33], and swarming drones [140], each node needs to collect information (e.g., location) from every other node and disseminate its own information to all others. Indeed, a certain class of closed-loop control problems is only tractable if each node can make decisions with knowledge of the full system state [22], requiring many-to-all communication.

The same need arises in support of programming abstractions [130] and fault-tolerance mechanisms [156], for example, when some application logic is replicated across multiple distributed devices and nodes need to report every message to all replicas [50]. The initial distribution of messages across sources depends on the application and can also change dynamically at run-time. For instance, in drone-assisted disaster response, all nodes need to regularly exchange one message with all others to prevent collisions or to keep a desired flight formation [30], while sometimes one node may have multiple messages to disseminate, such as an image taken with an on-board camera informing a group of human rescuers on the ground [61].

To support these emerging applications, a many-to-all broadcast primitive needs to meet the following key requirements:

- *Fast and reliable:* To reduce the impact on application performance and to keep up with the dynamics of physical processes, many-to-all communication must be fast (i.e., support end-to-end communication delays and intervals of a few hundred milliseconds [4] or less) and also highly reliable [160].
- *Support for dynamic mesh topologies:* Rotating, flying, or otherwise mobile entities cause significant network dynamics, while multi-hop communication and mesh topologies are either beneficial or a necessity for the application scenario [18, 65, 108].
- *Support for adequate message sizes:* Many applications feature payloads that are tens of bytes in size or larger [61, 108].
- *Energy efficient:* The employed devices are often battery-powered [4, 132] or harvest energy from the environment [33]. Moreover, size and weight constraints call for small batteries, low-power radios, and resource-limited microcontrollers [61, 132].

Existing many-to-all solutions fall short of these requirements. Approaches based on routing, such as WirelessHART, ISA100.11a, and RPL on top of TSCH (6TiSCH), which exchange messages via an explicitly built and maintained structure, target different scenarios with static nodes and packet intervals of several seconds [43]. Using them for many-to-all broadcast may require many-to-one upward routing followed by one-to-all downward routing, which suffers from scalability, efficiency, and reliability issues [72]. Furthermore, since these solutions rely on a known and stable network topology, they may fail in distributed or uncoordinated settings [59] and perform poorly in the presence of mobile devices [49] or other network dynamics [157]. Some recent proposals based on ST overcome this problem by decoupling the protocol logic from the time-varying network topology. For example, Chaos works well for all-to-all exchange of small payloads (e.g., one byte per node) as required for network-wide consensus [5] and data aggregation [91], but performs inefficiently for payloads larger than a few bytes [113]. A series of network-wide Glossy floods [48] is then a better option; however, the required bandwidth and overall latency increase rapidly with the number of messages to be exchanged.

Contribution and road map. This chapter presents Mixer, a new many-to-all broadcast primitive for dynamic wireless mesh networks. Mixer supports the full spectrum from one-to-all to all-to-all communication, and significantly outperforms prior many-to-all solutions in latency, goodput, and radio-on time while providing nearly perfect reliability despite significant network dynamics.

The key ideas behind Mixer are as follows: a) Rather than performing M sequential floods to disseminate M messages, Mixer *overlays* all M floods by letting nodes mix packets using *RLNC*. This way, Mixer disseminates all M messages at once and approaches the theoretically optimal scaling as M increases. b) Mixer combines RLNC with ST. While RLNC aims to maximize the utility of individual packets, ST aim to maximize spatial reuse.

To exploit the synergy of both concepts for efficient many-to-all communication in real wireless networks, we must tackle a number of challenges as outlined in Section 2.2. Our design of Mixer, described in Section 2.3, addresses these challenges and yields significant improvements compared to a straightforward combination of the two concepts.

We prototype Mixer on the TelosB [138] (see Section 2.4), which has a IEEE 802.15.4 radio and a 16-bit MSP430 microcontroller, to allow for a fair comparison with the state of the art on public testbeds. We also port compute-intensive parts of Mixer to modern 32-bit ARM Cortex-M0+/M4 platforms to project the performance gains with more processing power and faster PHYs, such as IEEE 802.11.

We evaluate Mixer in Section 2.5 using experiments on two testbeds with up to 94 nodes, on dynamic networks with failing devices and a mobile node attached to a car driving 20–60 km/h, and through microbenchmarks on four different platforms. We find that Mixer is up to 3.8× faster and more efficient than fine-tuned sequential

Glossy floods and provides a reliability greater than 99.99 % even in the presence of node mobility. For example, Mixer achieves a goodput of up to 53.7 kbps and needs less than 300 ms to exchange 27 60 B messages in an all-to-all fashion on FlockLab [99]. Our microbenchmarks indicate that the same scenario would take less than 10 ms when running Mixer on faster CPUs and PHYs.

In summary, this work contributes the following:

- Mixer, a many-to-all broadcast primitive that approaches the order-optimal scaling in the number of messages in real dynamic wireless mesh networks.
- A design that combines RLNC with ST and thereby enables Mixer to perform efficiently in practice, while being highly reliable and resilient to network dynamics.
- An open-source implementation and experiments demonstrating several-fold performance gains over the state of the art.

2.2 Overview

This section introduces relevant concepts and provides an overview of Mixer's operating principle, scope, and key design challenges.

2.2.1 Basic Operation and Terminology

The principle behind Mixer's operation is best explained by an analogy with flooding. Assume a set of M messages is to be exchanged between N nodes. Using sequential flooding, this takes $O(M \cdot T)$, where T is the time needed to flood a single message. Although protocols like Glossy [48] achieve the theoretically minimum T in practice, the scaling with factor T becomes problematic as M grows. Mixer improves the scaling to $O(M + T)$ by considering all M messages together: Rather than performing M floods in sequence, Mixer *overlays* the M floods and simultaneously disseminates all messages. To this end, nodes mix packets using RLNC and transmit random linear combinations of previously received packets.

Mathematically speaking, each Mixer node maintains a system of linear equations given in (2.1).

$$\underbrace{\begin{pmatrix} p_1 \\ \vdots \\ p_M \end{pmatrix}}_{\substack{\text{coded} \\ \text{payloads}}} = \underbrace{\begin{pmatrix} c_{11} & \cdots & c_{1M} \\ \vdots & \ddots & \vdots \\ c_{M1} & \cdots & c_{MM} \end{pmatrix}}_{\text{coding matrix}} \cdot \underbrace{\begin{pmatrix} m_1 \\ \vdots \\ m_M \end{pmatrix}}_{\text{messages}} \tag{2.1}$$

The set of messages $m_1, \ldots, m_M$ forms a *generation* of size M. Mixer nodes exchange linear combinations of these messages, that is, the ith packet's *payload* equals $p_i = \sum_k c_{ik} m_k$, with $c_i = (c_{i1}, \ldots, c_{iM})$ the *coding vector* of packet i. A sender transmits c_i together with p_i in the same packet. A receiver extracts c_i and p_i from the packet and

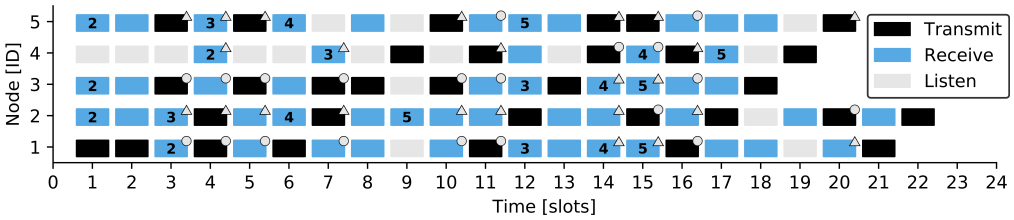


FIGURE 2.1: Real trace of MIXER in a 2-hop IEEE 802.15.4 network with 5 nodes exchanging 5 messages in an all-to-all fashion. Numbers indicate when the rank of the coding matrix C at each node increases and are equal to the current rank. Each node can receive at most one packet per slot. Symbols in the upper corner mark which node received what packet in case there are multiple transmitters in the given slot.

maintains the *coding matrix* C. When C reaches *full rank* (i.e., a node has collected M linearly independent packets), then (2.1) has a unique solution and the node can decode all messages by solving the system of linear equations. Sending nodes build packets by adding up a random subset of already collected rows (p_i and c_i) of (2.1), which is possible irrespective of C's rank. All computations are performed over the finite field GF(2). As a consequence, the size of c_i is M bits (one bit per c_{ik}).

Figure 2.1 shows a trace from a 2-hop IEEE 802.15.4 network as $N = 5$ nodes exchange $M = 5$ messages in an all-to-all fashion using MIXER; that is, initially each node has its own (one) message, and the goal is that every node acquires the (four) messages from the other nodes. The operation of MIXER, called *round*, proceeds in a series of adjacent *slots*. Node 1, the *initiator*, starts the round by transmitting (Tx) its message in slot 1, which is received (Rx) by nodes 2, 3, and 5. Since the packet contains a message that they did not know before, the rank of their coding matrix C increases from 1 to 2. As a result of this, nodes 2, 3, and 5 may now start to mix packets using RLNC and transmit linear combinations of multiple messages. Note that the use of RLNC allows the nodes to pick the coefficients of c_i randomly without any knowledge of the current network topology.

Every time a node receives an *innovative* packet—one that is linearly independent from all previously received packets—the rank of the coding matrix C increases. Once C has full rank, all messages can be decoded using, for example, Gaussian elimination. In the example of Figure 2.1, node 2 is the first to reach full rank in slot 9.

We also see in Figure 2.1 that often multiple nodes transmit in the same slot. This happens for the first time in slot 3; however, both node 1 and node 2 receive despite the collision. While traditional solutions try to *avoid* collisions using carrier sensing, handshaking, or scheduling, MIXER and several other recent proposals (e.g., [48, 143]) aim to *take advantage* of collisions to improve spatial reuse. Since nodes in MIXER typically transmit different packets (as they mix messages randomly), a common receiver can successfully receive one of the packets only due to the *capture effect* [94, 102].

The capture effect occurs if certain signal power and timing conditions are met. For instance, using IEEE 802.15.4 radios operating in the 2.4 GHz band with OQPSK modulation, the SINR at the receiver must exceed 3 dB and the packet with the strongest signal must arrive no later than 160 µs after the first packet [91]. Although the exact SINR and timing conditions are highly dependent on the concrete PHY, the capture effect has already been exploited in many popular wireless technologies, including Bluetooth Low Energy (BLE) [149], IEEE 802.11 [93], and IEEE 802.15.4 [102].

Scope. Although this work focuses on low-power embedded systems and IEEE 802.15.4, in principle Mixer works on any PHY that features the capture effect. Moreover, unlike analog network coding [81, 131], Mixer does not require any changes to existing PHYs and hence runs on commodity low-power devices.

Similar to Glossy [48] and other recent works [41, 42, 91], Mixer is a communication primitive that conceptually sits between the PHY and a higher-layer protocol. This higher-layer protocol is responsible for informing all N nodes about the (dynamically changing) initial distribution of M messages to nodes before each Mixer round. Alternatively, Mixer can also be used with a statically configured initial distribution. Determining such distributions and building generations of messages is beyond the scope of this work, but existing techniques can be used [64, 98] and coupled with existing higher-layer protocols (e.g., Low-Power Wireless Bus (LWB) [49] and A^2 [5]) that readily support the network-wide scheduling of Mixer rounds.

2.2.2 Design Challenges

Theoretical results suggest that RLNC-based gossip protocols like Mixer perform optimally in static and dynamic networks [36, 60]. Specifically, it has been shown that the number of slots needed to disseminate all M messages has order-optimal scaling $O(M + T)$ [60]. This result is based on specific connectivity measures of the (time-varying) network graph, and the constant factors hidden by the O-notation heavily depend on these properties. In Mixer, the connectivity of the wireless network is tightly coupled to the extent to which the capture effect can be exploited and changes from one slot to the next even if nodes are stationary. Our overarching goal in designing Mixer is to combine RLNC and ST such that the constants hidden in $O(M + T)$ are as small as possible. This entails addressing four main challenges:

C1: *When should a node send or listen?* A capture threshold of 3 dB is quite small, so there is a good chance to benefit from capture in practice. However, because of the very same condition, the probability of capture drops rapidly as the number of synchronous transmitters increases [102]. How can a node locally decide whether it should send or listen in a slot, maximizing spatial reuse without destroying capture?

C2: *What should a node transmit?* To achieve low latency (i.e., small number of slots), we must devise a policy that allows senders to easily build packets that are likely innovative for their neighbors even if the nodes are mobile (i.e., neighborhood changes quickly).

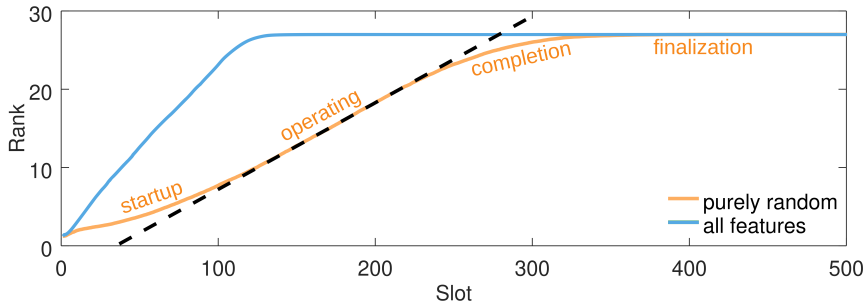


FIGURE 2.2: Average rank increase in a one-to-all scenario on FlockLab with different feature sets ($M = 27$ messages).

C3: *How to ensure ST in the absence of a global clock?* To meet the timing condition of capture, nodes must communicate in a globally slotted fashion (see Figure 2.1). This is difficult because nodes spend varying amounts of time on processing in each slot (e.g., when building the next transmit packet), which impairs synchronization in the face of different clock drifts across nodes.

C4: *How to achieve an efficient run-time operation?* RLNC improves the utilization of the wireless medium (i.e., helps reduce the number of slots), but requires nodes to store and process the coding vector and payload bytes of the packets. Limited memory and compute power may hinder harnessing these benefits (e.g., by blowing up the length of the slots), so we need to design efficient coding and decoding strategies, which are preferably running in parallel to radio activities whenever possible. Moreover, nodes need a way to locally figure out whether they are still helpful for the dissemination process or can turn off their radio to save energy.

2.2.3 Phases Within a MIXER Round

Before describing the design of MIXER in detail, we illustrate its effectiveness in addressing challenges **C1** and **C2**—the *when* and *what* to transmit—via a comparison with a naïve application of RLNC, where every node sends packets at random with a fixed transmit probability P_{TX} and builds packets by randomly summing up already collected rows. Figure 2.2 shows the average rank increase across 27 nodes on the FlockLab testbed [99] for the random approach (including our mechanisms to address challenges **C3** and **C4**) and when all design features of MIXER are enabled. Despite the fact that MIXER reduces the number of slots required to reach full rank (27 in this scenario) from 320 to 120, we can distinguish four distinct phases, which are most apparent in the purely random approach.

In the middle we see a phase with almost linear behavior, emphasized by the dashed line. We refer to this as the *operating phase* since the rate of average rank increase is high and steady. Before comes a *startup phase* with less but increasing rank growth. After the operating phase follows a time with decreasing rank improvements. We call this the *completion phase* as it eventually leads to all nodes reaching full rank. Last,

there is a *finalization phase* where nodes still communicate but no longer depend on incoming packets.

The poor performance of the random approach, especially during the startup and completion phases, induces questions on the reasons for the observed behavior and potential improvements. Consequently, these questions drive the discussion in Section 2.3. As visible in Figure 2.2, our design of Mixer improves significantly on the purely random approach, effectively addressing challenges **C1**–**C4**.

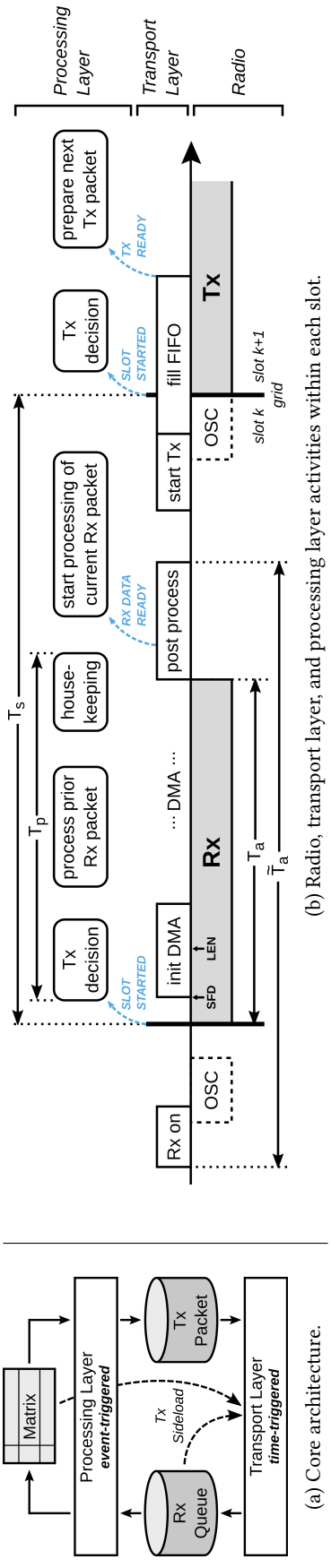
2.3 Design

This section describes the architecture of Mixer and discusses all major design elements. In Section 2.3.1 we introduce the core architecture of Mixer, which consists of a transport layer and a processing layer. Since the latter contains the most characteristic features of Mixer, we present it first in Section 2.3.2. It combines two core mechanisms (Section 2.3.2.1 and Section 2.3.2.2) with a number of phase-related features (Section 2.3.2.3 to Section 2.3.2.5), systematically addressing challenges **C1**, **C2**, and (partially) **C4**. In Section 2.3.3 we present important mechanisms of the transport layer targeting challenges **C3** and **C4**. We conclude with a short list of other design properties facilitating **C4** in Section 2.3.4.

2.3.1 Core Architecture

To significantly exploit the capture effect, the design of Mixer incorporates two key ingredients: well-synchronized, time-slotted communication to meet the capture window and a lightweight but effective mechanism to steer the set of active transmitters per time slot. From a single node's perspective, these two design elements are responsible to decide *when* to transmit (and when to receive). Their counterpart is composed of a number of mechanisms influencing *what* to transmit with the particular goal to improve the efficiency of RLNC in the given scenario. Most of the components interact with each other; Mixer represents the entirety of all design elements, translated to an appropriate software architecture.

The timing conditions of capture suggest the encapsulation of the low-level packet transport functionality in a self-contained component that provides reliable synchronous packet exchange. Mixer follows this approach with a two-layer architecture composed of a time-triggered *transport layer* and an event-triggered *processing layer* (Figure 2.3a). The layers are decoupled by receive and transmit queues, allowing a high degree of parallelized activities on both layers. Figure 2.3b shows the main activities on each layer within the slots. Each slot has the same fixed length T_s, which accounts for the air time T_a of one packet and processing time T_p. By default, the transport layer is in receive mode. Transmit decisions are made by the processing layer and passed to the transport layer on demand, which executes them in the next slot.



(a) Core architecture.

(b) Radio, transport layer, and processing layer activities within each slot.

FIGURE 2.3: Software architecture and interactions between design components in each MIXER node.

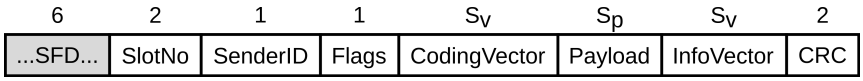


FIGURE 2.4: MIXER's packet format. Parts in gray are defined by IEEE 802.15.4, CRC is generated by the radio. Sizes in bytes.

Figure 2.4 shows the packet format. Note that the payload field contains a linear combination of messages, so its size S_p is equal to the message size. The packet size, instead, also accounts for all other fields, such as the size of the coding vector S_v.

2.3.2 Processing Layer

In the following subsections we detail the individual design mechanisms of MIXER. The transport layer is considered in Section 2.3.3; for the moment we assume it to be working such that we can take slots as the base unit of the communication grid. We start from the baseline of the purely random approach introduced in Section 2.2.3. The discussion is driven by the question on how to improve on the observed results, particularly with respect to the identified phases (Figure 2.2).

2.3.2.1 Semi-Coordinated Transmissions

Since real-world and in particular dynamic networks have varying local node densities, a fixed transmit probability P_{TX} performs unsatisfactory (Figure 2.2). An adequate policy should adapt P_{TX} to the local densities, striving for the goal to maximize the number of received packets per slot within the network. To reach this objective, the number and selection of transmitters in each slot should be well balanced—high enough and spatially distributed to reach as many receivers as possible, but still low enough to allow the capture effect to occur.

For this purpose, each MIXER node maintains a list of received SenderIDs (Figure 2.4) within the last H slots. Using this sliding-window history information, which is discarded at the end of a round, nodes monitor their current neighborhood to drive an *adaptive transmit policy* as shown in Algorithm 1. This policy updates the transmit probability P_{TX} of a node depending on the estimated local node density d. Furthermore it incorporates a kind of local round-robin scheduling on a selected subset of slots by assigning an owner to each slot. Owners use their slots to transmit for sure while neighbors respect this behavior.

The level of determinism induced by this policy increases with node density because a higher density means that a larger portion out of N consecutive slots is owned by some node within a neighborhood (in return, the number of shared slots is lower). Hence, in high-density regions of the network nodes use stronger coordination than in sparsely populated areas. In this way, we effectively reduce the uncertainty in the expected number of transmitting nodes per slot and decouple the capture probability from the node density (see **C1**). This behavior also implies that MIXER does not purely depend

Algorithm 1 Semi-coordinated Transmission (core algorithm)

```
1:  d ← 1 + num_neighbors                          ▷ local density (from history)
2:  owner ← (slot_no + 1) mod N                    ▷ assign owner to next slot
3:  if owner = my_node_id then                     ▷ if my slot: transmit
4:      P_TX ← 1
5:  else if owner is neighbor then                 ▷ if foreign slot: receive
6:      P_TX ← 0
7:  else if Tx in current slot then                ▷ do not transmit twice in a row
8:      P_TX ← 0
9:  else                                           ▷ if shared slot: transmit with probability
10:     P_TX ← 1 / (d + 1)                         ▷ (+1 accounts for an unknown neighbor)
```

on the capture effect; for example, in a one-hop network, Mixer would tend toward a fully coordinated operation, where nodes transmit one after another in dedicated slots.

2.3.2.2 Explicit Innovation Forwarding

A sender assembles a packet using RLNC: It adds every row from its matrix to the Tx packet with probability 0.5. In Mixer we add several features somewhat restricting the randomness to improve average performance. The most basic one rests upon the assumption that an innovative packet is also innovative for a node's neighborhood. Thus, a node adds every innovative packet immediately to the prepared Tx packet so the innovation gets relayed with the next transmission for sure.

To thoroughly justify this behavior, we distinguish two cases: If the innovative packet arrives from *outside* the common neighborhood (cluster), the above assumption is clearly reasonable. Otherwise, if the packet is sent from *inside* the cluster, there is still a chance that it is innovative for some neighbor(s). On the other hand, including it has no disadvantage for the other neighbors: Since the packet is innovative for the current node, it is linearly independent from its Tx packet (i.e., including it corresponds to adding an additional matrix row). Note that this behavior influences only the content of the next Tx packet (*what*), not the transmit policy (*when*).

2.3.2.3 Improving Startup: Adapted Coordination

The reasons for a slow startup phase (Figure 2.2) are best understood with a one-to-all scenario in mind. In this case, all messages reside at the initiator at the beginning of a round. After it started the round, the awoken neighbors are not able to add any innovation to the packets they send as they simply do not have any. Innovation can only be added by the initiator if it decides to transmit and if the (randomly built) packets include some. Clearly, the situation will improve a) if the initiator ensures that it sends something innovative, and b) if there is a mechanism that prioritizes the transmissions of the initiator at the beginning of a round. Again, a) and b) address the *when* and *what* to transmit.

Improving the number of innovative packets during the startup phase is easy: Since every message is initially available at exactly one node, its *originator*, this node knows that a packet will be innovative for all other nodes if it incorporates the message for the first time. Hence, if a node initially has x messages, it can easily generate x innovative packets by simply transmitting each of its x messages once. MIXER nodes do exactly this before starting to build packets at random.

Prioritizing the transmission of innovative packets requires adaptations of the transmit policy since there is no connection to packet contents so far. MIXER bridges this gap by assigning the owner role of slot k to the originator of message k during the first M slots. Thus, in case of a perfect wake-up order of all nodes, every slot $k \leq M$ is used to transmit message k by its originator, generating a fast-growing coding potential. As a side effect, this rule circumvents the question on how the standard transmit policy performs as long as the history information is very incomplete.

However, using this strategy, we have to cope with two issues: First, if the originator of message k is not awake in slot k, the slot is unused. This can lead to corner case situations in which nothing happens for a long time. MIXER avoids this problem by including the shared slot rule (cf. Algorithm 1) also during the startup phase, but with $P_{TX} = 1/\min(k, 16)$ which is independent of d and stimulates a fast wake-up of all nodes. Second, in one-to-all scenarios the initiator would transmit in every of the M startup slots and hence would have no chance to discover its neighbors. Further, if its neighbors stay silent the whole time, nodes farther away would not wake up. MIXER avoids this problem with the help of a flag (used only during startup): If a node sending in slot k is also the originator of message $k + 1$, it marks this in the packet header's flags field and stays silent in slot $k + 1$. Nodes receiving the packet in slot k then know that the owner of slot $k + 1$ will not use its slot. Thus, they transmit with $P_{TX} = 1$ in slot $k + 1$ to push packets into their "back country."

2.3.2.4 Improving Completion: Active Requests

In the completion phase we see a significant slowdown in the average rank increase with the purely random approach (Figure 2.2) due to the well-known coupon collector's problem [35, 47]. It is present here because a node cannot include messages that are outside the row space of its current matrix. Since there is a high probability that the missing pieces are also missing at a considerable number of neighbors, it is difficult to resolve the situation efficiently without any feedback on the missing pieces. MIXER nodes address this problem by progressively providing such feedback in the form of *active requests* and by adapting their transmit policy in response to these requests.

Recall that a node requires rank M to recover all messages, that is, M linearly independent rows in the coding matrix C in (2.1). MIXER nodes keep C in row echelon form. Therefore, rows can be identified with their pivot elements (the main diagonal of C): If c_{ii} is zero, then row i is missing. Below, we describe how MIXER deals with missing rows. MIXER also includes mechanisms to deal with missing columns, which

are conceptually similar, but we do not discuss them here due to space limitations. For the same reason, we skip minor details and instead focus on the main concepts.

The transition from the operating to the completion phase is floating and encompassed by a simple rule which provokes an increasing number of active requests when a node's rank tends toward full rank. Every request is communicated in the form of a flag and corresponding markers in the InfoVector field (Figure 2.4) of a packet that is anyway being sent; thus, requests do not consume extra packets or slots. If a node receives active requests, it has to decide how to react, which again translates into the questions of *what* and *when* to transmit. Before answering these two questions, we discuss how a node stores requests. Storing them is wise since it may be possible to help multiple nodes with one response packet.

In case of a request, InfoVector contains a bit field wherein each set bit marks one missing row. Nodes could store every received request separately, but this may consume a considerable amount of memory and slow down processing. Instead, Mixer nodes maintain an *any-mask* and an *all-mask*. If a request arrives, they `OR` the bit field to the any-mask and `AND` it to the all-mask. Thus, the all-mask contains bits that have been set in *all* incoming requests, while the any-mask contains bits that have been set in *at least one* request. This way, nodes get an idea of which rows might be more helpful than others. In addition to storing requests, nodes monitor the traffic and try to discover if pending requests got serviced. If not, they drop the stored requests after three slots so they do not affect the communication for a long time.

In case of pending requests, a node decides *what* to transmit as follows: It selects a requested row index from the all-mask or, if the all-mask is empty, from the any-mask. Then it tries to build a packet whose coding vector contains no non-zero elements to the left of the selected index. If this is possible, the packet is definitely innovative for the requesting node. In this case, a node is a *helper*. Otherwise, if a node cannot build such packet, it is a *non-helper* and it does not matter what it sends. Instead, it should consider *not* to send, which brings us to the question on *when* to transmit.

Extending the transmit policy with rules for request handling is nontrivial. We pursue three goals: a) Potential helpers should send preferred while non-helpers should decrease their transmit probability P_{TX}. b) NodeID-based owner roles should be more and more dissolved toward the end of a round in favor of helper/non-helper roles. c) Phases without pending requests should be unaffected. With these goals in mind, choosing P_{TX} breaks down into three tasks:

1) Identify own role: helper or non-helper.
2) Estimate role of all n neighbors respectively the number of helpers n_+ and non-helpers $n_- = n - n_+$.
3) Adapt P_{TX} based on n_+, n_-, and original owner role.

Step 1) is explained above. For step 2) we need some heuristic since it is impossible for a node to determine the required information precisely. In fact, we do not even know

TABLE 2.1: Transmit probabilities with requests pending.

	Own Slot	Foreign Slot	Shared Slot
P_{TX} Helper	$\tilde{r}/n_+ + (1-\tilde{r})$	$\tilde{r}/n_+$	$1/n_+$
P_{TX} Non-Helper	$\tilde{r}/(\mathrm{e}n_-) + (1-\tilde{r})$	$\tilde{r}/(\mathrm{e}n_-)$	$1/(\mathrm{e}n_-)$

if all neighbors process the same request masks, though this appears as a reasonable assumption at least for the majority of neighbors. A simple heuristic could estimate n_+ as a fixed percentage of n. To get more precise estimates, we design a more advanced variant which uses an additional prerequisite: Since the InfoVector field is sent with every packet anyway, the nodes always utilize it to transmit their current row state. Receiving nodes store this information in their history such that every node has a reasonably up-to-date information on the row state of its neighbors. Using this information, a MIXER node is able to estimate n_+ (and hence n_-) more precisely.

The adaptation of P_{TX} in step 3) is based on the following reasoning. In expectation, one helper should send a response packet. Thus, the transmit probability for helpers $P_{TX+} = 1/n_+$. The transmit probability for non-helpers P_{TX-} should match the probability that no helper sends a packet, that is,

$$P_{TX-} = \frac{\left(1-\frac{1}{n_+}\right)^{n_+}}{n-n_+} \xrightarrow{n_+\to\infty} \frac{\mathrm{e}^{-1}}{n-n_+} = \frac{1}{\mathrm{e}\,n_-} \tag{2.2}$$

The numerator in (2.2) converges fast, so it is sufficient to use the simplified term as approximation. Since we do not want to break the original NodeID-based owner roles abruptly, each node lets P_{TX} slide based on its relative rank $\tilde{r} = r/M$, leading to the transmit probabilities listed in Table 2.1.

2.3.2.5 Improving Finalization: Smart Shutdown

From a single node's perspective, the main communication task is accomplished when reaching full rank. The remaining slots are needless for that node, but may be useful to support unfinished neighbors. Though, at some point in time all neighbors have full rank and there is no reason to stay active; the node could turn off to save energy. However, we need a prudent signaling mechanism to establish an efficient but cautious shutdown of the nodes.

If a node has full rank, it sets a flag in every packet it sends. Receiving nodes mark the full rank status in their history. A node can shutdown if all its neighbors reached full rank. It informs its neighbors by sending a packet with a shutdown flag set before it turns off. The shutdown flag allows a neighbor to immediately remove that node from its history, leading to an immediate adaptation of its transmit policy. Without the flag, the update would be delayed until the vanished node falls out of the history window H.

Since the packet with the shutdown flag is sent only once, it could easily get lost at *some* neighbors, but it is received with high probability by *at least one* neighbor. We address this issue with two mechanisms: First, on receiving a packet with the full-rank flag set, the history window applied to the sending node gets shortened (we use $H/3$ in Section 2.5) based on the expectation that the node may shutdown in the near future. Second, the full-rank status of all nodes is actively propagated through the network with the help of full-rank nodes. Since those do not have the need to provide request information, they use the InfoVector field to mark finished nodes instead. With this mechanism it is very unlikely that a full-rank hint gets lost. Although this does not replace lost shutdown flags, it enables finished nodes to shutdown themselves instead, which eventually leads to the same result.

2.3.3 Transport Layer

2.3.3.1 Establishing the Slot Grid

On the packet transport layer we have to solve the task of establishing a reliable slot grid as a prerequisite for ST. Since system parameters are known in advance, it is possible to select a fixed nominal slot length T_s. Unfortunately, real-world hardware suffers from clock frequency offsets, so we need some mechanism that compensates for such effects and avoids that grid points drift apart among nodes. As a natural solution, we implement a phase-locked loop.

The reference signal of the phase-locked loop is built from timestamps taken on the reception of start of frame delimiter (SFD) fields, which is part of the synchronization header of each packet (see Figure 2.4). This requires that senders start transmissions with appropriate temporal accuracy, which is typically achieved via meticulous timer polling. The phase difference between the reference signal and the local slot grid is low-pass filtered and fed into a proportional-integral controller that computes a correction term for the next grid point and a start offset for transmissions (the latter counteracts a potential cumulative drift induced by time-of-flight delays). All filter coefficients and gain parameters are chosen empirically based on simulations and results from testbed experiments.

In principle it is possible to consider only SFD timestamps that stem from *specific* neighbors (e.g., predefined ones or those with minimum hop distance to the initiator). Our experiments suggest that taking reference values from arbitrary nodes is sufficient and works most of the time. If it sporadically fails, a node recognizes this situation and resynchronizes itself as part of a fallback mechanism.

We want to emphasize that the slot grid is only needed during a round (i.e., when Mixer is active). Mixer does not require to keep the slot grid between rounds. In fact, each Mixer node except the initiator assumes to be out of sync at the beginning of a round. To lock onto the slot grid, a node activates its receiver continuously until it receives the first valid frame. The maximum length of this initial listen phase is only limited by a timeout for the whole round. This timeout is chosen by the user and can be

set to a large value (e.g., in the range of seconds or even infinity), effectively decoupling the timing requirements of MIXER from the rest of the system. At the end of a round, each node returns a reference time, which can be used (e.g., by a higher-level protocol) to schedule the next round.

2.3.3.2 Updating Packets via On-the-Fly Sideload

Immediately forwarding innovation as described in Section 2.3.2.2 seems to be easy from a conceptual point of view, but it appears ambitious when looking for an efficient implementation. The reason is a hard time constraint which becomes clear when considering the transition from a receive to a transmit slot (Figure 2.3b). To achieve an optimal performance, the slot length T_s should be as small as possible (bounded by T_a plus a minimal overhead for pre- and postprocessing). On the other hand, updating the transmit packet takes time (process received packet, determine if innovative, if so: add) and has to take place (theoretically) between end of Rx and start of Tx.

With MIXER, we introduce a feature that allows to solve this problem in an elegant and efficient way. First, if a node transmits a packet, it always starts the radio *before* filling the radio chip's transmit buffer. This is possible because the transmitter needs to generate a number of synchronization symbols (despite some device specific tasks, denoted OSC in Figure 2.3b) before it sends the actual data, which provides some time for the program. As a result, a processing task can change the packet content until right before the slot starts (irrespective of the packet size). Second, our low-level transmit routines allow to add additional data to the packet content *while* the packet is moved to the transmit buffer. In other words, they allow to *sideload a second payload into the transmit data stream on-the-fly*. This second payload is incorporated into the transmit data during very short, anyway required waiting periods in the transfer loop and hence incur no extra CPU load—it literally *comes for free*.

With the help of the sideload feature, it is easy to solve the immediate update problem: Every time a packet is received, the Rx routine marks it as the current sideload, so it gets added to the next Tx packet. If it is innovative, the Tx packet carries the innovation. If not, it does not hurt the Tx packet (by neutralizing its coding vector to zero) with high probability. If this happens anyway, the transmission is aborted before it becomes "visible in the air."

Besides innovation forwarding, the sideload feature also simplifies interlocking critical activities between the transport layer and the processing layer. Overall, it proves to be a very useful tool.

2.3.4 Efficient Run-Time Operation

Besides the features described above, we facilitate an efficient run-time operation of MIXER by:

- computing over finite field GF(2), which enables an efficient implementation on standard hardware and needs just one bit per message in the coding vectors;

- keeping the coding matrix in row echelon form, which limits the amount of memory needed for storing packets, reveals useful information for free (e.g., the rank of the matrix), and spreads computational load across multiple slots;
- parallelizing radio and CPU activities, which boosts performance by allowing to reduce the slot length to the minimum.

2.4 Implementation

We prototype MIXER on TelosB [138] devices running at 4 MHz CPU clock rate. Our implementation is publicly available at `https://mixer.nes-lab.org` and comprises about 7900 lines of C code, where 3500 lines account for the hardware abstraction layer. The compiled program has a footprint of 21 kB in flash and 300 B in RAM (w/o stack, Rx/Tx queues, matrix, history, and request masks). With payload size S_p, coding vector size $S_\mathrm{v} = \lceil M/8 \rceil$, and packet size $S = 12 + 2 \cdot S_\mathrm{v} + S_p$ (see Figure 2.4), the amount of RAM needed for the variable-sized elements can be approximated[1] as follows:

$$m[\mathrm{B}] = \underbrace{5 \cdot S}_{\text{queues}} + \underbrace{M \cdot (S_\mathrm{v}{+}S_p{+}2)}_{\text{matrix}} + \underbrace{(N{+}9) \cdot S_\mathrm{v}}_{\text{request masks}} + \underbrace{4 \cdot N}_{\text{history}} \tag{2.3}$$

Using MIXER requires choosing the slot length T_s based on the payload size S_p and the generation size M. To achieve good performance, T_s should be small. However, as visible in Figure 2.3b, T_s is lower-bounded by the minimum time needed for low-level packet transport $\tilde{T}_a$ and by the time taken by the processing layer T_p. We profiled our code to derive formulas for both bounds that make it easy to find a reasonable value for the slot length $T_s \geq \max(\tilde{T}_a, T_p)$.

The low-level packet transport time can be expressed as

$$\tilde{T}_a[\mu\mathrm{s}] = (440 + 32 \cdot S) \cdot 1.037 \tag{2.4}$$

which accounts for the packet air time (i.e., 32 µs per byte in IEEE 802.15.4 networks) and a static overhead for basic buffer handling and RF oscillator calibration. The multiplier in (2.4) matches internal tolerance settings; the chosen values compensate for clock drift of up to 1000 ppm. Further, the processing time reads as

$$T_p[\mu\mathrm{s}] = 600 + (26 + 0.155 \cdot (S_\mathrm{v} + S_p)) \cdot M + 1.8 \cdot S \tag{2.5}$$

A value determined using (2.5) ensures that the processing layer can handle the stream of packets *on average*; temporary overload is compensated by the Rx queue (see Figure 2.3).[2]

[1] Eqn. (2.3) is simplified, in particular it ignores padding bytes introduced for alignment purposes and some small internal data elements like flags.

[2] Rearranging (2.5) gives $T_p \approx 0.02M^2 + 27M + 0.16S_pM + 1.8S_p + 620$, which reveals a quadratic dependency on the generation size M. However, for realistic values of M the linear term clearly dominates due to the coefficients. This is mainly because processing happens in machine words, not bits.

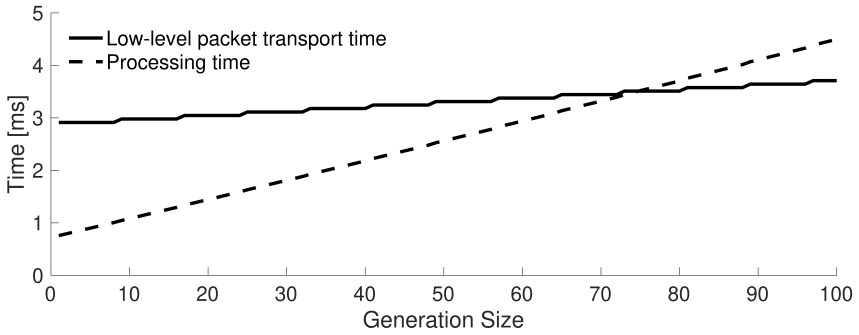


Figure 2.5: Low-level packet transport time $\tilde{T}_a$ and processing time T_p depending on the number of messages M (i.e., generation size) for a payload size of 60 B.

Figure 2.5 plots (2.4) and (2.5) against the generation size M for a payload size of 60 B. We see that the crossover point is for $M = 75$ messages. For smaller M, Mixer effectively processes packets at line rate, as the slot length is bounded by the packet air time and therefore by the bitrate of the PHY.

2.5 Evaluation

Our evaluation answers the following questions:

- How does Mixer's performance compare to the state of the art for different number of messages, message sizes, and initial message distributions (Section 2.5.1)?
- How robust is Mixer to network dynamics caused by node failures (Section 2.5.2) and node mobility (Section 2.5.3)?
- To what extent does Mixer benefit from faster CPUs and/or faster PHYs (Section 2.5.4)?

2.5.1 Performance

We compare Mixer against the state-of-the-art many-to-all solution based on sequential flooding, called S-Glossy. To implement S-Glossy, we use the original Glossy code[3], which achieves the minimum latency for flooding a single message in IEEE 802.15.4 networks [48]. Note that S-Glossy is equivalent to a data-only round in LWB [49]. It has already been shown [49] that LWB greatly outperforms routing-based many-to-many solutions [130], rendering a comparison against such schemes obsolete. Similarly, results from our work-in-progress reports show that an earlier version of Mixer outperforms Chaos for messages larger than a few bytes [113, 114]. Other RLNC-based many-to-all approaches like [36, 51] provide only theoretical or simulation results and are not applicable to practical wireless mesh networks because the assumed communication models do not fit. See our discussion in Section 2.7 for more details.

[3]The source code of the original Glossy implementation for the TelosB platform is available at `https://sourceforge.net/p/contikiprojects/code/HEAD/tree/ethz.ch/glossy/`.

Testbeds. We run experiments on two testbeds. On FlockLab, we use 27 TelosB [138] nodes sparsely deployed across one floor in an office building [99]. Indriya features 94 devices densely deployed across three floors [40]. Nodes transmit at maximum power (0 dBm) on channel 26, yielding a network diameter of 4 and 8 hops on FlockLab and Indriya, respectively. Both testbeds experience interference from devices affecting the 2.4 GHz band, such as microwaves ovens and Wi-Fi, Bluetooth, and BLE devices. Each run lasts 30–60 min.

Metrics. *Reliability* is the percentage of delivered messages (i.e., received *and* decoded in case of Mixer). *Latency* is the time from the beginning of a round until all messages are delivered (i.e., including the time needed for decoding in Mixer). *Goodput* is the number of delivered message bits per unit of time. *Radio-on time*, typically used as a proxy for energy efficiency, denotes the accumulated time the radio is on during a round. We report averages over all nodes and rounds during a run as well as 25th/75th percentiles.

Parameters. We fine-tune S-Glossy based on several test runs so it achieves the shortest possible latency at a reliability above 99.9 %. We set the slot length in Mixer as detailed in Section 2.4, and set the round length conservatively. The size of the history window is $3 \cdot N$. Since we always measured a reliability of 100 % with Mixer in the following experiments, we do not report this metric.

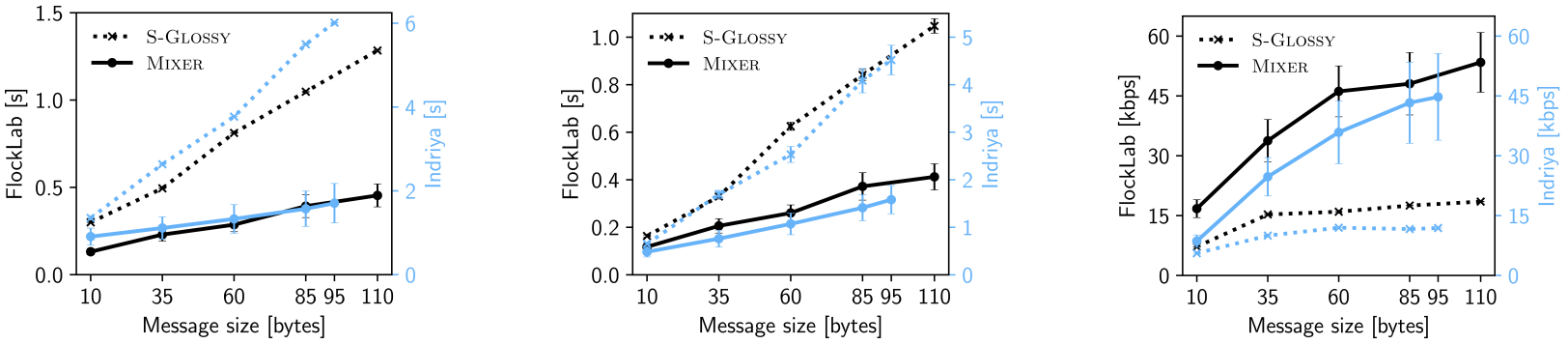
2.5.1.1 Impact of Message Size

To evaluate the impact of the message size, we run tests in which each node initially has exactly one message (all-to-all). We consider message sizes of 10, 35, 60, 85, and 95/110 bytes. Note that 95 and 110 bytes are very close to the largest message sizes that fit into an IEEE 802.15.4 packet given the overhead of Mixer's header information for 27 and 94 messages on FlockLab and Indriya, respectively. [4]

Results. Figure 2.6 plots performance of Mixer and S-Glossy against message size. We see that Mixer outperforms S-Glossy across all metrics and settings, by 2.3–2.8× on FlockLab and by 1.4–3.8× on Indriya. The main reason is that Mixer needs significantly fewer slots than S-Glossy (e.g., 95 vs. 260 on FlockLab): The combination of RLNC and ST is more efficient in terms of communication. The range of performance improvements on Indriya is wider than on FlockLab as a) with $M = 94$ messages the slot length is determined by the processing time T_p, which limits the improvement to 1.4× for small messages, and b) the larger network diameter allows for higher spatial reuse, pushing the improvement up to 3.8× for large messages.

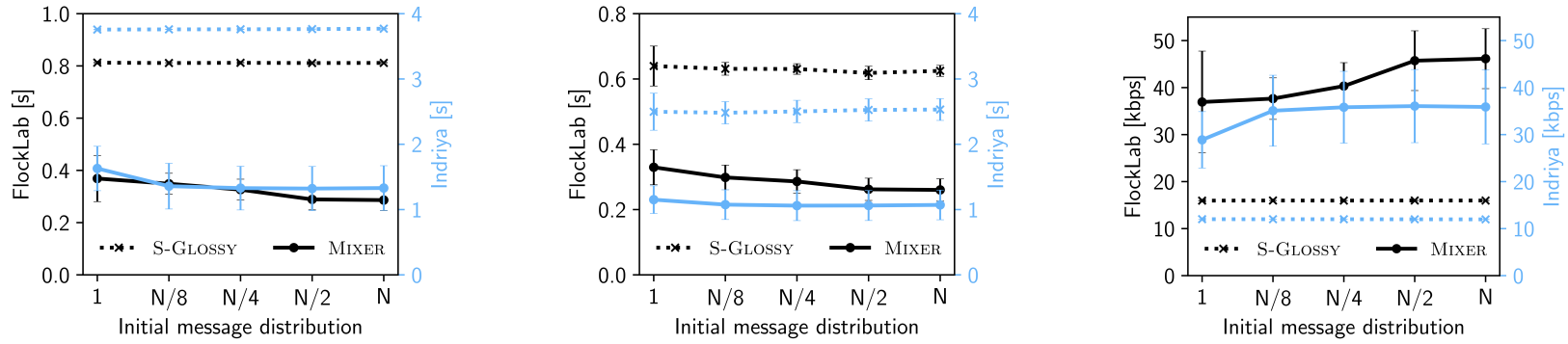
Looking at each individual metric, we find that latency, shown in Figure 2.6a, increases linearly with message size because the slot length of both primitives increases linearly,

[4] The limited RAM on the TelosB would prevent us from running all-to-all experiments with messages larger than 35 B on Indriya. Thus, for these message sizes on Indriya, nodes do not store the transmitted full-size payloads (i.e., only the coding vectors are stored), and instead perform cycle-accurate computations on a fake payload.



(a) Latency. (b) Radio-on time. (c) Goodput.

FIGURE 2.6: Performance in an all-to-all scenario on FlockLab and Indriya for different message sizes.



(a) Latency. (b) Radio-on time. (c) Goodput.

FIGURE 2.7: Performance on FlockLab and Indriya for different initial message distributions.

too. On FlockLab, Mixer's latency is 131–453 ms, a range enabling feedback control in industrial automation [4]. Thanks to Mixer's smart shutdown feature, radio-on time (Figure 2.6b) follows the same linear trend. Both primitives reach their highest goodput (Figure 2.6c) with 110 B messages on FlockLab: 18.5 kbps for S-Glossy and 53.7 kbps for Mixer.

2.5.1.2 Impact of Initial Message Distribution

Theoretical analysis shows that starting with a well-mixed distribution, where initially all messages are equally spread over the network, can boost performance compared to the case where all messages reside at the same node [60]. To evaluate this aspect, we fix the message size (60 B) and the number of messages M (27 on FlockLab, 94 on Indriya), and vary the fraction of the N nodes that initially holds the M messages: N (all-to-all), $N/2$, $N/4$, $N/8$, and 1 (one-to-all). Messages are equally distributed across the respective source nodes.

Results. Our results in Figure 2.7 confirm that Mixer benefits from a well-mixed initial message distribution, while performance with S-Glossy is unaffected. The effect becomes noticeable when messages are pooled at $N/4$ or fewer nodes, yet the performance loss is at most 22 % compared to the all-to-all case. We attribute this behavior especially to Mixer's improved startup phase (Section 2.3.2.3), which ensures fast-growing coding potential even in one-to-all scenarios.

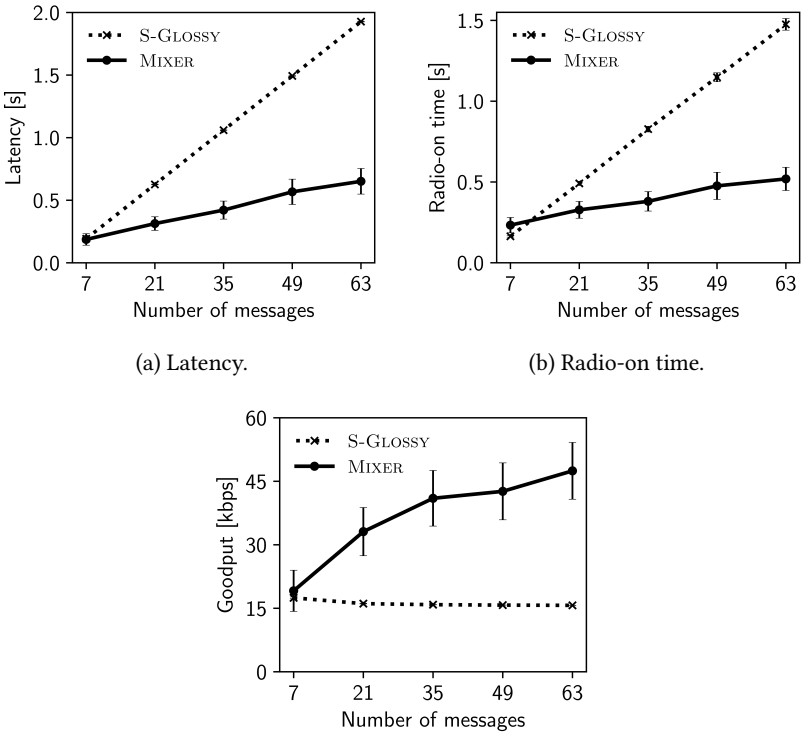
2.5.1.3 Impact of Number of Messages (Generation Size)

In a final set of performance experiments, we investigate the impact of the number of messages M to be exchanged in a round (generation size). We use again a message size of 60 B and equally distribute 7, 21, 35, 49, and 63 messages to seven source nodes on FlockLab.[5]

Results. Looking at Figure 2.8, we see that Mixer only has a performance advantage over S-Glossy if there are at least a handful of messages to be exchanged. Otherwise, the coding potential is too small and fine-tuned sequential Glossy floods perform better. Nevertheless, Mixer's performance advantage grows quickly with the number of messages: Mixer is 2× faster and more efficient than S-Glossy for 21 messages, and already 3× better for 63 messages.

Interestingly, we find in every *all-to-all* experiment that Mixer needs on average about $3 \cdot M$ slots to deliver M messages despite vastly different payload sizes, network diameters, and node densities on the two testbeds. This gives an idea of the constants hidden by the O-notation in the order-optimal scaling $O(M+T)$ of RLNC-based gossip for our Mixer implementation. Indeed, for small generation sizes M, the number of slots T needed to disseminate one message dominates, whereas M dominates for large generation sizes. T is determined by the diameter of the network.

[5] We only show result from FlockLab because the number of active nodes on Indriya changed significantly during our experiments so that the results would not be comparable to those in Figs. 2.6 and 2.7.



(a) Latency.

(b) Radio-on time.

(c) Goodput.

Figure 2.8: Performance on FlockLab for different number of messages equally distributed to seven source nodes.

2.5.2 Network Dynamics: Node Failures

In practice, nodes can suddenly drop out due to disconnection or failure. While the absence of a node for several rounds is handled by a higher-layer protocol, Mixer must cope with situations where nodes disappear shortly before or during a round.

To study this aspect, we run experiments on FlockLab in which we let a given number of nodes simultaneously fail. We consider the failure of 1, 2, and 5 nodes and vary the slot in which the failure occurs from 1 to 50. For each combination we perform at least 100 rounds. Nodes exchange 10 B messages in an all-to-all fashion, and non-failing nodes log after each round how many of the 27 messages they can successfully decode.

Results. Figure 2.9 plots the probability that a non-failing node decodes all messages against the slot in which the failure occurs. We see that a failure before or at the very beginning of a round prevents the non-failing nodes from decoding all messages. This is because the nodes fail before they can transmit for the first time. Thus, a failure in later slots increases the probability that the non-failing nodes can decode all messages. The increase shifts to the right with more failing nodes as nodes also need to transmit linearly independent packets. Beyond a certain point (e.g., slot 20 for 1 failing node), the probability to decode all messages is close to 100 %.

Note that, barring packet losses due to other reasons, a node *for sure* decodes all

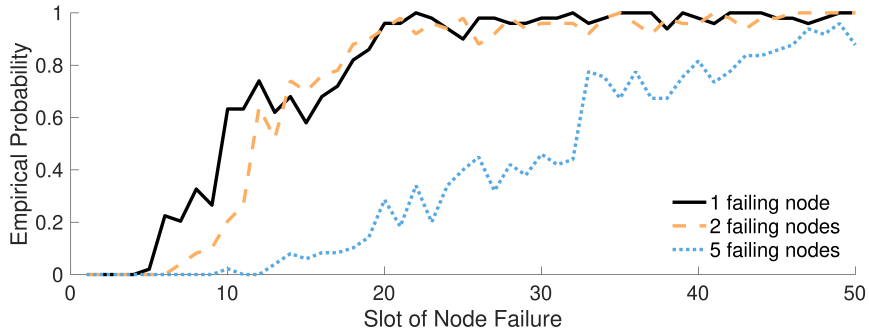


FIGURE 2.9: Probability that nodes decode all messages against the slot in which a set of nodes concurrently fails.

messages that initially resided at non-failing nodes. For instance, in our runs each non-failing node always decodes at least 26, 25, and 22 messages for 1, 2, and 5 failing nodes, respectively. We can therefore conclude that MIXER is highly robust to node failures, providing guaranteed service to non-failing nodes while salvaging messages of failing nodes with high probability.

2.5.3 Network Dynamics: Node Mobility

Emerging applications increasingly rely on nodes attached to mobile entities [86, 140]. We investigate MIXER's resilience against the resulting network dynamics in an outdoor experiment.

Scenario. We deploy 14 battery-powered TelosB nodes on cardboard boxes in a 80 m by 230 m area as illustrated in Figure 2.10. Another node is mounted on a car and attached over USB to a laptop. Nodes transmit with 0 dBm on channel 26, using MIXER to periodically exchange 15 28 B messages in an all-to-all fashion. The messages contain performance counters and the IDs of all nodes from which the nodes have *directly* received a packet during the previous MIXER round (i.e., their 1-hop neighbors). We set the slot length to 2 ms according to the guidelines in Section 2.4. Nodes initiate a round every 500 ms, while the round length is 150 slots. At the end of each round, the node on the car uses the remaining 200 ms to log the messages it received over USB before the next round begins.

We first measure for 10 min with the car standing next to location A (see Figure 2.10). Then we measure for 10 min while performing different maneuvers with the car. We repeatedly pass locations A→B→C→D, then make a turn at location E, and repeatedly pass locations D→C→B→A. We drive with a speed of 20–40 km/h while going in circles, and hit 60 km/h between locations E and C. Such speeds are typical of state-of-the-art mini and micro drones [31].

Results. Figure 2.11 shows the 1-hop neighbors of the node on the driving car over time. We can see that the mobile node cannot directly communicate with all other nodes: It has different sets of neighbors depending on its location. Indeed, we recognize

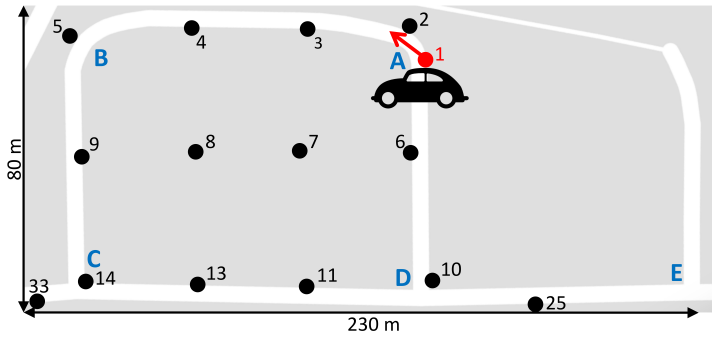


FIGURE 2.10: Setup of outdoor experiment with a mobile node mounted on a car that drives at a speed of 20–60 km/h.

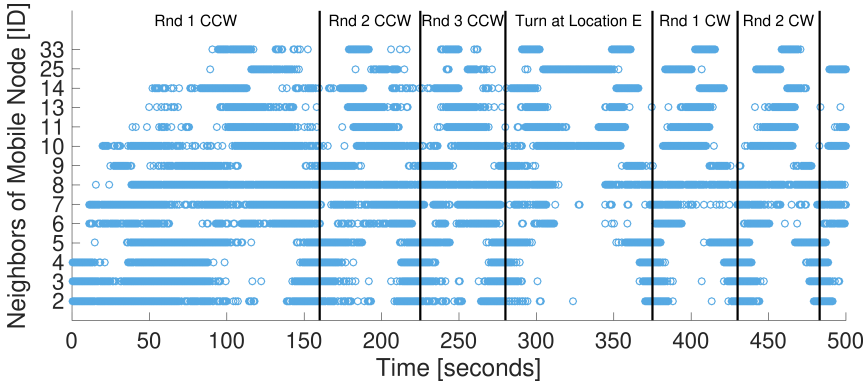


FIGURE 2.11: 1-hop neighbors of mobile node as the car drives rounds counterclockwise (CCW) and clockwise (CW).

a recurring pattern that allows us to infer how often the car drove around the circle. For example, before completing the fourth round counterclockwise (CCW), the car drove up to location E, leaving the mobile node with only two neighors and increasing the network diameter to at least three hops. Then, the car made a turn and continued to drive the circle clockwise (CW) three more times.

Despite heavy network dynamics due to the high speed of movement, we measure the same performance compared to when the car was standing. As visible from Table 2.2, MIXER consistently provides high reliability >99.99 % and low latency ≤98.4 ms. For shorter messages (e.g., a few bytes carrying GPS data), one could further reduce the slot length to 1 ms. As a result, latency in this scenario would reduce to about 50 ms, which is sufficient for drone swarm coordination requiring all-to-all communication every 100 ms [30, 140]. In summary, the results show that MIXER is highly robust to network dynamics and satisfies the demands of emerging applications.

2.5.4 Potential of Faster CPUs and Physical Layers

MIXER can benefit from a more powerful CPU: Using the same PHY, a faster CPU allows MIXER to process packets at line rate for a wider range of payload sizes and number of messages. For example, 2× faster processing shifts the crossover point

Table 2.2: Mixer's performance with and without mobility.

Performance Metric	With Mobility	Without Mobility
Reliability [%]	>99.99	>99.99
Latency [ms]	96.2	98.4

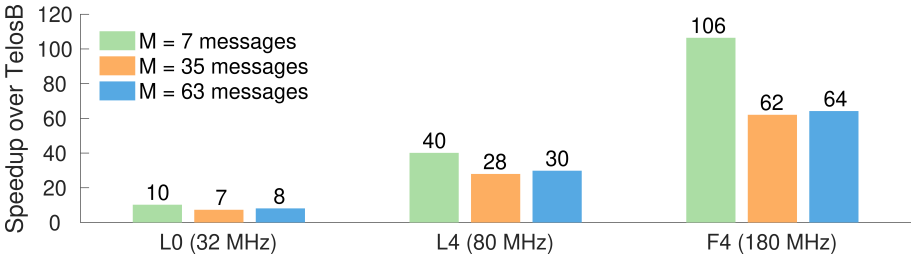


Figure 2.12: Speedup of ARM cores over TelosB in processing received packets for different generation sizes.

in Figure 2.5 from 75 to 215 messages. Conversely, to fully exploit a faster PHY, the processing speed should increase as well.

CPU cores. We perform microbenchmarks on the TelosB and three different 32-bit ARM cores: Cortex-M0+ running at 32 MHz (labeled L0), Cortex-M4 running at 80 MHz (L4), and Cortex-M4 running at 180 MHz (F4). Despite higher clock speeds, L4 and F4 offer extended instruction sets and richer hardware capabilities than the L0.

Processing speedup. To measure the processing speedup, we port Mixer's packet processing routines to the ARM cores. We profile the time needed to process received packets (building transmit packets involves similar operations, merely with lower variance). To obtain real-world execution times for the same input on all cores, we run tests on FlockLab with 60 B messages for different generation sizes ($M = \{7, 35, 63\}$) and trace the sequence of received packets at each node. We feed the collected traces into the four cores to get a total of 40,000 execution time measurements per core.

The speedups, shown in Figure 2.12, are 7–10×, 28–40×, and 62–106× for the L0, L4, and F4, respectively. The speedup depends on the generation size M, because Mixer processes payloads in batches and delayed from coding vectors. This optimization becomes more important with higher average batch size and thus with more messages M. We use a highly optimized implementation of this approach on the TelosB, and expect that similar ARM-specific code optimizations reduce the differences in speedups for different M.

Projected benefits. We use (2.4) and (2.5) to project the impact of a faster CPU and/or a faster PHY on Mixer's performance. Specifically, we multiply $\tilde{T}_a$ with the speedup in PHY bitrate and T_p with the processing speedup over the TelosB. Using the speedups from Figure 2.12, we assume that all processing activites in Mixer benefit as much as

Table 2.3: Projected latency of Mixer on FlockLab (all-to-all, 60 B messages) for different PHY bitrates and CPU cores. Latencies in italics are CPU-bound (i.e., core is underdimensioned).

PHY Bitrate [Mbps]	TelosB [ms]	L0 [ms]	L4 [ms]	F4 [ms]
0.25	295.6	295.6	295.6	295.6
1	*161.8*	73.9	73.9	73.9
11	*161.8*	*23.1*	6.7	6.7
54	*161.8*	*23.1*	*5.8*	*2.6*

the processing of received packets. In this way, we can check, for example, whether a CPU core is under-, over-, or well-dimensioned for a given PHY bitrate.

As an example, consider an all-to-all scenario with 60 B messages on FlockLab. We know from Section 2.5.1 that Mixer needs about 95 slots with IEEE 802.15.4, and studies suggest that capture (and hence Mixer) works comparable or better with other PHYs [118]. Table 2.3 lists projected average latency in milliseconds for 16 PHY/CPU combinations, using the *minimum* speedups for each core from Figure 2.12. We see, for example, that the L0 is sufficient to fully leverage a PHY bitrate of 1 Mbps used by BLE, while the L4 or the F4 is needed to match 11 or 54 Mbps of IEEE 802.11 variants. The latter combination (F4, 54 Mbps) would reduce latency by 100× compared to the TelosB.

2.6 Discussion

Larger finite fields. Mixer currently uses GF(2), which keeps the coding vectors small and allows for a straightforward and efficient implementation on standard hardware. Instead, the network coding literature favors larger finite fields, such as GF(2^8) [51], to increase the chances that a received packet is innovative. We studied the impact of larger finite fields on Mixer's performance in simulation and found that the gains are smaller than one may expect: Using GF(2^2) reduces the average number of slots by about 10 % compared with GF(2), but GF($2^3 \ldots 2^8$) does not provide further improvements. We attribute this to the fact that the spreading of innovation in an area (as promoted by larger finite fields) is upper-bounded by the influx of messages into that area. It is also questionable whether the fewer slots with larger finite fields can indeed translate into shorter latencies in practice as the computational load and the size of the coding vectors would increase by several orders of magnitude.

Robustness to interference. Mixer achieves nearly perfect reliability in almost all our experiments conducted under typical wireless interference in office buildings. Nevertheless, it would be possible to borrow standard techniques such as frequency hopping from other technologies (e.g., Bluetooth) to make Mixer even more robust to interference. To this end, the slot number may serve as an index into a pseudo-random sequence of channel frequencies that is known to all nodes. One may also increase the

slot length T_s to tolerate interference bursts (at the cost of higher latency), or adapt T_s in a pseudo-random fashion to evade systematic jamming.

Setting the length of a round. Mixer provides a parameter that specifies the number of slots constituting a round. Together with the slot length T_s this parameter defines the nominal length of a round and makes the running time of Mixer predictable, in addition to the timeout mentioned in Section 2.3.3.1. However, one limitation of Mixer is that the number of slots required until all nodes have reached full rank is difficult to predict. While some applications allow for using a conservative estimate, this approach may be problematic for applications with critical time constraints. Theoretical works have looked at the worst-case number of slots for different network and communication models [36, 51, 60, 129]. It would be worthwhile to adapt these models to Mixer (e.g., by incorporating existing capture models [83, 182]) to determine safe bounds on the length of a round. Our experiments suggest that useful predictions are within reach: We observe in all all-to-all runs that Mixer needs on average about $3 \cdot M$ slots despite different network topologies and payload sizes.

2.7 Related Work

Theoretical foundations. Ahlswede et al. introduced network coding, showing that it achieves the multicast capacity of wireline networks [3]. It was later found that these bounds can be achieved using linear codes, and that encoding and decoding can be done in polynomial time [84, 97]. This also holds if nodes pick random coefficients [70]. These works form the theoretical foundation of RLNC, which we combine in Mixer with the following technique.

Synchronous transmissions. Mixer exploits simultaneous transmissions from multiple senders. SourceSync is the first system that demonstrates the benefits of multiple senders transmitting the *same* packet in real IEEE 802.11 networks [143]. Glossy uses this concept for fast and reliable flooding in multi-hop IEEE 802.15.4 networks [48]. These protocols rely on accurate symbol-level synchronization to benefit from sender diversity. In Mixer, instead, nodes transmit *different* packets, which relaxes the required synchronization to the length of the preamble to possibly receive one of the transmitted packets due to the capture effect [93, 94, 189]. The capture effect has been used for collision resolution [181], network flooding [102], aggregation [91], and agreement [5]. Instead, Mixer exploits the capture effect for efficient many-to-all broadcasting of sizable messages.

Practical wireless network coding. Network coding has been extensively studied in wireless and sensor networks; however, the vast majority of works focuses on theoretical gains or evaluates new protocol designs only in simulation (see [134] for a recent survey), thereby ignoring many practical issues that complicate or even prevent a real implementation.

COPE [80] and MORE [29] are the first implementations of network coding for multiple unicast flows and a single multicast flow in IEEE 802.11 networks. Pacifier [87] achieves a higher multicast throughput than MORE. These works target stationary networks and specific traffic patterns, which allows them to leverage long-lived network and routing state for coding and packet forwarding. This, however, makes them unfit for dynamic networks and concurrent multicast flows, both of which Mixer readily supports. Moreover, they target PC-class devices with plenty of compute power, memory, wireless bandwidth, and energy. Mixer can cope with stringent constraints on any of these resources, allowing low-power wireless systems to benefit from network coding without putting restrictions on the traffic pattern.

Splash [41] and Pando [42] integrate pipelined flooding with XOR and fountain coding for one-to-all data dissemination in IEEE 802.15.4 networks. While these solutions run on resource-constrained devices, only the source encodes packets; all other nodes forward the encoded packets and decode, which simplifies design and implementation. They also assume stationary networks and support only a single source node. Instead, Mixer supports dynamic networks, any number of sources, and efficiently performs forwarding, en-/recoding, and decoding at every node in the network. As a result, Mixer performs comparable or better to these specialized protocols, and yet supports a much broader range of scenarios.

Many-to-all broadcasting. The unstructured spreading of messages in Mixer is reminiscent of gossip [78]. Deb and Médard showed that combining gossip with RLNC for the dissemination of multiple messages outperforms any non-coding approach (in terms of needed slots) in a specific communication scenario [36]. More precisely, they consider a *random phone call model* where the underlying network graph is complete. Later works study variants of this approach theoretically [129] and in simulation [51], primarily on static networks. However, the underlying network model does not fit wireless mesh networks because a) the random phone call model implies that all links work independently (i.e., there is no interference); b) it is assumed that a node is able to receive multiple packets simultaneously or can perfectly avoid collisions. Further, the results rely on assumptions regarding the initial message distribution and the field size used for network coding. In particular, [36, 129] consider cases where the field size q has been chosen such that $q \geq M$ while the simulations in [51] use GF(2^8). With packet size constraints as in IEEE 802.15.4, such field sizes can lead to significant limitations and performance degradation as discussed in Section 2.6.

Recently, it was shown that RLNC-based gossip achieves the optimal scaling $O(M+T)$ also in dynamic networks for any initial message distribution and field size [60]. Furthermore, [60] provides results for a *broadcast model* that fits much better to the inherent nature of wireless networks. However, the analysis is purely theoretical and still assumes that nodes are able to receive multiple packets simultaneously. As for dynamic wireless mesh networks, Mixer is the first design that translates the projected benefits from theory into practice by combining RLNC with ST.

Concurrently to our work, Mohammad and Chan [127] proposed Codecast, combining LT codes [103] with ST. LT codes can be interpreted as a special variant of RLNC. However, they lack the recoding capabilities of generic RLNC: In Codecast, a node is not able to recode arbitrary payloads, it can only (re-)encode previously decoded messages. Thus, the coding potential at the nodes grows slower than it does with Mixer, eventually leading to longer rounds. Indeed, the results in [127] suggest that Mixer outperforms Codecast by up to 3× on FlockLab. Further, the design of Codecast suffers from severe scalability issues for more than $M = 30$ messages.

Postscript

With Mixer, we have contributed an efficient, scalable, and reliable many-to-all communication primitive for wireless mesh networks. Moreover, Mixer's versatility supports the full spectrum from one-to-all to all-to-all communication and from static to dynamic networks, including fast mobile agents. Compared to prior practical many-to-all protocols that rely on message routing or flooding, Mixer exploits the synergy of RLNC and ST to disseminate all messages simultaneously. Our implementation of Mixer is up to 3.8× faster and more efficient than the state of the art while providing nearly 100 % reliability. Thus, Mixer empowers emerging wireless CPS, as we demonstrate in Chapter 6, and enables applications that have seemed out of reach.

3

Butler: Increasing the Availability of Low-Power Wireless Communication Protocols

Preface

Dependability is undoubtedly one of the most important requirements of CPS applications, especially in medical and industrial scenarios. Our communication protocol Mixer, presented in the previous chapter, already provides a very reliable communication service. However, Mixer and *all* other ST-based communication protocols contain a single point of failure that fundamentally impairs the availability of the communication service in the presence of node crashes and network partitions. The problem is that these protocols require tight time synchronization and use one dedicated node for this task. Distributing this task across multiple nodes comes with various challenges.

This chapter addresses this problem and proposes Butler, a lightweight and distributed time synchronization mechanism. Butler synchronizes all nodes in the network, regardless of node failures, so any set of nodes can start the communication process, effectively eliminating the single point of failure. In addition, we also formally prove Butler's correct synchronization behavior. The experimental evaluation shows that Butler can reliably synchronize the network to within ±3 µs despite unpredictable node failures and network partitions. Experiments with Mixer *and* Butler demonstrate a truly fault-tolerant communication service. Moreover, Butler has no noticeable impact on the overall communication performance, and its temporal overhead ranges well below 1 %. Thus, the efficiency and effectiveness of Butler can significantly improve the availability of existing ST-based protocols to meet the strict dependability requirements of CPS applications.

3.1 Introduction

In recent years, wireless sensor networks (WSNs) have become an integral part of CPS and the Industrial Internet of Things with applications ranging from personalized medicine through infrastructure control to smart factories. WSNs offer unprecedented flexibility and cost efficiency in terms of installation, operation, and maintenance compared to wired communication systems [17, 88, 147]. However, as defined by the International Society of Automation, the *dependability* of the wireless communication service is essential in these critical application domains, where small disruptions can cause system outages involving huge financial losses or even catastrophic consequences [190].

In parallel to advances in hardware enabling more capable yet ultra-low-power microcontrollers, communication protocols have also evolved. With Glossy [48], ST became popular, and many different ST-based protocols emerged in the following years that greatly outperform the traditional link-based protocols [196]. For example, in the EWSN Dependability Competition [157], which attracted participants from industry and academia, teams with ST-based protocols consistently placed in the top three ranks. These new protocols can satisfy higher application requirements and make WSNs suitable even for demanding closed-loop control applications (e.g., [172], and our work later in Chapter 4). One of the key advantages of many ST-based protocols is their topology-independent protocol logic, which provides unprecedented resilience and flexibility as required, for example, in highly dynamic application scenarios with mobile robots [17] or drone swarms [61].

Problem. From an application's perspective, a dependable communication service should transport messages reliably across the network *and* be available when needed to ensure efficient and timely message delivery. State-of-the-art ST-based protocols [196] provide a highly reliable and efficient message transport. Furthermore, protocols like Virtus [50] and Wireless Paxos [137] provide mechanisms (virtual synchrony and consensus) to build higher-layer fault-tolerant systems.

Despite these achievements, *these protocols are themselves not fault-tolerant*: already the failure of a single node can lead to the unavailability of the communication service in the entire network. The fundamental problem is that ST-based protocols require tight time synchronization, which is achieved by selecting a particular node, often named *initiator* (see, e.g., [48, 91], and our Mixer protocol in Chapter 2), that provides a time reference. Typical faults in WSN deployments (e.g., due to software/hardware failures, fabrication problems, environmental factors, adversarial attacks, and battery depletion [77]) can cause the initiator and hence the communication service to fail. Moreover, the single initiator is also a problem when the network splits into different partitions, for example, because the node connectivity is affected by environmental factors such as obstacles or interference and by moving nodes (e.g., a swarm of drones splitting up in flight). In these situations, the single initiator is only part of one partition, and the nodes in all other partitions are no longer able to exchange any messages.

Contribution. To solve the availability problem, we present the design, theoretical analysis, and experimental validation of Butler, a lightweight and distributed synchronization mechanism. Butler enables ST-based protocols to distribute the role of the initiator across multiple nodes (randomly selected at run-time) so that all non-faulty nodes that are physically able to communicate can do so at the required time. To achieve this, Butler uses a fully distributed and highly efficient mechanism where nodes probabilistically propose and distribute reference times using short messages that feature a natural order. This order allows nodes to quickly converge toward the same accurate reference time despite possible node failures, message losses, and network partitions. Afterward, communication can be initiated in a dependable way by multiple *synchronized* initiators, which is key to leveraging the efficiency and reliability of ST.

After describing the design of Butler in Section 3.3, we formally analyze and prove its correctness in Section 3.4. Section 3.5 presents an open-source implementation of Butler on the popular nRF52840 platform, which we use in Section 3.6 to evaluate performance and efficiency on the FlockLab testbed [173]. Our results demonstrate that Butler reliably synchronizes all nodes in the network to within ±3 µs despite large initial time offsets and unpredictable node failures. Butler achieves this while incurring only a minimal temporal overhead that ranges below 1 % in realistic scenarios. Moreover, experiments with our ST-based communication protocol Mixer (Chapter 2) show that the communication performance in terms of latency and reliability significantly decreases when using multiple initiators. When instead extending the standard Mixer protocol with Butler to synchronize the initiators, our results indicate no performance degradation: latency and reliability are at least as good as for the original Mixer with a single initiator while providing superior availability.

In summary, this work contributes the following:

- The design of Butler, a lightweight and distributed synchronization mechanism that pushes the availability of ST-based protocols to previously unseen heights.
- A rigorous theoretical analysis of Butler, including a formal proof of Butler's correctness.
- Real-world experiments that validate the theoretical analysis by demonstrating outstanding synchronization accuracy at minimal temporal overhead despite node failures.
- A case study demonstrating that Butler increases the availability of a state-of-the-art ST-based protocol without sacrificing overall communication performance.

3.2 Motivation and Background

Providing high availability in WSNs is an important yet unsolved research problem. The problem originates from the dependability requirements of emerging CPS applications. We discuss these requirements next, then review previous approaches toward providing dependability in WSNs, and, finally, state the problem. Section 3.7 discusses existing work that is most closely related to our specific contributions.

Application requirements and fault model. WSNs offer high flexibility and cost efficiency, making them a key building block for many current and future CPS, including mission- and safety-critical applications [17]. In addition to high performance demands, these applications require a dependable communication service that enables reliable data exchange and is available when needed despite certain failures [190].

We define a communication service to be *available* if messages can be exchanged at the required time (i.e., as requested by the application or a high-layer protocol) between all nodes in the network that are physically able to communicate with each other (i.e., when the signal-to-interference-plus-noise ratio between these nodes is high enough to permit information transfer). Providing availability thus requires robustness and fault tolerance such that failures of individual nodes do not affect the availability of the communication service between the non-faulty nodes.

WSNs are deployed at scale and consist of many low-cost, resource-constrained embedded devices that can fail for various reasons [45]. For example, because of software and hardware faults or depleted batteries, nodes may suddenly stop working (i.e., fail-stop). Nodes may also recover from a failure and resume operation (i.e., crash recovery). In addition, the environment of the deployment also has a significant impact on the network. Due to obstacles, external interference, and node movement, the communication links are constantly changing, which leads to time-varying message losses and may split the network into several isolated partitions. While we consider all aforementioned types of failures (i.e., node crashes, message losses, and network partitions), we do not consider Byzantine (i.e., erratic or malicious) faults. That is, we assume that a node works according to its specification whenever it is operational, and modifications to messages during transmission can be reliably detected (e.g., using error detection codes such as a cyclic redundancy check (CRC)).

Dependability in wireless sensor networks. Early WSN protocols adopted link-oriented and routing-based communication techniques [56] to meet the requirements of uncritical applications (e.g., environmental monitoring [168]). Yet, already in 2003, Stankovic et al. noted the importance of real-time and dependability guarantees in WSNs to meet the requirements of more demanding applications, such as those involving control [164].

Motivated by the growing importance of such CPS applications, recent WSN protocols based on ST have been shown to provide real-time guarantees [195] and certain dependability properties. For example, Virtus [50] provides atomic multicast and view

management, while A^2 [5] and Wireless Paxos [137] provide distributed agreement and consensus. These dependability properties are fundamental to building fault-tolerant systems through redundancy and replication [156]. Providing these properties has been possible since ST allow the protocol logic to abstract away the complexity and dynamics of wireless networks. For instance, the temporary or long-term failure of individual wireless links can be smoothly handled by the spatio-temporal diversity of the ST technique [196].

Problem statement. However, ST-based protocols cannot deal with the failure of critical nodes, such as the node that initiates the communication process. Wireless Paxos, Virtus, A^2, Mixer, and many other ST-based protocols (e.g., [48, 91]) rely on one dedicated initiator that starts a packet exchange by transmitting first. This single initiator is a serious threat to the protocols' availability: a failure of this node prevents *any* communication in the network as all other nodes will keep waiting for an incoming transmission event that never occurs.

The reason for the single initiator is the need for an accurate time reference. ST require tight synchronization across nodes which must minimally satisfy the constraints set by the capture effect (e.g., 160 µs for IEEE 802.15.4) known as *capture window* [91]. If the time offsets of nodes exceed the capture window, communication becomes inefficient and unreliable.

To solve the availability problem, an ST-based protocol should ideally use a large set of multiple initiators, randomly and independently selected at run-time before every individual packet exchange. The problem, however, is that the nodes, and therefore the set of potential initiators for the next packet exchange, quickly get out of sync due to the inevitable clock drift between nodes. For example, the IEEE 802.15.4 standard [71] requires a clock drift of at most ±40 ppm, which means that in the worst case, two initially perfectly synchronized nodes violate the capture window already after 2 s.

Based on these requirements, a scheme is needed that synchronizes the nodes to within the size of the capture window. Moreover, an effective synchronization mechanism must itself be fault-tolerant under the above-mentioned fault model, and be lightweight (i.e., low overhead) to avoid negatively affecting the overall communication performance.

3.3 Design

We introduce Butler, a synchronization mechanism that solves the problem outlined above to boost availability. Before presenting the details of Butler's design, we provide a high-level overview and state the scope of our work.

3.3.1 Butler Overview

Butler is a lightweight and distributed synchronization mechanism that is designed to directly integrate with existing communication protocols. It does not require periodic

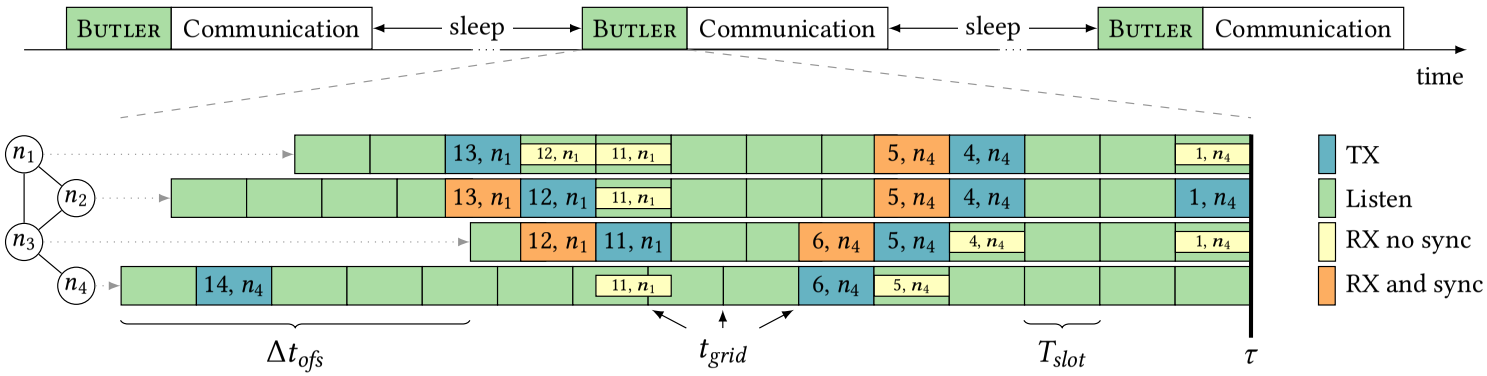


FIGURE 3.1: BUTLER is executed right before scheduled communication to synchronize the network. This removes the need for a single initiator in ST-based communication protocols, which increases their availability. The example shows how 4 nodes (n_1 - n_4) with a maximum initial offset of Δt_{ofs} synchronize their local slot grids. The contents of the sync messages (T_τ and σ) are shown inside the slots, whereas T_τ is expressed in terms of the remaining number of slots until BUTLER terminates. Initially, n_1 - n_3 synchronize on the proposed reference time of n_1, but re-synchronize upon reception of the earlier reference time from n_4.

activity and is executed right before a scheduled communication round, as illustrated in the upper part of Figure 3.1. Butler reliably establishes a common reference time τ among nodes despite node failures, message losses, and network partitions such that multiple initiators can start the communication protocol at the same time.

Because of clock drift, the nodes are initially unsynchronized and start Butler with different time offsets t_{ofs}, as shown in the lower part of Figure 3.1. We assume a maximum initial offset Δt_{ofs} between the nodes, which can be calculated based on the current communication period and the known maximum clock drift. During the execution of Butler, the nodes probabilistically exchange *sync messages*. Each sync message is associated with a certain reference time. Using Butler, the nodes always synchronize to the earliest (minimum) reference time and propagate this reference time further in the network. Eventually, all nodes have the same reference time, which marks the end of Butler.

Scope. Butler is just one important piece of the puzzle to achieve high availability of the overall system. Specifically, Butler is meant to improve the availability of ST-based communication protocols that are responsible for the message exchange among nodes (Mixer (Chapter 2), Chaos [91], and others [196]). These protocols, sometimes also referred to as *communication primitives*, can benefit from Butler if they are in principle able to support multiple initiators—any protocol that does not fulfill this requirement cannot provide availability! The necessity for multiple initiators rules out protocols that can only realize a one-to-all message exchange in each execution (e.g., Glossy [48]) since there can only be one specific source node by design. Moreover, all protocols that build upon such protocols, for example, the LWB [49] or Crystal [73], are therefore inherently limited and cannot be made available with Butler. Higher-layer WSN protocols that also perform network management tasks at one dedicated node (e.g., scheduling) require additional mechanisms to provide availability by avoiding this single point of failure.

3.3.2 Butler in Detail

In the following, we will explain the structure and operation of Butler. Additionally, we will discuss certain design decisions and their impact.

Slot grid and sync message. In Butler, each node follows a local slot grid, as shown in Figure 3.1, which is defined by the grid reference t_{grid} and the slot length T_{slot}. While T_{slot} is fixed and known to all nodes, each node uses t_{ofs}, the time at which it started Butler, as their initial t_{grid}. During the execution, nodes exchange sync messages to align their local slot grids and determine a common τ. Because the nodes align transmissions to their local slot grid, the receiving nodes will know the grid reference $\hat{t}_{grid}$ of the sender based on the receive timestamp. Each sync message is associated with a certain τ and contains two pieces of information: The duration T_τ, which is the time from $\hat{t}_{grid}$ until τ, and origin σ, the ID of the node that sent the particular τ for the first time. Note that sync messages with the same σ always describe the same τ.

Algorithm 2 BUTLER (core algorithm)

```
1:  procedure BUTLER_START ( )
2:      T_τ ← slots * T_slot                          ▷ Remaining duration
3:      t_grid ← t_ofs                                ▷ Slot grid reference
4:      τ ← t_grid + T_τ                              ▷ Reference time
5:      σ, tx ← 0

6:      while T_τ > 0 do          ▷ Each loop iteration corresponds to one local slot
7:          if tx or (P_TX > RANDOM()) then           ▷ Make transmit decision
8:              if σ = 0 then
9:                  σ ← n_i
10:             TRANSMIT(T_τ − T_slot, σ)             ▷ Transmission starts at t_grid + T_slot
11:             tx ← 0
12:         else
13:             if RECEIVE() ≠ 0 then
14:                 τ̂ = t̂_grid + T̂_τ                  ▷ Received reference time
15:                 if (σ = 0) or (τ̂ < τ) then
16:                     SYNC ()
17:                 else if (τ̂ = τ) and (σ̂ < σ) then
18:                     SYNC ()
19:         t_grid ← t_grid + T_slot
20:         T_τ ← T_τ − T_slot

21: procedure SYNC ( )
22:     σ, T_τ, t_grid, τ ← σ̂, T̂_τ, t̂_grid, τ̂
23:     tx ← 1                                        ▷ Relay new τ in the next slot
```

We use the size of a sync message to determine T_{slot}, because we want transmissions with the same σ to either overlap completely or not at all.

Operation. We explain BUTLER's operation based on the example in Figure 3.1 and directly refer to the relevant lines in Algorithm 2. At the beginning, the nodes initialize the remaining BUTLER duration T_τ (i.e., the time until τ) as a multiple of T_{slot} (line 2). The grid reference t_{grid} and T_τ are used to determine the initial τ (line 4). Furthermore, all nodes start unsynchronized and set σ to 0 (line 5). BUTLER's main loop (line 6) is executed once per slot, at the end of which t_{grid} and T_τ are updated accordingly (lines 19-20). At $T_\tau = 0$, τ is reached and BUTLER terminates. In each slot, the nodes decide independently whether to transmit a sync message or listen. We start with the transmit decision and explain reception afterward.

The transmit decision is made with probability P_{TX} (line 7). If $\sigma = 0$, the node is still unsynchronized and will propose its own τ to the network, setting σ to its node ID (lines 8-9). The node then aligns the start of the transmission to the next slot ($t_{grid}+T_{slot}$) and sets the duration in the sync message to $T_\tau - T_{slot}$ (line 10). This is the case for the first transmission of n_1 and n_4 in Figure 3.1.

In case a node decides not to transmit, it will listen for an incoming sync message. After a successful reception (line 13), the node computes the reference time $\hat{\tau}$ associated with the sync message (line 14). If $\sigma = 0$ or the received $\hat{\tau}$ is earlier than τ, the node synchronizes to $\hat{\tau}$ (lines 15-16). The rationale behind this is that the earliest reference time had the most time to propagate through the network and therefore is expected to reach the most nodes compared to other reference times. In Figure 3.1, n_2 and n_3 synchronize to their first reception (RX and sync) because of $\sigma = 0$ and later re-synchronize due to $\hat{\tau} < \tau$. Synchronizing to a new reference time involves updating the local information with the information from the sync message and adjusting the local slot grid (line 22). To quickly spread the new τ, the node will always transmit in the next slot (line 23). In contrast, n_4 discards the first reception (RX no sync) because it is synchronized to an earlier reference time, hence the local slot grid remains unchanged as can be seen in Figure 3.1. In the unlikely case that both $\hat{\tau}$ and τ are equal, the node synchronizes to the reference time of the sync message with the lower σ (lines 17-18).

Using Algorithm 2, Butler aligns the local slot grids of initially unsynchronized nodes to within T_{cap}, so that multiple nodes can reliably initiate the communication process at the same time in the subsequent communication round.

3.4 Analysis

After describing the design of Butler, we now theoretically analyze its synchronization behavior.

System model. We consider a system consisting of a set $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ of N embedded devices (nodes). Each node n_i has a local clock that runs at a specific clock speed v_i, which may vary from node to node due to imperfect clock sources (e.g., a crystal oscillator). The nodes have unique IDs and are equipped with half-duplex RF transceivers to transmit and receive messages wirelessly. Communication over the shared wireless medium is unreliable, and the probability of successful packet reception is always below 1. Moreover, we assume that nodes do not have access to external synchronization sources such as GPS and must exclusively synchronize via communication. In general, multi-hop communication is needed to reach all nodes in the network because of the limited communication range. Due to environmental factors (e.g., interference), node faults, or node mobility, the network can split into partitions. We define a *network partition* as a subset of $\mathcal{N}$ where all nodes in the same partition can bidirectionally exchange information with each other over one or more hops.

3.4.1 Correctness of Butler

The goal of Butler is to achieve synchronicity among the nodes in the network such that multiple nodes can safely initiate the upcoming communication round (i.e., within the capture window T_{cap}). Therefore, Butler is *correct* if the maximum difference $\Delta\tau$

TABLE 3.1: State transition matrix for node n_i in a network with N nodes. Reference times represent the different states and are ordered from the latest (τ_1) to the earliest (τ_N). Each entry describes the transition probability from one state to another. The triangular form results from BUTLER's behavior to synchronize only to earlier reference times.

	To							
From	u	τ_1	τ_2	…	τ_i	…	τ_{N-1}	τ_N
u	$P_{RX}(\varnothing)$	$P_{RX}(\tau_1)$	$P_{RX}(\tau_2)$	…	P_{TX}	…	$P_{RX}(\tau_{N-1})$	$P_{RX}(\tau_N)$
τ_1	0	$P_{RX}(\varnothing) + P_{TX}$	$P_{RX}(\tau_2)$	…	0	…	$P_{RX}(\tau_{N-1})$	$P_{RX}(\tau_N)$
τ_2	0	0	$P_{RX}(\varnothing) + P_{TX}$	…	0	…	$P_{RX}(\tau_{N-1})$	$P_{RX}(\tau_N)$
…	0	0	0	…	0	…	$P_{RX}(\tau_{N-1})$	$P_{RX}(\tau_N)$
τ_i	0	0	0	0	$P_{RX}(\varnothing) + P_{TX}$	…	$P_{RX}(\tau_{N-1})$	$P_{RX}(\tau_N)$
…	0	0	0	0	0	…	$P_{RX}(\tau_{N-1})$	$P_{RX}(\tau_N)$
τ_{N-1}	0	0	0	0	0	0	$P_{RX}(\varnothing) + P_{TX}$	$P_{RX}(\tau_N)$
τ_N	0	0	0	0	0	0	0	1

between the reference times of all nodes in the same network partition does not exceed T_{cap}.

3.4.1.1 Equal Clock Speeds

We begin with the simpler case, assuming all clocks run at the same speed, and prove the correctness of Butler for nodes in the same network partition.

Lemma 1. *If all clock speeds are equal, then there is a unique total order of all reference times over the entire execution of Butler.*

Proof. Reference times in Butler have a natural temporal order. If two reference times are equal, we select the reference time with the lower σ as the earlier one. However, nodes in Butler do not have a shared time base, so a sync message describes the reference time relative to its transmission time ($\tau = t_{grid} + T_\tau$). This relative duration is affected by the clock speed of the transmitting node, which would not be the case with absolute timestamps and a shared time base. Leveraging the assumption that all nodes have the same clock speed, the order of the reference times will be the same at any point during the execution of Butler. □

Theorem 1. *If all clock speeds are equal, Butler is correct.*

Proof. In Butler, the current reference time τ and its origin σ essentially describe the state τ_σ of a node. Based on Lemma 1 and without loss of generality, we assume the following (arbitrary) order among reference times $\tau_1 > \tau_2 > ... > \tau_N$, such that τ_N is the earliest. We can then create a corresponding state transition matrix for a node n_i, shown in Table 3.1, with the different states represented by the reference times. All nodes start in the unsynchronized state u, with $u > \tau_j$ for $1 \leq j \leq N$. The table entries describe the transition probabilities, for example, n_i transitions from u to τ_2 with $P_{RX}(\tau_2)$, the probability of receiving τ_2 from any other node. These transition probabilities are highly dependent on the situation and continuously change based on factors such as network topology, environment, node behavior, as well as the state of its neighboring nodes. In Butler, nodes propose their own reference time only if they are in state u, that is, they have not received any other reference time. Consequently, other nodes can reach τ_i only if n_i proposes it in the first place. Nodes remain in their current state if they either transmit (P_{TX}) or receive nothing ($P_{RX}(\varnothing)$), which includes receiving later reference times that are ignored.

In a network partition, all nodes can communicate with each other either directly or over multiple hops. Since the nodes decide randomly and independently if they transmit or listen, all transitions probabilities $P_{RX}(\tau_j)$ for proposed reference times τ_j in Table 3.1 are greater than 0. Butler ensures that transitions are only allowed toward earlier reference times, leading to the triangular form of the state transition matrix. Therefore, all nodes synchronize to the same reference time with high probability.

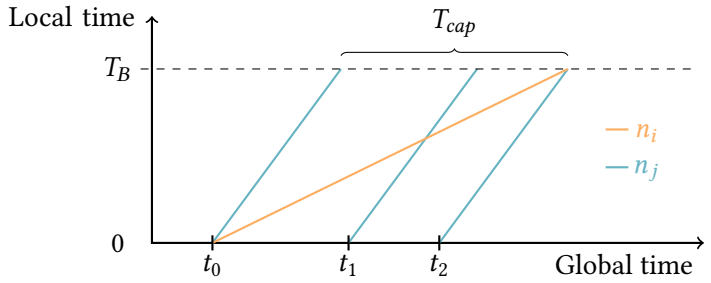


FIGURE 3.2: Local time vs. global time of nodes n_i and n_j with $\Delta\nu$ relative to each other. The nodes are synchronized to different reference times. Three scenarios with different initial offsets for n_j are shown. Although the order of reference times can change during execution, BUTLER is still correct.

Because we assume that all clock speeds are equal, it follows that the pairwise difference between the reference times of all nodes is $\Delta\tau = 0 < T_{cap}$, which proves correctness. □

3.4.1.2 Varying Clock Speeds

In the real world, the clock speeds of the nodes are imperfect and may vary within a certain range, specified by the frequency tolerance and stability properties of the clock source (e.g., a crystal oscillator). We can determine $\Delta\nu$, which is the maximum clock speed difference between two nodes based on the hardware specifications. For example, the IEEE 802.15.4 standard [71] requires $\Delta\nu = 80$ ppm (±40 ppm). We now extend the correctness proof for varying clock speeds by incorporating $\Delta\nu$.

Lemma 2. *Correctness of BUTLER can only be ensured if the maximal execution duration* $T_B \leq T_{cap}/\Delta\nu$.

Proof. We consider two nodes with $\Delta\nu$ relative to each other. Assuming that both nodes synchronize simultaneously to the same reference time, they are perfectly time-aligned at this point ($\Delta\tau = 0$). As time progresses, the local times of both nodes drift away from each other due to $\Delta\nu$, and it takes $T_{cap}/\Delta\nu$ time to have a difference of $\Delta\tau = T_{cap}$ between them. If BUTLER progresses further, correctness is violated, although both nodes are synchronized to the same reference time. Therefore, with varying clock speeds, it is necessary to limit the duration of BUTLER's execution to $T_B = T_{cap}/\Delta\nu$. □

Theorem 2. *BUTLER is correct as long as* $T_B \leq T_{cap}/\Delta\nu$.

Proof. With varying clock speeds, the order of the reference times can change during the execution of BUTLER. We will now prove BUTLER's correctness by analyzing the different situations that can occur with two reference times proposed by nodes n_i and n_j. Figure 3.2 shows how the local times of n_i and n_j progress compared to the global time. We assume that both nodes have $\Delta\nu$ relative to each other, with n_i having the lowest and n_j having the highest clock speed. The nodes execute BUTLER for a duration of T_B according to the local time (y-axis), leading to different execution times

concerning global time (x-axis). To visualize the different possible situations, we depict three scenarios for n_j, each with a different initial offset (t_0, t_1, and t_2) for the start of Butler, while n_i always starts at t_0.

If n_j starts Butler before t_0, τ_j will be earlier than τ_i for the entire Butler execution. Similarly, if n_j starts Butler after t_2, τ_i will always be earlier than τ_j. In these cases, all nodes eventually synchronize to the same reference time, and because of Lemma 2, $\Delta\tau$ will not exceed T_{cap}.

We now look at the case when n_j starts Butler between t_0 and t_2, for example at t_1. Since n_i starts Butler before n_j, τ_i is earlier than τ_j, and nodes would synchronize to τ_i upon reception. However, during the execution, the local time of n_j "overtakes" n_i (intersection), and the order of the reference times changes, that is, $\tau_i > \tau_j$. Depending on the initial offset of n_j, this can happen at any time during the execution, leading to nodes possibly being synchronized to different reference times when Butler terminates. Nevertheless, because of Lemma 2, the difference $\Delta\tau$ between the reference times is less than or equal to T_{cap}.

As a result, either there is a unique reference time, or all chosen reference times differ by at most T_{cap}, implying Butler's correctness. □

3.4.2 Network Partitions

Butler is a distributed synchronization mechanism with probabilistic transmit behavior that seamlessly supports network partitions. We assume that the network partitions can arbitrarily change between executions of Butler but that they remain stable while synchronization is ongoing, except that nodes can leave or fail at any point in time. This assumption is necessary to prevent a node with the earliest reference time from joining a new partition at the end of Butler, leaving no time for the other nodes in the partition to resynchronize. However, this is usually not a problem since the execution of Butler only takes a few tens of milliseconds (see Section 3.6.3).

With symmetrical communication links, and based on our definition of a network partition, all nodes that can communicate with each other must be in the same partition. Then, the presented proofs apply directly to each network partition. The state transition matrix in Table 3.1 would contain disjoint sets of states with one set per partition, and the transition probabilities between states of different sets would be 0 as no messages can be exchanged.

However, with asymmetrical links, nodes from one partition could receive a reference time τ_i from another node that is not in the same partition. If τ_i is earlier than all other reference times in the partition, all nodes will eventually synchronize to τ_i. Otherwise, if there exists an earlier reference time in the partition, then τ_i will be ignored. Therefore, the correctness of Butler is not affected by network partitions because it is irrelevant whether the node that proposed the reference time is part of the same partition.

3.4.3 Discussion

Limited time to converge. Our analysis shows an interesting area of tension between the theoretical proof of Butler's correctness and the practical challenge that the duration of Butler's execution is limited due to imperfect clocks ($\Delta\nu \neq 0$). Whether the maximum duration T_B of Butler (see Lemma 2) is sufficient for the nodes to converge on a single reference time depends on the network topology and environment, the node behavior, and the tolerance and stability of the clock source. In general, a network can have arbitrarily weak communication links so that the time to converge cannot be bounded. However, as we show in the evaluation in Section 3.6, these problems may be of low relevance in practice as the time to converge is several orders of magnitude lower than T_B. Among others, one reason is that the number of proposed reference times during the execution of Butler is low (as discussed in Section 3.5) compared to the overall number of nodes N, as most of the nodes will never propose their own reference time. Thus, the matrix in Table 3.1 will typically be sparse, which reduces the convergence time.

Impact of interference. During the execution of Butler, the nodes will receive a reference time several times due to the random transmit behavior; thus, missing some messages, for example, due to interference, can usually be compensated. In general, stronger interference leads to more message loss and increases the average time to converge but does not violate the correctness of Butler, which is independent of the receive probability. Note that a receive probability of 0 means the node is not connected. In practice, stronger interference can be proactively accounted for by deliberately extending the duration of Butler.

3.5 Implementation

We have implemented Butler on the popular Nordic nRF52840 platform using the IEEE 802.15.4 PHY [71]. The code is published as open source at `https://gitlab.com/nes-lab/butler`.

Usage of Butler. It is straightforward to combine Butler with an existing ST-based communication protocol. Butler's API is a single function `butler_start(id)`, which takes the ID of the node as an argument. During the execution, Butler takes care of correctly handling all interrupts and should not be interfered with from the outside. Upon termination, the function returns the final reference time and origin shortly after reaching it. At this point, the nodes are synchronized and can start the next communication round. Butler does not require periodic or repeated execution and is scheduled on demand, provided the maximum initial offset Δt_{ofs} based on the communication period is known.

Sync message. Figure 3.3 shows the structure of a sync message for the IEEE 802.15.4 PHY. The synchronization header (SHR) is responsible for the capture window T_{cap} and contains the preamble and the SFD. SHR and the length field (len) are mandatory

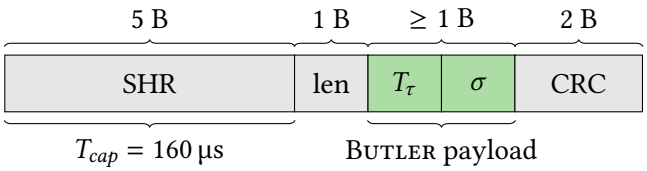


Figure 3.3: Packet structure of a sync message in Butler using the IEEE 802.15.4 PHY.

parts of the communication standard. Butler adds the remaining duration T_τ and the origin σ as payload, whose space requirements are known at compile-time but vary depending on the application and network size. Since the duration of Butler is initialized as a multiple of the slot length T_{slot} (line 2 in Algorithm 2), we can represent T_τ in the sync message more compactly as the number of remaining slots. To detect and filter out corrupted packets, we use a hardware-supported CRC.

From design to implementation. Butler operates in a slotted fashion (see Algorithm 2), where nodes decide to transmit or receive in every slot. The corresponding transmit probability P_{TX} will be low in practice (e.g., 2–4 % in the evaluation), so nodes will often be receiving for multiple consecutive slots in a row. The nodes will continuously listen for incoming messages independent of slot boundaries and only align TX decisions to the slot grid. This implementation increases efficiency and avoids possible sync message misses at the slot boundaries, as sync messages can be received at any point in time.

In Butler's design, nodes switch instantly between RX and TX and vice versa, for example, after synchronizing to a new reference time. However, the radio hardware requires a turnaround time of 40 µs to execute this mode change. During the switch, the radio is deaf and cannot receive or transmit, effectively causing service downtime. One option to alleviate this issue would be to increase the slot length T_{slot} by the turnaround time, which would affect all slots. However, the number of slots in which a node synchronizes to a new reference time and is thereafter forced to transmit is only a fraction compared to the overall number of slots. Therefore, we instead opted to skip one slot when switching from RX to TX as it is more efficient to keep T_{slot} unchanged.

A crucial point in Butler is the computation of the reference time $\hat{\tau}$ from the received sync message (line 17 in Algorithm 2). The remaining duration $\hat{T}_\tau$ is part of the sync message and $\hat{t}_{grid}$ is determined based on the receive timestamp. This requires that the receive timestamp is equal to t_{grid} of the sender, except for negligible differences due to the time of flight of packets. We discovered that this is not the case on the nRF52840 platform and the receive timestamp is delayed by around 10 bit durations, depending on the data rate of the current radio mode. For example, using a data rate of 250 kbps and the IEEE 802.15.4 PHY results in a receive timestamp delay of 40 µs, which has to be considered for the computation of $\hat{t}_{grid}$. As this delay splits equally between the TX and RX paths, we have to add 20 µs to the slot length T_{slot}, which is otherwise oriented at the size of the sync message (Figure 3.3).

Variable transmit probabilities for efficiency. A probabilistic and independent transmit decision is essential to make Butler fault-tolerant and avoid single points of failure. As shown in Figure 3.1, the local slot grids are initially unaligned, but increasingly synchronize as the execution progresses. At the beginning, the chances are high that transmissions overlap arbitrarily and violate the timing requirement T_{cap} of the capture effect, which leads to a reduced communication efficiency (i.e., a lower packet reception rate). Thus, P_{TX} should initially be chosen cautiously to reduce the number of transmissions. A side effect of a lower initial P_{TX} is that fewer reference times will be proposed, which decreases the overall convergence time. However, nodes already aligned to the same reference time can benefit from the capture effect since their transmissions start concurrently within T_{cap}. With an increasing number of aligned nodes, a higher P_{TX} improves the convergence to the final reference time. Therefore, the optimal value for P_{TX} varies over time and is network-specific. A similar challenge is faced in the Mixer protocol (Chapter 2), where the transmit decisions depend on the local node density. We tested different topologies and found that choosing $P_{TX} = \frac{100\,\%}{N*2}$ until a node first transmits and doubling after that provides a conservative starting point for many topologies.

3.6 Evaluation

Based on our implementation, we evaluate Butler in a real-world wireless testbed. We investigate Butler's behavior and confirm its correctness, together with measurements regarding performance and efficiency. Finally, we examine the interaction between Butler and our ST-based communication protocol Mixer (Chapter 2), and its impact on the communication performance. Our key findings are:

- *Correctness:* In our experiments, all nodes always synchronize to the same reference time at the end of Butler, validating Butler's correctness from the analysis.
- *Accuracy:* Butler synchronizes nodes to within ±3 µs, which is well below the maximum tolerable time offset of 160 µs (i.e., size of the capture window).
- *Efficiency:* Thanks to Butler's efficient run-time execution, the temporal overhead of synchronizing the nodes is small and significantly below 1 % in most scenarios.
- *End-to-end performance:* Butler increases the availability of existing communication protocols without any negative impact on the communication performance.

3.6.1 Experimental Settings

All our experiments are executed on the FlockLab [173] testbed with 23 nodes deployed in an office environment as shown in Figure 3.4. The experiments were conducted during the daytime and, thus, exposed to various sources of interference, for example,

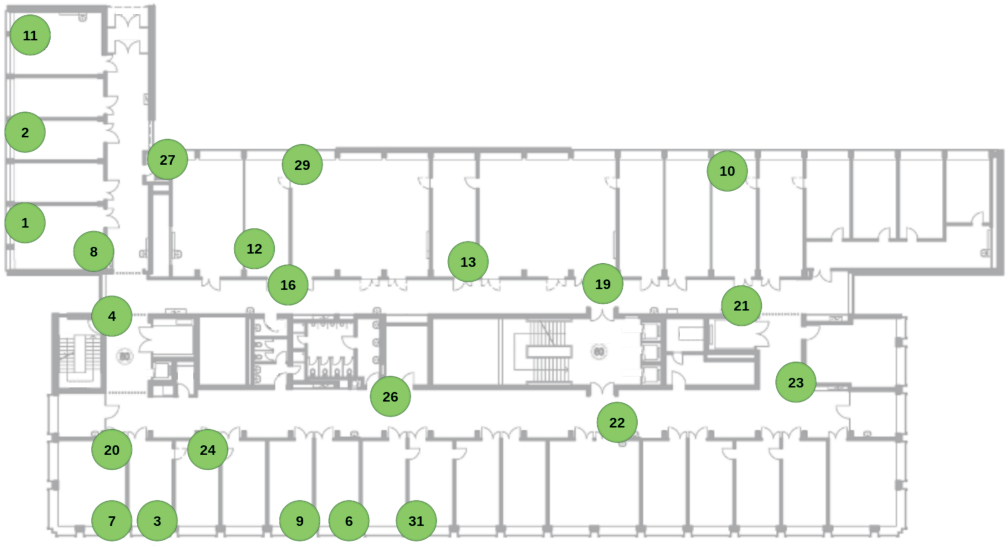


FIGURE 3.4: The FlockLab testbed with 23 nRF52840 devices.

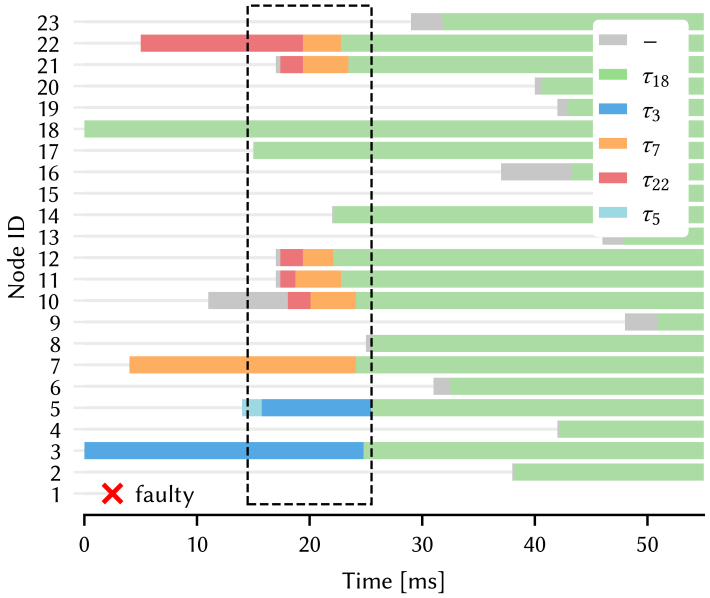
Wi-Fi. Our implementation uses the nRF52840 platform and the IEEE 802.15.4 PHY. Using a transmit power of 8 dBm, the nodes form a network with 3-4 hops.

Since the clock drift of the nodes is unknown and also subject to change, we use the GPIO actuation capabilities of FlockLab to control the timing in our experiments. This also allows us to test larger initial offsets between the nodes without having to run excessively long experiments (i.e., with huge gaps between communication rounds). A final aspect is that by using the GPIO to purposely control the timing, we also increase the repeatability of our experiments. This way, each node realizes a random initial offset in the range $0 \leq t_{ofs} \leq 50\,\text{ms}$. The maximum offset $\Delta t_{ofs} = 50\,\text{ms}$ corresponds to a communication period of $\approx$10 min. To assess the robustness of BUTLER, we inject artificial node faults. Therefore, each node independently decides with a probability of 5 % not to participate in the next BUTLER execution. Furthermore, with this fault probability, the network sometimes splits into two partitions during the experiments, which permits an investigation of the behavior under network partitions.

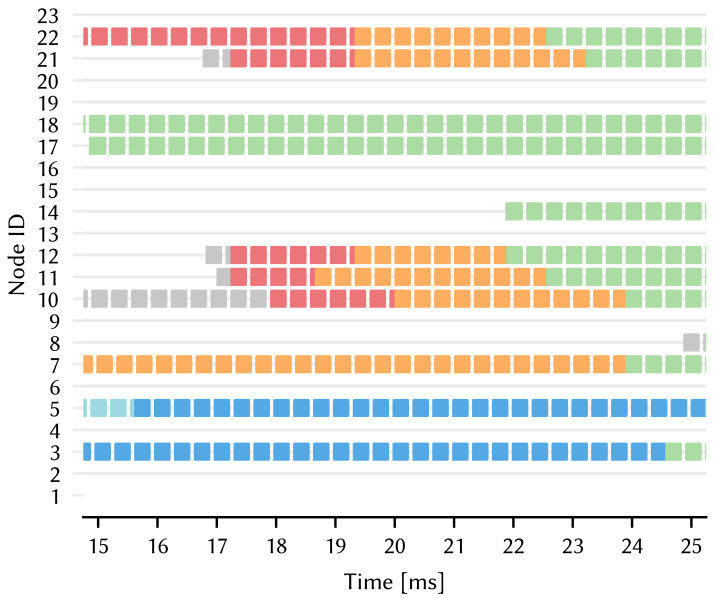
In BUTLER, the sync information σ and T_τ require 1 B each, resulting in a packet size of 10 B (see Section 3.5) and a slot length T_{slot} of 335 µs. Using a few trial runs, we find that 250 slots (83.75 ms) are sufficient for BUTLER to synchronize all nodes on FlockLab for the considered range of initial offsets (see Section 3.6.3). We use two different values for the transmission probability (see Section 3.5), which are $P_{TX} = 2.2\,\%$ for the time until the first transmission, and $P_{TX} = 4.3\,\%$ afterward.

3.6.2 BUTLER in Action

Before delving into the evaluation of BUTLER's performance, we look at its operation in a real low-power wireless network. Figure 3.5 depicts the behavior of each of the 23 nodes on FlockLab during one representative execution of BUTLER. The beginning of



(a) Nodes start BUTLER at different times but eventually synchronize to the same reference time of node 18 at around 50 ms. Node 1 is faulty.



(b) Zoom into the marked area of the left plot, showing the individual slots. During this interval, the most synchronization events happen.

FIGURE 3.5: Time synchronization of nodes during the execution of BUTLER with colors indicating which reference time a node follows during execution. Reference times in the legend (gray means unsynchronized) are sorted from earliest (τ_{18}) to latest (τ_5).

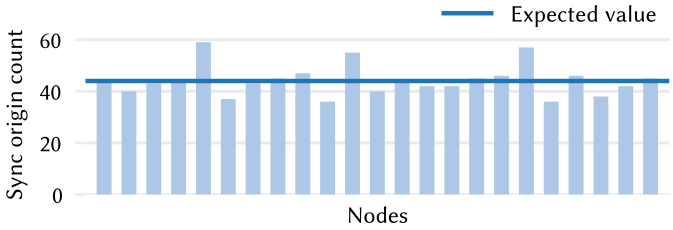


FIGURE 3.6: The number of times each node proposed the final reference time is uniformly distributed in BUTLER. With 1016 runs, each node is expected to be the sync origin 44 times.

a bar in Figure 3.5a indicates when a node started the execution of BUTLER. The color of the bar indicates which proposed reference time a node currently follows. We can see that the nodes start BUTLER at different times with $\Delta t_{ofs} = 50\,\text{ms}$. Despite these significant initial offsets, BUTLER eventually makes all nodes follow the same reference time, as indicated by the green bars in the figure.

Diving a bit deeper into BUTLER's behavior in this particular run, we can see that most nodes initially have a gray bar. This means that these nodes did not propose their own reference time, but synchronized to the reference time of the first sync message they received. In contrast, nodes with a colored bar at the beginning did propose their own reference time at some point before receiving a sync message. Overall, there were 5 different reference times proposed in this execution, originating from nodes 3, 5, 7, 18, and 22. Node 18 started BUTLER first, a few microseconds before node 3, so the reference time of node 18 (green) is the earliest to which all other nodes eventually synchronize.

During the framed time interval between 15–25 ms, many nodes synchronize to new reference times. We zoom into this interval in Figure 3.5b, where we can see the individual slots of BUTLER. Looking at the orange slots, we can see how the respective reference time propagates from hop to hop through parts of the network. At around 19 ms node 11 synchronizes to it. Two slots later, which is the implementation-specific delay due to the RX-TX turnaround time of the radio, node 11 transmits the new synchronization information (not shown in the figure) that is then received by the nodes 12, 21, and 22. Again two slots later, the orange reference time is further relayed to node 10. Eventually, however, the green reference time prevails as it is the earliest among all proposed reference times in this particular run.

The origin of the final reference time, that is, the node which proposed it, is uniformly distributed among all nodes over all runs as shown in Figure 3.6, underlining BUTLER's distributed nature. Moreover, the experiments show that the number of proposed reference times is related to the transmit probability P_{TX}. There are, on average, ≈4 different reference times.

The network splits into two partitions in a few runs due to multiple node faults. We find that the nodes correctly synchronize to the respective reference time in each

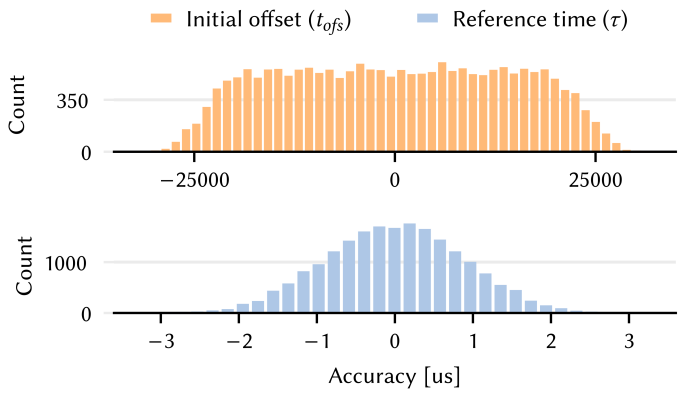


Figure 3.7: Synchronization accuracy before and after Butler. Despite excessive initial offsets between the nodes (top), Butler synchronizes all nodes to within a few microseconds (bottom), which is far better than the required $T_{cap} = 160\,\mu s$.

partition. Composition into even more partitions would also work seamlessly without adjusting the protocol.

3.6.3 Butler's Performance and Efficiency

Accuracy. The main goal of Butler is to achieve synchronicity, such that the nodes in the network are time-aligned within the capture window T_{cap}. Our experiments show that Butler achieves this goal and synchronizes the nodes well below T_{cap}.

We use the GPIO tracing capabilities of FlockLab and mark the times when a node starts (t_{ofs}) and finishes (τ) Butler. The accuracy is measured as the difference between the observed values and the empirical mean in each Butler execution. Figure 3.7 shows the results for around 900 executions. At the top, the distribution of the initial offsets t_{ofs} across all nodes is as expected, since each node picks a random initial offset between 0 and 50 ms. [1] In the lower plot we can see that despite the excessive initial offsets, most of the nodes achieve an accuracy of ±2 µs, with at most 6 µs between any two reference times. These values are well below the requirement of $T_{cap} = 160\,\mu s$ and validate the correctness of Butler, a necessary precondition for achieving high accuracy.

Efficiency. We evaluate the efficiency of Butler by measuring the time it takes to synchronize the network. To this end, we run experiments with varying maximum offsets (Δt_{ofs}), and nodes choose a random initial offset in the range of $0\,ms \leq t_{ofs} \leq \Delta t_{ofs}$. The time to synchronize the network starts with the first node entering Butler and ends when all nodes are synchronized to the same reference time, for example, ≈51 ms in Figure 3.5a.

[1] The accuracy exceeds −25 000 µs and 25 000 µs because it is based on the empirical mean of each Butler execution, which has some variation.

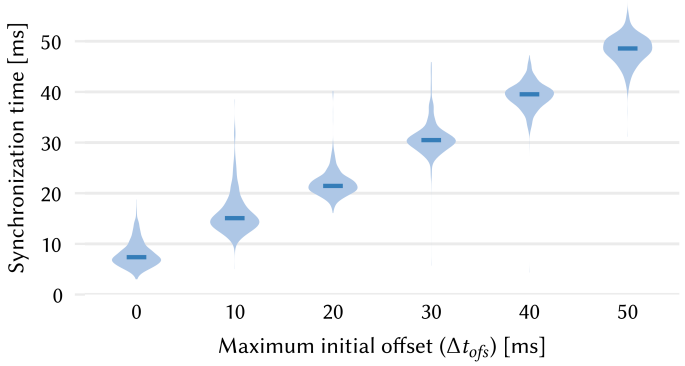


FIGURE 3.8: Time needed to synchronize all nodes for different initial offsets. BUTLER reliably synchronizes all nodes within a few milliseconds and the synchronization time scales well with the clock drift.

Figure 3.8 shows the distribution and mean of synchronization times for all nodes and all BUTLER executions (≈1000) per experiment. We see that on FlockLab it takes on average about 8 ms to synchronize all nodes using BUTLER for $\Delta t_{ofs} = 0$ ms. With larger offsets, the synchronization time converges toward the respective Δt_{ofs} value. This is because many nodes are already synchronized to the final reference time before the last node starts BUTLER, as can be seen in Figure 3.5a.

Duration and overhead. In general, the duration of BUTLER is the sum of two factors. One factor is the convergence time on the specific network topology, which can be experimentally explored at $\Delta t_{ofs} = 0$ ms and is ≈35 ms (100 slots) on FlockLab. The other factor is the maximum initial offset Δt_{ofs} to compensate for the accumulated clock drift since the last synchronization. Since the latter only depends on the communication period (assuming maximum clock drift $\Delta\nu$), the duration of BUTLER can be easily adjusted to any period during run-time; thus, BUTLER can be executed on demand. However, for very long communication periods, Δt_{ofs} can become large and exceed BUTLER's maximum duration T_B (Section 3.4.1). For instance, in our experiments $T_B = 2$ s, which corresponds to a communication period of ≈7 h. To support longer periods, BUTLER would have to be executed in-between to reset the accumulated clock drift.

In most scenarios, BUTLER only needs to be executed once per communication period, so we report its temporal overhead in relation to the period. Our results listed in Table 3.2 demonstrate that BUTLER is a lightweight mechanism with very little to negligible temporal overhead, enabling easy integration with communication protocols as BUTLER does not constrain their execution. Moreover, the temporal overhead decreases as the initial offset and associated communication period increase. For example, at a communication period of ≈3 s, the temporal overhead drops below 1 %, with only 0.01 % at a period of 10 min. However, the increased availability of BUTLER does come at the cost of increased energy consumption, as the execution time of BUTLER could otherwise be spent in sleep mode.

Table 3.2: Temporal overhead of Butler for different initial offsets and their corresponding communication periods.

Max. initial offset (Δt_{ofs})	Corresponding com. period	Butler duration	Temporal overhead
80 µs	1 s	33.58 ms	3.36 %
400 µs	5 s	33.9 ms	0.68 %
800 µs	10 s	34.3 ms	0.34 %
4.8 ms	1 min	38.3 ms	0.06 %
24 ms	5 min	57.5 ms	0.02 %
48 ms	10 min	81.5 ms	0.01 %

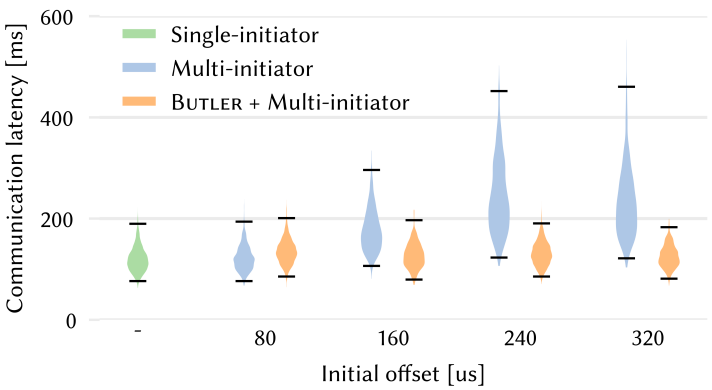
3.6.4 Making an Existing Protocol Available Through Butler

After investigating Butler's performance and efficiency in isolation, we now turn to the target use case where Butler is used to increase the availability of a low-power wireless communication protocol.

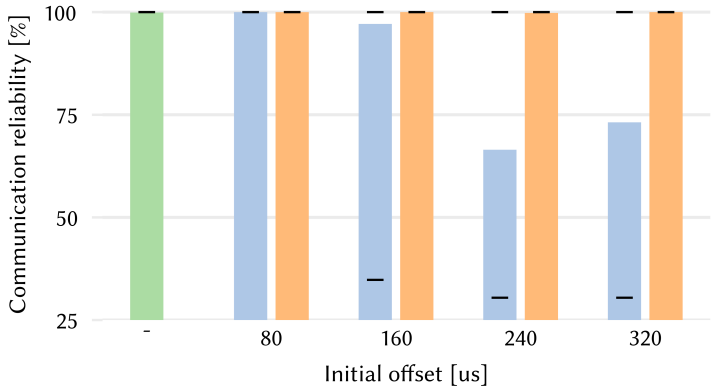
Scenario and settings. As an example, we use our novel communication protocol Mixer (Chapter 2), which offers an efficient, reliable, and scalable communication service. However, Mixer provides no availability, often a key requirement in control applications, as it relies on a single initiator node that starts the many-to-many packet exchange. To overcome this problem and enable the use of multiple initiators in Mixer, we let Butler run before every communication round to synchronize the set of initiators. To be able to compare different settings and reproduce our results, we use a fixed set of two initiators located at opposite ends of the FlockLab testbed and also refrain from injecting artificial node failures.[2]

We compare the performance of Mixer with and without Butler for different initial offsets between the two initiators. Compared to the previous experiments, we use much smaller initial offsets here to show that multi-initiator Mixer requires help from Butler already at short communication periods. However, Butler can be used efficiently with Mixer irrespective of the communication period and resulting initial offset. For each setting, we conduct an experimental run that involves around 500 communication rounds. We also run the original Mixer with a single initiator as a baseline for the communication performance. In every communication round, each node initially has a 16 B message that it needs to share with all other nodes in the network during the Mixer round, so that eventually every node has all 23 messages. We consider two key metrics: *Latency*, which is the time it takes for a node to receive all messages in a round, and *reliability*, which is the fraction of received messages per round. Note that latency does not include Butler's execution time because Butler finishes before the scheduled communication round starts.

[2]We note that in order to maximize availability, a larger set of initiator nodes randomly and dynamically chosen at run-time should be used, which is straightforwardly supported by our implementation.



(a) Communication latency. Butler effectively addresses the clock drift between multiple initiators and enables the communication protocol to deliver the expected performance at significantly increased availability.



(b) Communication reliability. With Butler, the reliability remains unchanged above 99.9 %. Without Butler, multiple initiators lead to unreliable and unpredictable communication rounds.

Figure 3.9: Communication performance of Mixer using multiple initiators with and without Butler.

Results. Figure 3.9a shows the latency distribution with single- and multi-initiator Mixer for different initial offsets; markers indicate the 1st and 99th percentiles. We can see that without Butler, the latency increases significantly by up to 2.8× for increasing initial offsets. When instead extending it with Butler, the latency of multi-initiator Mixer remains as low as for the original Mixer with a single initiator while providing higher availability. This is thanks to Butler's ability to accurately synchronize the initiators, which becomes an absolute necessity already for an initial offset of 160 µs to achieve high performance. Thus, the experiments also confirm our assumption about T_{cap} for correctness in Section 3.4.1.

Figure 3.9b shows the reliability of single- and multi-initiator Mixer for different initial offsets when we limit the length of the multi-initiator Mixer rounds to the time needed by single-initiator Mixer (about 300 ms). This scenario is representative of typical constraints found in control applications, where interactions between distributed sensors and actuators must be completed within hard real-time deadlines to match the dynamics of physical processes [4]. Looking at Figure 3.9b, we see without Butler the mean reliability decreases dramatically by up to 30 % as the initial offset increases. With Butler, the reliability remains unchanged and always above 99.9 %.

In summary, these results demonstrate that Butler effectively solves the problem of clock drift when using multiple initiators. Butler increases the availability of low-power wireless communication without sacrificing performance.

3.7 Related Work

Butler is the first work to address the availability problem of ST-based communication protocols. However, the underlying concept is closely related to the existing literature on time synchronization. Most time synchronization algorithms aim to provide an accurate, globally shared, and constantly available time base to all nodes in the network. While this is a powerful synchronization service that is essential for some applications, it needs to run periodically, and the associated overhead in terms of energy, time, and wireless bandwidth is very high. In fact, to distribute the initiator role among multiple nodes, which is what Butler aims for to increase availability, the nodes do not need a globally shared time base that is maintained for the entire lifetime of the system: all they need is to be able to perform a coordinated action [89]. This is also known as *synchronicity* [180] and can be achieved with less effort compared to full-fledged time synchronization.

Many time synchronization protocols, including TPSN [54], FTSP [119], Glossy [48], PulseSync [95], and TATS [100], use one dedicated node as a time reference for the entire network. Generally, these algorithms achieve excellent synchronization accuracy but are not fault-tolerant, which is a prerequisite for high availability. Furthermore, they often rely on topology information, causing instability in dynamic networks. The single node providing the reference time is also insufficient if the network splits

into several isolated partitions. Butler overcomes these issues by adopting a fully distributed approach that does not rely on topology information.

Over the years, several distributed synchronization protocols have been developed to mitigate centralization issues, such as RFA [180], DCTS [154], ATS [153], MTS [62], and MACTS [158]. These protocols do not rely on special nodes and are thus more robust and versatile than their centralized counterparts. The downside, however, is that these algorithms typically require a significant amount of time to synchronize the network, ranging from tens of seconds to multiple minutes. By contrast, Butler needs only tens of milliseconds to synchronize an entire network to within a few microseconds, thus substantially saving energy, time, and wireless bandwidth. The difference is that in most synchronization protocols, the nodes converge, for example, by averaging the local clocks in an iterative process, which is needed to find a stable global time but requires a large number of messages to determine and account for the different clock drifts [158]. Furthermore, nodes that lose their state, for example, due to a failure, potentially require all nodes to converge again. Butler does not adjust clock drift and uses the natural order of proposed reference times, requiring only a few messages to achieve the goal of synchronicity. Butler's short duration also simplifies the integration with communication protocols, which can be difficult with existing time synchronization protocols due to their significant overhead and the need for periodic executions.

Postscript

In response to the availability weakness of ST-based communication protocols, we have designed Butler, a new distributed low-power wireless synchronization scheme that eliminates their single point of failure. Butler can be used on demand anytime synchronization is needed, such as before communication with ST-based protocols. A formal analysis shows that Butler can correctly synchronize the network toward the same reference time, and experiments validate this claim and reveal an exceptional accuracy of ±3 µs. At the same time, its temporal overhead ranges well below 1 %. Moreover, we demonstrate that using Mixer and Butler together makes a truly fault-tolerant low-power wireless protocol feasible. Since Butler does not interfere with the communication protocol, the communication performance remains virtually unchanged while its last vulnerability is eliminated. Thus, Butler helps to fulfill the high dependability requirements of present and emerging CPS applications.

Part II
Wireless Cyber-Physical Systems

4

Feedback Control Goes Wireless: Guaranteed Stability over Low-Power Multi-Hop Networks

Preface

In the second part of this thesis, we are devoted to developing and analyzing real-world wireless CPS based on a tight co-design approach of communication and control. Specifically, we focus on the need to close feedback loops fast and over long distances, which is key to emerging applications; for example, robot motion control and swarm coordination require update intervals of tens of milliseconds. Prior works on wireless control that validate their designs through experiments on physical platforms have only been demonstrated for systems with slow dynamics or in small single-hop networks, significantly alleviating the challenges.

In this chapter, we demonstrate the feasibility of fast feedback control with stability guarantees via low-power wireless multi-hop networks. Our wireless embedded system is based on a co-design that tames imperfections impairing the control performance (e.g., jitter and message loss) to the extent possible and exploits the essential properties of this system in the control design. Due to careful orchestration of communication and control tasks at run-time, we can meet real-time constraints and provably guarantee closed-loop stability for physical processes with LTI dynamics. Experiments on a CPS testbed with 20 wireless nodes and multiple cart-pole systems show, for the first time, feedback control and coordination over wireless multi-hop networks for update intervals of 20–50 ms.

4.1 Introduction

CPS use embedded computers and networks to monitor and control physical systems [39]. While monitoring using *sensors* allows, for example, to better understand certain characteristics of environmental processes [32], it is control and coordination through *actuators* what nurtures the CPS vision exemplified by robotic materials [33], smart transportation [19], and multi-robot swarms for disaster response and manufacturing [61].

A key hurdle to realizing this vision is how to close the *feedback loops* between sensors and actuators as these may be numerous, mobile, distributed over large spaces, and attached to devices with size, weight, and cost constraints. Wireless multi-hop communication among low-power, possibly battery-powered devices[1] provides the cost efficiency and flexibility to overcome this hurdle [101, 179] if two requirements are met. First, fast feedback is needed to keep up with the dynamics of physical systems [12]; for example, robot motion control and drone swarm coordination require update intervals of tens of milliseconds [1, 140]. Second, as feedback control modifies the dynamics of physical systems [11], guaranteeing *closed-loop stability* under imperfect wireless communication is a major concern.

Hence, this chapter investigates the following question: *Is it possible to enable fast feedback control and coordination across* ***real-world*** *low-power wireless multi-hop networks with formal guarantees on closed-loop stability?* Prior works on control over wireless that validate their design through experiments on physical platforms do not provide an affirmative answer. As shown in Figure 4.1 and detailed in Section 4.2, solutions based on *multi-hop* communication have only been demonstrated for physical systems with *slow* dynamics (i.e., update intervals of seconds) and do not provide stability guarantees. Practical solutions with stability guarantees for *fast* process dynamics (i.e., update intervals of tens of milliseconds as typical of, e.g., mechanical systems) exist, but these are only applicable to *single-hop* networks and therefore lack the scalability and flexibility required by many future CPS applications [61, 108].

Contribution and road map. This chapter presents the design, analysis, and practical validation of a wireless CPS that fills this gap. Section 4.3 highlights the main challenges and corresponding system design goals we must achieve when closing feedback loops fast over wireless multi-hop networks. Our approach is based on a careful co-design of the wireless embedded components (in terms of hardware and software) and the closed-loop control system, as described in Section 4.4 and Section 4.5. More concretely, we tame typical wireless network imperfections, such as message loss and jitter, so that they can be tackled by well-known control techniques or safely neglected. As a result, our solution is amenable to a formal end-to-end analysis of all CPS components (i.e., wireless embedded, control, and physical systems), which we exploit to provably guarantee closed-loop stability for physical systems with LTI dynamics. Moreover,

[1] While actuators may need wall power, low-power operation is crucial for sensors and controllers, which may run on batteries and harvest energy from the environment [4].

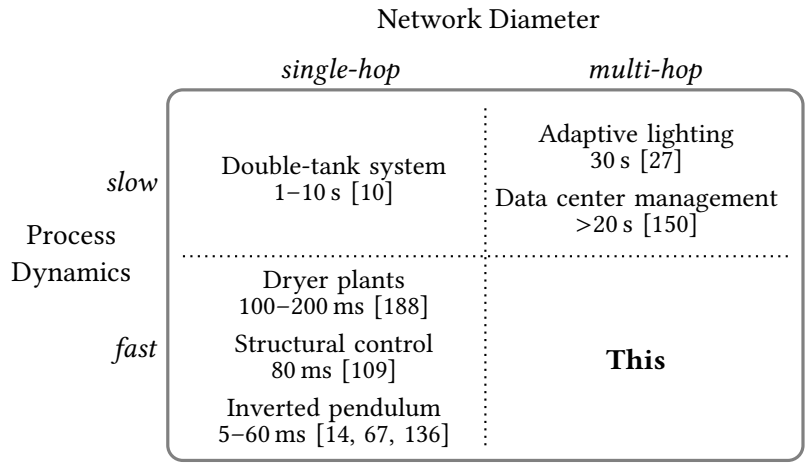


Figure 4.1: Design space of wireless CPS that have been validated on physical platforms and real wireless networks.

unlike prior work, our solution supports control and coordination of multiple physical systems out of the box, which is a key asset in many CPS applications [1, 61, 140].

To evaluate our design in Section 4.6, we have developed a CPS testbed that consists of 20 wireless embedded devices forming a three-hop network and multiple cart-pole systems whose dynamics match a range of real-world mechanical systems [11, 171]. As such, our testbed addresses an important need in CPS research [101]. Our experiments reveal the following key findings: a) two inverted pendulums can be safely stabilized by two remote controllers across the three-hop wireless network; b) the movement of five cart-poles can be synchronized reliably over the network; c) increasing message loss rates and update intervals can be tolerated at reduced control performance; and d) the experiments confirm our analyses.

In summary, this work contributes the following:

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

4.2 Related Work

Feedback control over wireless communication networks has been extensively studied. For instance, the control community has investigated control design and stability analysis for wireless (and wired) networks based on different system architectures, delay models, and message loss processes (see, e.g., [6, 55, 104, 142, 160, 161, 175, 185, 193]); recent surveys provide an overview of this body of fundamental research [69, 191].

However, the majority of those works focuses on theoretical analyses or validates new wireless CPS designs (e.g., based on WirelessHART [96, 111]) only in simulation, thereby ignoring many fundamental challenges that may complicate or prevent a real implementation [101]. One of the challenges, as detailed in Section 4.3, is that even slight variations in the quality of a wireless link can trigger drastic changes in the routing topology [27]—and this can happen several times per minute [56]. Hence, to establish trust in feedback control over wireless, a real-world validation against these *dynamics* on a realistic CPS testbed is absolutely essential [101], as opposed to considering setups with a *statically configured* routing topology and only a few nodes on a desk as, for example, in [155].

Figure 4.1 classifies prior control-over-wireless solutions that have been validated using experiments on physical platforms and against the dynamics of real wireless networks along two dimensions: the diameter of the network (*single-hop* or *multi-hop*) and the dynamics of the physical system (*slow* or *fast*). While not representing absolute categories, we use slow to refer to update intervals on the order of seconds, which is typically insufficient for feedback control of, for example, mechanical systems.

In the *single-hop/slow* category, Araújo et al. [10] 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 1–10 s 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 [44], IEEE 802.11 [136], and IEEE 802.15.4 [14, 67]. 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 [188], 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 [109].

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 [27]. Owing to the length of the tunnels, multi-hop communication becomes unavoidable, yet the required update interval of 30 s 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 s or greater [150].

In contrast to these works, as illustrated in Figure 4.1, we demonstrate *fast* feedback control over wireless *multi-hop* networks at update intervals of 20–50 ms, which is significantly faster than existing multi-hop solutions. Moreover, we provide a formal stability proof, and our solution seamlessly supports both control and coordination of multiple physical systems, which we validate through experiments on a real-world CPS testbed.

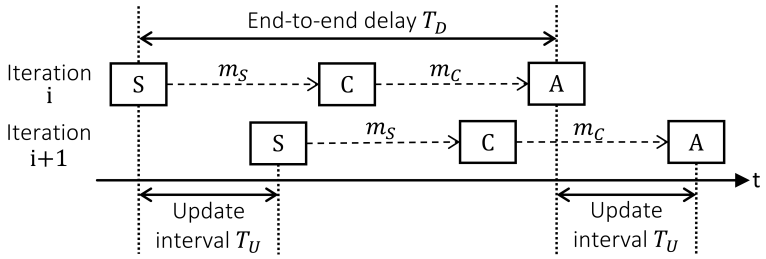


FIGURE 4.2: Application tasks and message transfers of a single feedback loop. In every iteration, the sensing task (S) takes a measurement of the physical system and sends it to the control task (C), which computes a control signal and sends it to the actuation task (A).

4.3 Problem Setting and Approach

Scenario. We consider wireless CPS that consist of a set of embedded devices equipped with low-power wireless radios. The devices execute different *application tasks* (i.e., sensing, control, and actuation) that exchange *messages* over a wireless multi-hop network. Each node may execute multiple application tasks, which may belong to different distributed feedback loops. As an example, Figure 4.2 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 [133]. 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 [27], introducing significant jitter. Delays and jitter can both destabilize a feedback system [175, 183].
- *Constrained traffic patterns.* In a single-hop network, each node can communicate with every other node due to the broadcast property of the wireless medium. This is generally not the case in a multi-hop network. For example, WirelessHART only supports communication to and from a gateway that connects the wireless network to the control system. Feedback control under constrained traffic patterns is more challenging and may imply poor control performance or even infeasibility of closed-loop stability [187].

- *Correlated message loss.* Message loss is a common phenomenon in wireless networks, which complicates control design. Further, because there is often significant correlation among the message losses [162], 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 [43, 56] and may further hinder control design and stability analysis [191].

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

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

G2 support arbitrary traffic patterns in multi-hop networks with real dynamics (e.g., time-varying link qualities);

G3 operate efficiently in terms of limited resources, while accommodating the computational needs of the controller.

On the other hand, the control design aims to:

G4 incorporate all essential properties of the wireless embedded system to guarantee closed-loop stability for the entire CPS for physical systems with LTI dynamics;

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

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

4.4 Wireless Embedded System Design

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

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

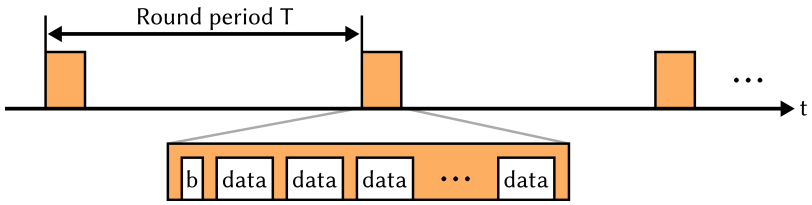


FIGURE 4.3: Time-triggered operation of low-power wireless multi-hop protocol. Communication occurs in rounds that are scheduled with a given round period T. Every beacon (b) and data slot in a round corresponds to an efficient, reliable one-to-all Glossy flood [48].

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

4.4.1 Low-Power Wireless Protocol

To support arbitrary traffic patterns (**G2**), we require a multi-hop protocol capable of many-to-all communication. Moreover, the protocol must be highly reliable and the time needed for many-to-all communication must be tightly bounded (**G1**). It has been shown that a solution based on Glossy floods [48] can meet these requirements with high efficiency (**G3**) in the face of wireless dynamics (**G2**) [195]. Thus, similar to other recent proposals [49, 73], 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 4.3, 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 %, and provides microsecond-level network-wide time synchronization [48]. In particular, a flood initiated by a dedicated node in the *beacon* slot (b) at the beginning of every round is used for synchronization. Nodes exploit the synchronization to remain in a low-power sleep mode between rounds and to awake in time for the next round, as specified by the round period T.

It is important to note that, because of the way Glossy exploits ST [48], the wireless protocol operates *independently* of the time-varying network topology. In particular, this means that the wireless protocol and any logic built on top of it, such as a control algorithm, need not worry about the state of individual wireless links in the network. This is a key difference to existing wireless protocols based on routing, such as WirelessHART and 6TiSCH, which, as we shall see, simplifies the control design and allows for providing formal guarantees that also hold in practice.

As detailed in Section 4.4.3, we compute the communication schedule offline based on the traffic demands, and distribute it to all nodes before the application operation starts. A schedule includes the assignment of messages to *data* slots in each round

(see Figure 4.3) and the round period T. Using static schedules brings several benefits. First, we can a priori verify if closed-loop stability can be guaranteed for the achievable latencies (see Section 4.5). Second, compared to prior solutions [49, 73, 195, 74], we can support significantly shorter latencies, and the protocol is more energy efficient (no need to send schedules) and more reliable (schedules cannot be lost).

4.4.2 Hardware Platform

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

For this reason, we leverage a heterogeneous dual-processor platform (DPP). Application tasks execute exclusively on a 32-bit MSP432P401R ARM Cortex-M4F application processor (AP) running at 48 MHz, while the wireless protocol executes on a dedicated 16-bit CC430F5147 communication processor (CP) running at 13 MHz. The AP has a floating-point unit and a rich instruction set, accelerating computations related to sensing and control. The CP features a low-power microcontroller and a low-power wireless radio operating at 250 kbps in the 868 MHz frequency band.

AP and CP are interconnected using Bolt [166], an ultra-low-power processor interconnect that supports asynchronous bidirectional message passing with formally verified worst-case execution times. Bolt decouples the two processors with respect to time, power, and clock domains, enabling energy-efficient concurrent executions with only small and bounded interference, thereby limiting jitter and preserving the time-sensitive operation of the wireless protocol.

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

4.4.3 Scheduling Framework

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

Figure 4.4 shows a possible schedule of the application tasks and message transfers. After sensing (S1), AP_P writes a message containing the sensor reading into Bolt (w).

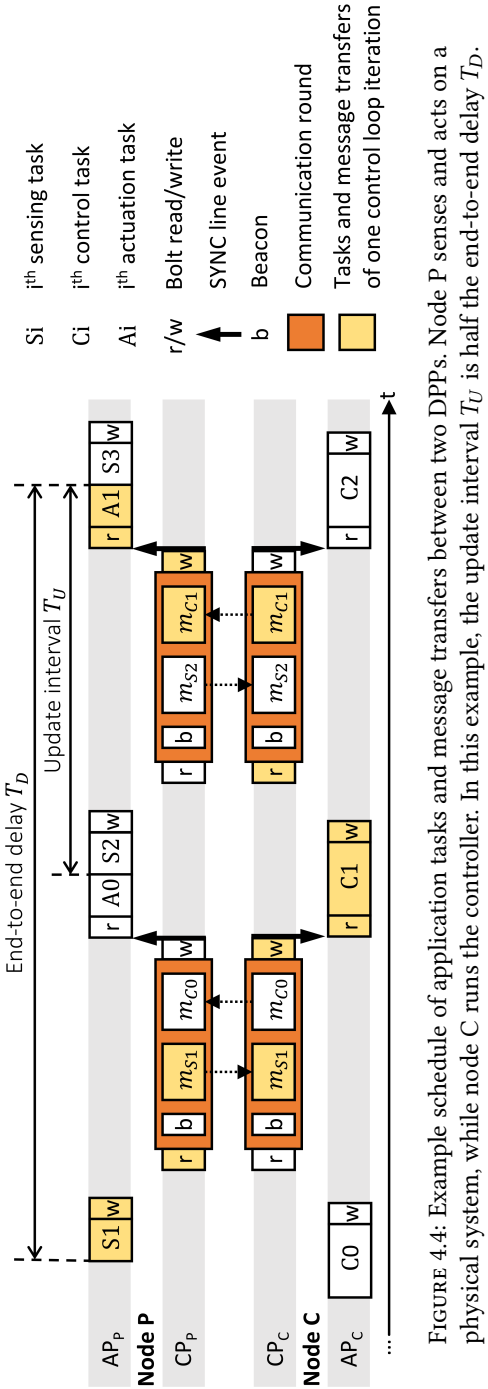


FIGURE 4.4: Example schedule of application tasks and message transfers between two DPPs. Node P senses and acts on a physical system, while node C runs the controller. In this example, the update interval T_U is half the end-to-end delay T_D.

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

This schedule resembles a pipelined execution, where in each communication round the last sensor reading and the next control signal (computed based on the previous sensor reading) are exchanged ($m_{S1}\, m_{C0}$, $m_{S2}\, m_{C1}$, ...). Note that while it is indeed possible to send the corresponding control signal in the same round ($m_{S1}\, m_{C1}$, $m_{S2}\, m_{C2}$, ...), doing so would increase the update interval T_U at least by the sum of the execution times of the control task, Bolt read, and Bolt write. For the example schedule in Figure 4.4, the update interval T_U is exactly half the end-to-end delay T_D.

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

To this end, we leverage Time-Triggered Wireless (TTW) [75], 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 4.2. Based on an integer linear program, it computes the communication schedule and all offsets mentioned above. TTW provides three important guarantees: a) a feasible solution is found if one exists, b) the solution minimizes the energy consumption for wireless communication, and c) the solution can additionally optimize user-defined metrics (e.g., minimize the update interval T_U as for the schedule in Figure 4.4).

4.4.4 Essential Properties and Jitter Analysis

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

- **P1** As analyzed below, for update intervals T_U and end-to-end delays T_D up to 100 ms, the worst-case jitter on T_U and T_D is bounded by ±50 µs. It holds $T_D = 2T_U$.
- **P2** Statistical analysis of millions of Glossy floods [194] and percolation theory for time-varying networks [79] have shown that the spatio-temporal diversity in a Glossy flood reduces the temporal correlation in the series of received and lost messages by a node, to the extent that the series can be safely approximated by an independent and identically distributed (i.i.d.) Bernoulli process. The success probability in real multi-hop networks is typically larger than 99.9 % [48].
- **P3** By provisioning for multi-hop many-to-all communication, arbitrary traffic patterns are efficiently supported.

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

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

$$|J| \leq 2\left(\hat{e}_{ref} + \hat{e}_{SYNC} + \widetilde{T}_{end}\,(\hat{\rho}_{AP} + \hat{\rho}_{CP})\right) + \hat{e}_{task} \tag{4.1}$$

where each term on the right-hand side of (4.1) is detailed below.

1) Time synchronization error between CPs. Using Glossy, each CP computes an estimate of the reference time [48] 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 [48], 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 $\hat{e}_{SYNC} = 1/f_{AP}$, where $f_{AP} = 48\,\text{MHz}$ is the frequency of APs clock.

3) Different clock drift at CPs and APs. The real offsets and durations of activities on the CPs and APs depend on the frequency of their clocks. Various factors contribute to different frequency drifts ρ_{CP} and ρ_{AP}, including the manufacturing process, ambient temperature, and aging effects. State-of-the-art clocks, however, drift by at most $\hat{\rho}_{CP} = \hat{\rho}_{AP} = 50\,\text{ppm}$ [95].

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

Using (4.1) 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 work requires $\widetilde{T}_{end} = T_D = 2T_U \leq 100\,\text{ms}$, which gives a worst-case jitter of $\pm 50\,\mu s$ on T_U and T_D, as stated by property **P1**.

4.5 Control Design and Analysis

Building on the design of the wireless embedded system and its properties **P1**–**P4**, this section addresses the design of the control system to accomplish goals **G4**–**G6**

from Section 4.3. Because the wireless system supports arbitrary traffic patterns (**P3**), various control tasks can be solved regardless of whether sensors, actuators, physical system(s), and controller(s) are co-located or spatially distributed. This includes typical single-loop tasks such as stabilization, disturbance rejection, or set-point tracking, as well as multi-agent scenarios such as synchronization, consensus, or formation control.

Here, we focus on remote stabilization and synchronization of multiple agents over wireless multi-hop networks as prototypical examples for both the single- and multi-agent case. For stabilization, modeling and control design are presented in Section 4.5.1 and Section 4.5.2, thus achieving **G5**. The stability analysis is provided in Section 4.5.3, which fulfills **G4**. Synchronization is discussed in Section 4.5.4, highlighting support for straightforward distributed control **G6**.

4.5.1 Model of Wireless Control System

We address the remote stabilization task depicted in Figure 4.5 (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 [111]. We consider stochastic LTI dynamics for the physical process

$$x(k+1) = Ax(k) + Bu(k) + v(k). \tag{4.2a}$$

This model describes the evolution of the system state $x(k) \in \mathbb{R}^n$ with discrete time index $k \in \mathbb{N}$ in response to the control input $u(k) \in \mathbb{R}^m$ and random process noise $v(k) \in \mathbb{R}^n$. As typical in the literature [11, 69], the process noise is modeled as an i.i.d. Gaussian random variable with zero mean and variance Σ_{proc}, $v(k) \sim \mathcal{N}(0, \Sigma_{\text{proc}})$, and captures, for example, uncertainty in the model.

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

$$y(k) = x(k) + w(k), \tag{4.2b}$$

with sensor measurements $y(k) \in \mathbb{R}^n$ and sensor noise $w(k) \in \mathbb{R}^n$, $w(k) \sim \mathcal{N}(0, \Sigma_{\text{meas}})$. If the complete state vector cannot be measured directly, it can typically be reconstructed via state estimation techniques [11].

The process model in (4.2) is stated in discrete time. This representation is particularly suitable here as the wireless embedded system offers a constant update interval T_U with worst case jitter of $\pm 50\,\mu\text{s}$ (**P1**), which can be neglected from controls perspective [28, p. 48]. Thus, $u(k)$ and $y(k)$ in (4.2) represent sensing and actuation at periodic intervals T_U, as illustrated in Figure 4.4.

As shown in Figure 4.5, 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

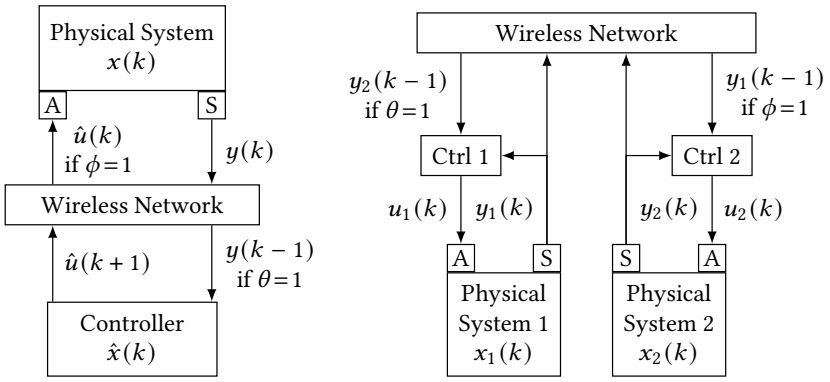


FIGURE 4.5: Wireless control tasks: stabilization (left) and synchronization (right). The feedback loop for stabilizing the physical system (left) is closed over the low-power wireless multi-hop network. This induces delays and message loss, which is captured by i.i.d. Bernoulli variables θ and ϕ. Two physical systems, each with a local controller (Ctrl), are synchronized over the wireless network (right).

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

$$u(k) = \phi\hat{u}(k) + (1-\phi)u(k-1). \tag{4.3}$$

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 4.5.4.

4.5.2 Controller Design

Designing a feedback controller for the system in (4.2), we proceed by first discussing state-feedback control for the nominal system (i.e., without delays, message loss, and noise), and then enhance the design to cope with the network and sensing imperfections.

Nominal design. Assuming ideal measurements, $y(k) = x(k)$ holds. A common strategy in this setting is static state-feedback control, $u(k) = Fx(k)$, where F is a

constant feedback matrix, which can be designed, for instance, via pole placement or methods from optimal control, such as the linear quadratic regulator (LQR) [7, 11]. Under the assumption of controllability [11], desired (in particular, stable) dynamics can be obtained for the state in (4.2a) in this way.

Actual design. We augment the nominal state-feedback design to cope with non-idealities, in particular, delayed measurements and message loss, as shown in Figure 4.5 (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:

$$\begin{aligned}\hat{x}(k) &= \theta A y(k-1) + (1-\theta)A\hat{x}(k-1) + B\hat{u}(k-1)\\ &= \theta A x(k-1) + (1-\theta)A\hat{x}(k-1) + B\hat{u}(k-1) + \theta A w(k-1),\end{aligned} \tag{4.4}$$

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 (4.4) 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,

$$\hat{u}(k+1) = F(A\hat{x}(k) + B\hat{u}(k)), \tag{4.5}$$

where F is as in the nominal design. The input $\hat{u}(k+1)$ is then transmitted over the wireless network (see Figure 4.5, left).

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

4.5.3 Stability Analysis

We now present a stability proof for the closed-loop system given by the dynamic system described in Section 4.5.1 and the proposed controller from Section 4.5.2. Because the model in Section 4.5.1 incorporates the physical process and the essential properties of the wireless embedded system, we achieve goal **G4**.

While the process dynamics in (4.2) are time invariant, message loss introduces time variation and randomness into the system dynamics. Therefore, we leverage stability results for linear, stochastic, time-varying systems [24]. For ease of exposition, we consider (4.2) without process and measurement noise (i.e., $v(k) = 0$ and $w(k) = 0$), and comment later on extensions. We first introduce required definitions and preliminary results, and then apply those results to our problem.

Consider the system

$$z(k+1) = \tilde{A}(k)z(k) \tag{4.6}$$

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

A common notion of stability for stochastic systems like the one in (4.6) is mean-square stability:

Definition 1 ([24, p. 131])**.** *Let $Z(k) := \mathbb{E}[z(k)z^{\mathrm{T}}(k)]$ denote the state correlation matrix. The system in (4.6) is* mean-square stable (MSS) *if $\lim_{k\to\infty} Z(k) = 0$ for any initial $z(0)$.*

That is, a system is called MSS if the state correlation vanishes asymptotically for any intial state. MSS implies, for example, that $z(k) \to 0$ almost surely as $k \to \infty$ [24, p. 131].

In control theory, linear matrix inequalities (LMIs) are often used as computational tools to check for system properties such as stability (see [24] for an introduction and details). For MSS, we employ the following LMI stability result:

Lemma 3 ([24, p. 131])**.** *System (4.6) is MSS if, and only if, there exists a positive definite matrix $P > 0$ such that*

$$\tilde{A}_0^{\mathrm{T}} P \tilde{A}_0 - P + \sum\nolimits_{i=1}^{N} \sigma_{p_i}^2 \tilde{A}_i^{\mathrm{T}} P \tilde{A}_i < 0. \tag{4.7}$$

We now apply this result to the system and controller from Section 4.5.1 and Section 4.5.2. The closed-loop dynamics are given by (4.2)–(4.5), which we rewrite as an augmented system

$$\underbrace{\begin{pmatrix} x(k+1) \\ \hat{x}(k+1) \\ u(k+1) \\ \hat{u}(k+1) \end{pmatrix}}_{z(k+1)} = \underbrace{\begin{pmatrix} A & 0 & B & 0 \\ \theta A & (1-\theta)A & 0 & B \\ 0 & \phi FA & (1-\phi)I & \phi FB \\ 0 & FA & 0 & FB \end{pmatrix}}_{\tilde{A}(k)} \underbrace{\begin{pmatrix} x(k) \\ \hat{x}(k) \\ u(k) \\ \hat{u}(k) \end{pmatrix}}_{z(k)}. \tag{4.8}$$

The system has the form of (4.6); the transition matrix depends on θ and ϕ, and thus on time (omitted for simplicity). We can thus apply Lemma 3 to obtain our main stability result, whose proof is given in Appendix 4.A.1.

Theorem 3. *The system (4.8) is MSS if, and only if, there exists a $P > 0$ such that (4.7) 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 3, we can analyze stability for any concrete physical system (4.2), a state-feedback controller F, and probabilities μ_θ and μ_ϕ. Searching for a $P > 0$ that satisfies the LMI in (4.7) can be done using efficient numerical tools based on convex optimization (e.g., [90]). If such a P is found, we have the stability guarantee (**G4**).

The stability analysis can be extended to account for process and measurement noise so that MSS then implies bounded $Z(k)$ (see [24, p. 138]). Moreover, other combinations of end-to-end delay T_D and update interval T_U are possible, including $T_D = nT_U$ ($n \in \mathbb{N}$). Also the sensor-to-controller and controller-to-actuator delays may be different.

4.5.4 Multi-Agent Synchronization

In distributed 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 [58, 105]. Which information a controller has access to depends on the traffic pattern and topology of the network. For instance, an agent may only be able to communicate with its nearest neighbor via point-to-point communication, or with other agents in a certain range. Property **P3** of the wireless embedded system in Section 4.4 offers a key advantage compared to these structures because every agent in the network has access to all information (except for rare message losses). We can thus carry out a centralized design, but implement the resulting controllers in a distributed fashion (cf. Figure 4.5, right). Such schemes have been used before for wired-bus networks (e.g., in [171]).

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 (4.2), but with possibly different dynamics parameters (A_i, B_i, …). We understand synchronization in this setting as the goal of having the system state of different agents evolve together as close as possible. That is, we want to keep the error $x_i(k) - x_j(k)$ between the states of systems i and j small. Instead of synchronizing the whole state vector, also a subset of all states can be considered. Synchronization of multi-agent systems is a common problem and also known as consensus or coordination [106].

We demonstrate feasibility of synchronization with multiple systems in Section 4.6.3. The synchronizing controller is based on an LQR [7]; details of the concrete design are given in Appendix 4.A.3.

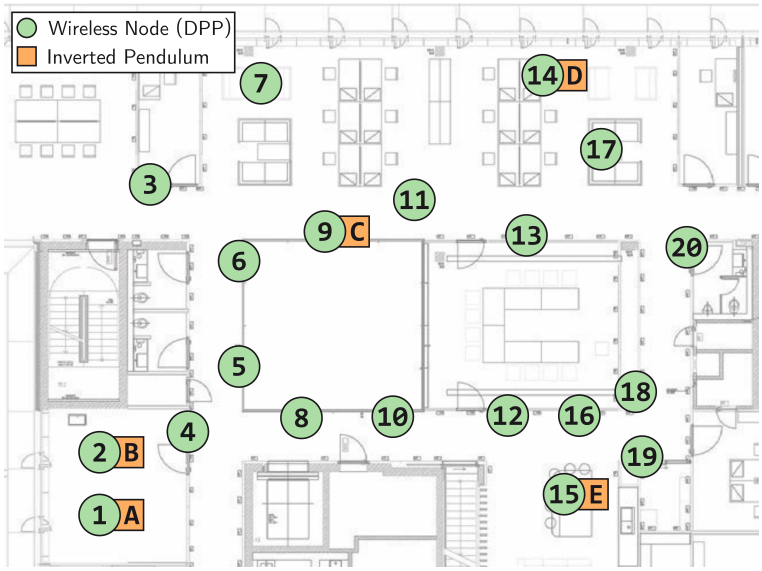


FIGURE 4.6: Layout of CPS testbed consisting of 20 DPP nodes that form a three-hop low-power wireless network and five cart-pole systems (two real ones attached to nodes 1 and 2, three simulated ones at nodes 9, 14, and 15).

4.6 Experimental Evaluation

This section uses measurements from a CPS testbed (see Figure 4.6) consisting of 20 wireless embedded devices (forming a three-hop network) and several cart-pole systems to evaluate the performance of the proposed wireless CPS design. Our experiments reveal the following key findings:

- We can safely stabilize two inverted pendulums via two remote controllers across the three-hop low-power wireless network.
- Using the same CPS design with a different control logic, we can reliably synchronize the movement of five cart-poles thanks to the support for arbitrary traffic patterns.
- Our system can stabilize an inverted pendulum at update intervals of 20–50 ms. Increasing the update interval decreases the control performance, but leads to significant energy savings on the wireless communication side.
- Our system is highly robust to message loss. Specifically, it can stabilize an inverted pendulum at an update interval of 20 ms despite 75 % i.i.d. Bernoulli losses and in situations with bursts of 40 consecutively lost messages.
- The measured jitter on the update interval and the end-to-end delay is less than ±25 µs, which validates our analysis of the theoretical worst-case jitter of ±50 µs from Section 4.4.4.

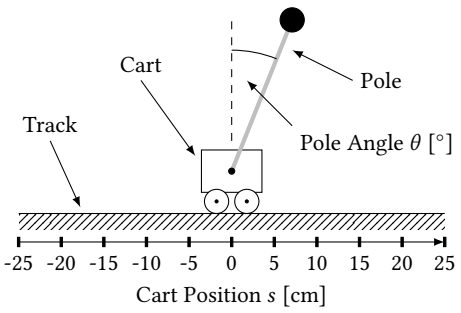


FIGURE 4.7: Schematic of a cart-pole system used in our testbed as physical systems. By controlling the force applied to the cart, the pole can be stabilized in the upright position around $\theta = 0°$.

4.6.1 Cyber-Physical Systems Testbed

Realistic CPS testbeds are essential for the validation and evaluation of CPS solutions [15, 101]. We developed the wireless CPS testbed depicted in Figure 4.6. It consists of 20 DPP nodes, two real physical systems (A and B), and three simulated physical systems (C, D, and E). The testbed is deployed in an office building and extends across an area of 15 m by 20 m. All nodes transmit at 10 dBm, which results in a network diameter of three hops. The wireless signals need to penetrate various types of walls, from glass to reinforced concrete, and are exposed to different sources of interference from other electronics and human activity.

We use *cart-pole systems* as physical systems. As shown in Figure 4.7, a cart-pole system consists of a cart that can move horizontally on a track and a pole attached to it via a revolute joint. The cart is equipped with a DC motor that can be controlled by applying a voltage to influence the speed and the direction of the cart. Moving the cart exerts a force on the pole and thus influences the pole angle θ. This way, the pole can be stabilized in an upright position around $\theta = 0°$, which represents an unstable equilibrium and is called the *inverted pendulum*. The inverted pendulum has fast dynamics, which are typical of real-world mechanical systems [23], and requires feedback with update intervals of tens of milliseconds.

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

The cart-pole system has a few constraints. Control inputs are capped at ±10 V. The track has a usable length of ±25 cm from the center (see Figure 4.7). Surpassing the track limits immediately ends an experiment. At the beginning of an experiment, we

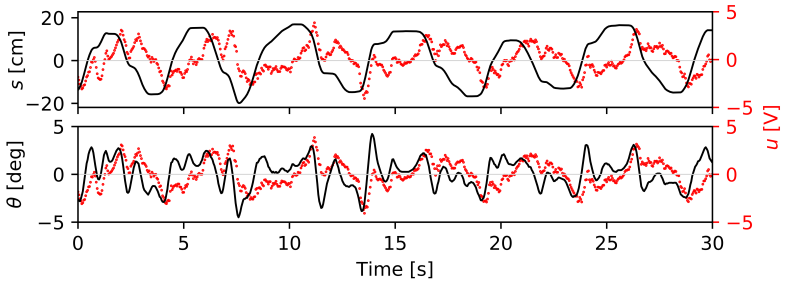


FIGURE 4.8: Cart position s, pole angle θ, and control input u of one cart-pole system when concurrently stabilizing two cart-pole systems over a multi-hop network at an update interval of 45 ms. The cart position and the pole angle always stay within safe regimes, and less than half of the possible control input is needed.

move the carts to the center and the poles in the upright position; then the controller takes over. Appendix 4.A.2 and Appendix 4.A.3 detail the implementation of the controllers for multi-hop stabilization and multi-hop synchronization, following the design outlined in Section 4.5.2 and Section 4.5.4.

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

4.6.2 Multi-Hop Stabilization

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

Setup. We use two controllers running on nodes 14 and 15 to stabilize the two real pendulums A and B at $\theta = 0°$ and $s = 0$ cm. Hence, there are two independent control loops sharing the same wireless network, and it takes in total six hops to close each loop. We configure the wireless embedded system and the controllers for an update interval of $T_U = 45$ ms. As per property **P2** and confirmed by our measurements discussed below, we expect a message delivery rate of at least 99.9 %. With this we have $\mu_\theta = \mu_\phi = 0.999$, and we can prove stability of the overall system using Theorem 3.

Results. The experimental results confirm the theoretical analysis: We are able to safely stabilize both pendulums over the three-hop wireless network, even while carrying around the controller because our design is independent of the network topology [2] (see Section 4.4.1). Figure 4.8 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.

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

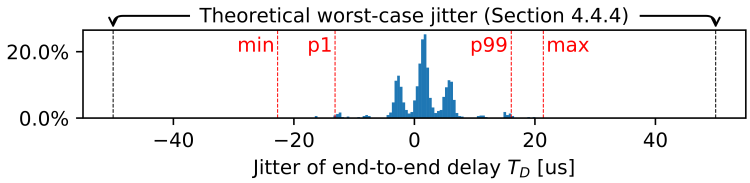


FIGURE 4.9: Distribution of the jitter on the end-to-end delay T_D; results for the update interval T_U are similar. The measurements are within the theoretical worst-case bounds determined in Section 4.4.4.

During the same experiment, we also use a logic analyzer to continuously measure the update interval T_U and the end-to-end delay T_D (see Figure 4.4). Figure 4.9 shows the measured jitter on T_D; the results for T_U look very similar. We see that the empirical results are well within the theoretical worst-case bounds, which validates our analysis in Section 4.4.4 and assumptions in Section 4.5.

4.6.3 Multi-Hop Synchronization

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

Setup. We use the two real pendulums A and B and the three simulated pendulums C, D, and E. The goal is to synchronize the cart positions of the five pendulums over the wireless multi-hop network, while each pendulum is stabilized by a local control loop. This scenario is similar to drone swarm coordination, where each drone stabilizes its flight locally, but exchanges its position with all other drones to keep a desired swarm formation [140]. In our experiment, stabilization runs with $T_U = 10$ ms, and nodes 1, 2, 9, 14, and 15 exchange their current cart positions every 50 ms.

Results. The left plot in Figure 4.10 shows the cart positions over time without synchronization. We see that the carts of the real pendulums move with different amplitude, phase, and frequency due to slight differences in their physics and imperfect measurements. The simulated pendulums, instead, are perfectly balanced and behave deterministically as they all start in the same initial state.

In the middle plot of Figure 4.10, we can observe the behavior of the pendulums when they synchronize their cart positions over the wireless multi-hop network. Now, all five carts move in concert. The movements are not perfectly aligned because, besides the synchronization, each cart also needs to locally stabilize its pole at $\theta = 0°$ and $s = 0$ cm. Since no message is lost during the experiment, the simulated pendulums all receive the same state information and, therefore, show identical behavior.

This effect can also be seen in our third experiment, shown in the bottom plot of Figure 4.10, where we hold pendulum A for some time at $s = -20$ cm. The other pendulums now have two conflicting control goals: stabilization at $s = 0$ cm and $\theta = 0°$, as well as synchronization while one pendulum is fixed at about $s = -20$ cm. As a result,

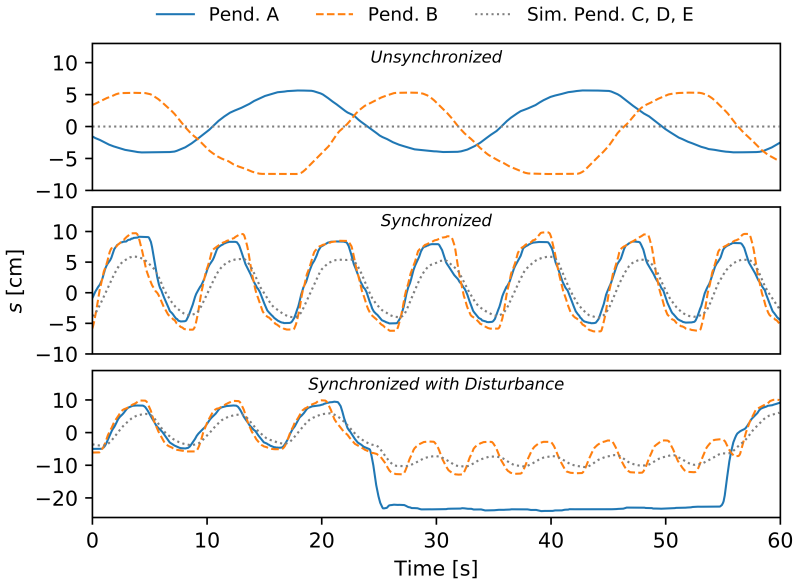


FIGURE 4.10: Cart positions of five cart-pole systems stabilized locally at an update interval of 10 ms and synchronizing their cart positions (middle and bottom plot) over the low-power wireless multi-hop network at an update interval of 50 ms. With synchronization enabled, all five carts move in concert and even try to mimic the temporary disturbance of pendulum A, shown in the bottom plot.

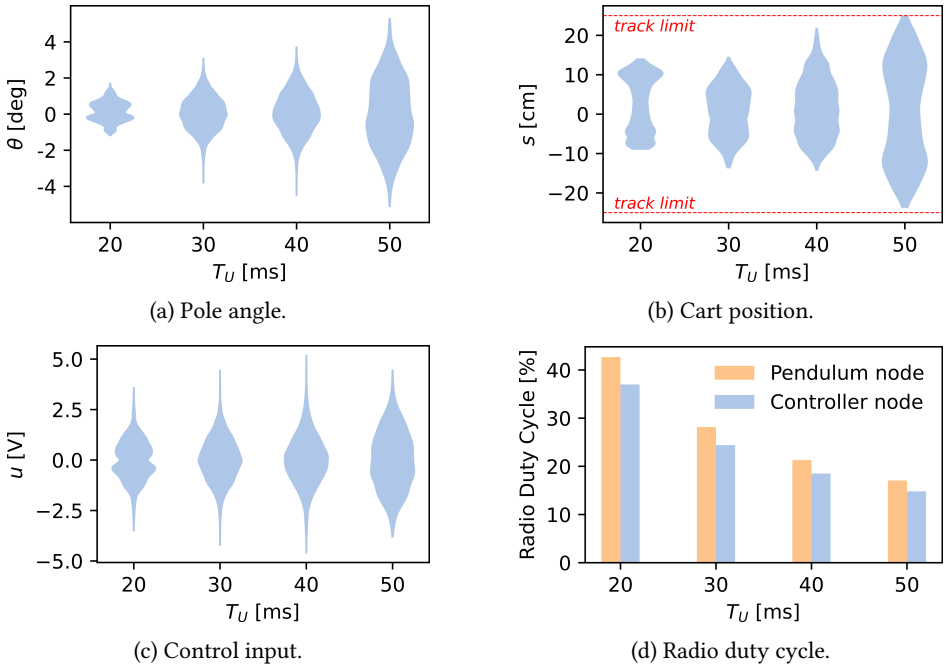
they all move toward this position and oscillate between $s = 0$ and $s = -20$ cm. Clearly, this experiment demonstrates that the cart-pole systems influence each other, which is enabled by the many-to-all communication over the wireless multi-hop network.

4.6.4 Impact of Update Interval

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

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

Results. Figure 4.11 shows control performance and radio duty cycle for different update intervals based on more than 12,500 data points. We see that a longer update interval causes larger pole angles and more movement of the cart. Indeed, the total distance the cart moves during an experiment increases from 3.40 m for 20 ms to 9.78 m for 50 ms. This is consistent with the wider distribution of the control input for longer update intervals. At the same time, the radio duty cycle decreases from 40 % for 20 ms to 15 % for 50 ms. Hence, there is a trade-off between control performance and energy efficiency, which may be exploited based on the application requirements.



(a) Pole angle.

(b) Cart position.

(c) Control input.

(d) Radio duty cycle.

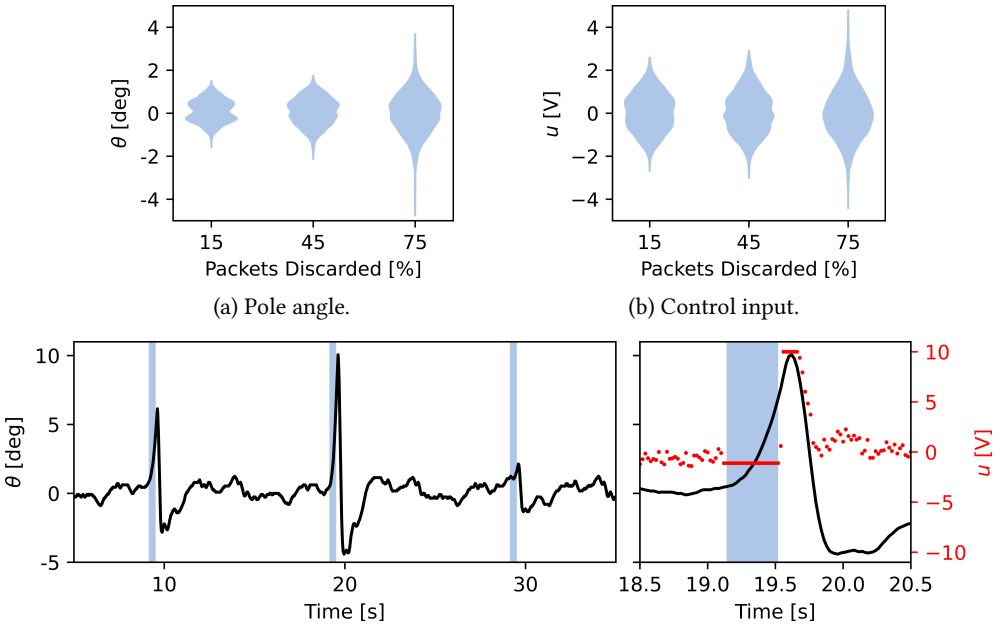
FIGURE 4.11: Distribution of control performance metrics and average radio duty cycle when stabilizing an inverted pendulum over low-power wireless at different update intervals. A larger update interval leads to larger pole angles and more movement of the cart, but also reduces the average radio duty cycle.

4.6.5 Resilience to Message Loss

Finally, we evaluate how control performance is affected by message loss, which is a well-known phenomenon in wireless networks [163].

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

Results. Figure 4.12a and Figure 4.12b show the distributions of the pole angle and the control input for varying i.i.d. Bernoulli message loss rates. We see that the control performance decreases for higher loss rates, but the pendulum can be stabilized even at a loss rate of 75 %. One reason for this is the short update interval. For example, losing 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.



(a) Pole angle.

(b) Control input.

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

FIGURE 4.12: Control performance and input when stabilizing one pendulum over wireless under artificially injected message loss, for i.i.d. Bernoulli losses (top), and for longer bursts of multiple consecutive losses (bottom). Depending on the update interval, the pendulum can be stabilized despite significant and bursty message loss, albeit with reduced performance.

Figure 4.12c plots the pole angle as a function of time for a burst length of 40 consecutively lost messages, with the right plot zooming into the time around the second burst phase. No control inputs are received during a burst, and depending on the state of the pendulum and the control input right before a burst, the impact of a burst may be very different as visible in Figure 4.12c. The magnified plot shows that the pole angle diverges from around 0° with increasing speed. As soon as the burst ends, the control input rises to its maximum value of 10 V in order to bring the pendulum back to a non-critical state, which usually takes 1–2 s. These results show that while property **P2** of our wireless embedded system design significantly simplifies control design and analysis, the overall system remains stable even if this property is dramatically violated, which is nevertheless very unlikely as demonstrated in prior work [79, 194].

4.A Control Details

In this appendix, we provide further details of the control design and analysis. We present the proof of Theorem 3, implementation details of the controllers we use for the multi-hop stabilization experiments, and outline the approach to multi-agent synchronization.

4.A.1 Proof of Theorem 3

For clarity, we reintroduce time index k for θ and ϕ. Following a similar approach as in [148], 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 $\mathrm{Var}[\delta_\theta(k)] = \sigma^2_{p_1}$, and similarly for $\delta_\phi(k)$. Employing this transformation, $\tilde{A}(k)$ in (4.8) 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 3. Thus, all properties of (4.6) are satisfied, and Lemma 3 yields the result.

4.A.2 Stabilizing Controllers

For the stability experiments of Section 4.6.2, we employ the design outlined in Section 4.5.2. The system matrices A and B of the cart-pole system that are used for predictions and nominal controller design are given by the manufacturer in [141]. The nominal controller is designed for an update interval $T_U = 40$ ms via pole placement, and we choose F such that we get closed-loop eigenvalues at 0.8, 0.85, and 0.9 (twice). In experiments with update intervals different from 40 ms, we adjust the controller to achieve similar closed-loop behavior.

To derive more accurate estimates of the velocities, filtering can be done at higher update intervals than communication occurs. For the experiments in Section 4.6, estimation and filtering occur at intervals between 10 ms and 20 ms, depending on the experiment.

4.A.3 Synchronization

For simplicity, we consider synchronization of two agents in the following, but the approach directly extends to more than two, as we show in the experiments in Section 4.6.3.

We consider the architecture in Figure 4.5, where each physical system is associated with a local controller that receives local observations directly, and observations from other agents over the network. We present an approach based on an optimal LQR [7] to design the synchronizing controllers. We choose the quadratic cost function

$$J = \lim_{K\to\infty} \frac{1}{K} \mathbb{E}\Big[\sum_{k=0}^{K-1} \sum_{i=1}^{2} \Big(x_i^{\mathrm{T}}(k) Q_i x_i(k) + u_i^{\mathrm{T}}(k) R_i u_i(k) \Big) + (x_1(k) - x_2(k))^{\mathrm{T}} Q_{\mathrm{sync}} (x_1(k) - x_2(k)) \Big] \tag{4.9}$$

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

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

Thus, the problem is in standard LQR form and can be solved with standard tools [7]. The optimal stabilizing controller that minimizes (4.9) has the structure $u_1(k) = F_{11}x_1(k) + F_{12}x_2(k)$ and $u_2(k) = F_{21}x_1(k) + F_{22}x_2(k)$; that is, agent 1 ($u_1(k)$) requires state information from agent 2 ($x_2(k)$), and vice versa. Because of many-to-all communication, the wireless embedded system directly supports this (as well as any other possible) controller structure (**P3**).

As the controller now runs on the node that is co-located with the physical process, local measurements and inputs are not sent over the wireless network and the local sampling time can be shorter than the update interval of the network over which the states of other agents are received. While the analysis in Section 4.5.3 can be generalized to the synchronization setting, a formal stability proof is beyond the scope of this work. In general, stability is less critical here because of shorter update intervals in the local feedback loop.

For the synchronization experiments in Section 4.6.3, we choose Q_i in (4.9) for all pendulums as suggested by the manufacturer in [141] and set $R_i = 0.1$. As we here care to synchronize the cart positions, we set the first diagonal entry of Q_{sync} to 5 and all others to 0.

Postscript

We have presented a CPS design that enables, for the first time, fast feedback control over low-power wireless multi-hop networks with update intervals of 20–50 ms. By contrast, existing feedback control approaches are either limited to small single-hop networks or systems with slow dynamics, where update intervals of several seconds significantly relax timing and reliability requirements. In our co-design approach, we tame network imperfections to the extent possible, take the resulting properties of the communication network into account in the control design, and carefully coordinate all run-time tasks to meet real-time demands. Thus, we can formally prove closed-loop stability of the entire CPS. We validate our findings with experiments on a novel CPS testbed featuring multiple physical systems. By demonstrating how feedback loops can be closed quickly and reliably over large distances, we are taking an important step toward realizing the CPS vision.

Furthermore, we have extended the adaptivity of our design in a follow-up work by introducing so-called mode changes. Because CPS potentially have to perform various tasks at run-time, they may switch between different modes of operation, for example, to realize a highly individual manufacturing process. Our extended system can dynamically change between different application tasks executing on spatially distributed embedded devices, while closed-loop stability is provably guaranteed even across mode changes. The corresponding publication is not part of this thesis:

Dominik Baumann[⋆], Fabian Mager[⋆], Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. "Fast Feedback Control over Multi-hop Wireless Networks with Mode Changes and Stability Guarantees." *ACM Transactions on Cyber-Physical Systems*, 4(2):18:1–18:32, 2019. doi:10.1145/3361846

[⋆] Both authors contributed equally to this work.

In addition to the laboratory experiments, we successfully demonstrated our system and testbed in other locations, including a public demonstration at the CPS-IoT Week 2019 in Montreal:

Fabian Mager[⋆], Dominik Baumann[⋆], Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. "Demo Abstract: Fast Feedback Control and Coordination with Mode Changes for Wireless Cyber-Physical Systems." In *International Conference on Information Processing in Sensor Networks (IPSN)*, pages 340–341, Montreal, Quebec, Canada, 2019. doi:10.1145/3302506.3312483 **(Best demo award)**

[⋆] Both authors contributed equally to this work.

5

Control-Guided Communication: Efficient Resource Arbitration and Allocation in Multi-Hop Wireless Control Systems

Preface

The CPS we have presented in the previous chapter uses a classical time-triggered control design, where each agent transmits information periodically; thus, the bandwidth demands increase linearly with the number of agents. It is expected that the number of agents in emerging CPS applications will increase significantly, making the limited communication bandwidth a highly contested resource. Modern ETC and STC designs reduce the generated traffic by transmitting data only when needed, which enables considerable energy savings and better scalability. However, no prior work can transfer these benefits to multi-hop networks and enable freed-up bandwidth to be reallocated to other agents because communication resources must be allocated ahead of time.

In this chapter, we propose *control-guided communication*, a novel co-design that integrates distributed STC and wireless multi-hop communication to close this gap. The core idea is that the control system predicts transmission demands ahead of time and informs the communication system such that communication resources can be allocated accordingly. We validate our approach on a CPS testbed and demonstrate how communication resources can be dynamically saved or reallocated to other traffic. Thus, we achieve a real end-to-end benefit, including control *and* communication, which enables more efficient and adaptive wireless CPS.

Table 5.1: Qualitative comparison of prior and our work on integrating STC with wireless communication, evaluated through real-world experiments.

Work	Fast update intervals	Multi-hop	Energy savings	Reallo-cation	Distributed implementation
[10]	✗	✗	✓	✓	✗
[9]	✗	✗	✓	✓	✗
[110]	✗	✓	✓	✗	✗
[151]	✓	✗	✓	✗	✗
[152]	✓	✗	✓	✗	✗
This	✓	✓	✓	✓	✓

5.1 Introduction

The unparalleled flexibility and cost efficiency when closing feedback loops over wireless networks enables many CPS applications. For instance, in a smart factory, plants are controlled via remote controllers, mobile robots interact with the plants, and distributed sensors provide additional measurements. Another example is drones regularly exchanging data to fly in formation. These and other applications demand wireless *multi-hop* communication to cover large distances and *fast* update intervals of tens of milliseconds to keep up with the dynamics of the systems to be controlled [4].

Challenges. Fast feedback control over wireless multi-hop networks is challenging owing to the inherent imperfections of wireless networks, such as transmission delays and message loss. Moreover, the limited network bandwidth can lead to congestion when many agents need to communicate at the same time, and wireless radios draw considerable power, which is a major concern for embedded sensors and mobile devices that must be untethered and thus powered by batteries. For these reasons, adaptive schemes are needed where agents use the network only when necessary to *save energy*, and available resources are *reallocated* at run-time to serve those in need.

To use the limited bandwidth and energy more efficiently, ETC and STC methods have been developed [63, 124]. Unlike periodic control, in ETC and STC the decision whether to communicate or not is based on events, such as an error exceeding a threshold. ETC *instantaneously* decides whether to communicate, leaving no time to save energy or reallocate bandwidth in case of a negative triggering decision. STC, instead, decides *ahead of time* about the next triggering instant. However, to utilize freed resources (e.g., to serve traffic from additional remote sensors), an integration of STC designs and wireless communication protocols is required. Moreover, such co-design approaches must be evaluated on real CPS testbeds to establish trust in feedback control over wireless [101]. While a large body of work on STC exists (see [63, 121, 124, 174, 178] and the references therein), the integration of STC designs with wireless protocols including an experimental evaluation has rarely been considered. The few exceptions are listed in Table 5.1 and discussed next.

Prior work. Existing approaches integrating STC and wireless communication target remote control, for example, of a double-tank process [9, 10], a simulated load-positioning system [110], or a mobile robot [151]. Coordination in multi-robot systems has been studied in [152], but the control commands are computed by a central entity, so the implementation is not distributed. All works show that STC allows for solving the control task with less communication than periodic control, enabling significant energy savings. However, reallocation of freed resources has only been demonstrated in [9, 10], for single-hop networks and update intervals of a few seconds. In fact, STC over a wireless multi-hop network has only been shown in [110], with an update interval of 1 s.

In summary, no solution exists that provides energy savings *and* reallocation of freed resources for the control of systems at fast update intervals over multi-hop networks. Moreover, no work has shown a distributed implementation of a STC law, where agents locally use information obtained over the network to solve a common control task. However, a complete solution is needed to enable novel applications, such as collaborative multi-robot swarms for future smart production systems.

Contribution. We present a co-design of control and communication for multi-hop wireless networks that fills this gap. Our approach arbitrates the available communication bandwidth among different types of traffic from any entity in the network, while simultaneously shutting down resources completely to save energy when neither the control system nor any other entity needs the full bandwidth. We evaluate the approach on a three-hop CPS testbed with multiple physical systems [15], demonstrating improved resource efficiency at high control performance for update intervals below 100 ms.

At the heart of our solution is the novel concept of *control-guided communication*: The control system informs the communication system *at run-time* about its resource requirements, and the communication system leverages this information to dynamically allocate or shut down resources. Concretely, we consider the setup depicted in Figure 5.1. Each agent uses STC to decide at the current communication instant when it will communicate next. The agent piggybacks the decision of its self trigger onto the messages it sends. The network manager uses this information as input when dynamically computing the communication schedule at run-time. For example, when some agents do not need to communicate, their share of the bandwidth can be reallocated to serve other traffic (e.g., from remote sensors) or can be shut down to conserve energy. The concrete scheduling policy is an exchangeable component of our design and can be adapted to the application requirements.

In essence, we make the following two main contributions:

- We propose control-guided communication, a tight integration of STC and wireless multi-hop communication in which the control system informs the network at run-time about future communication demands to enable both energy savings *and* reallocation of network bandwidth.

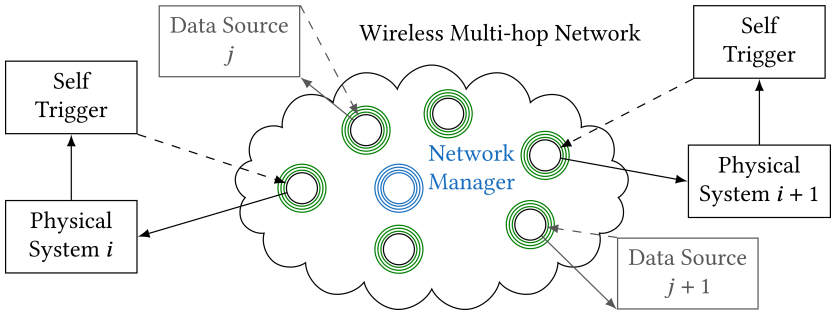


FIGURE 5.1: We consider multiple physical systems connected over a wireless multi-hop network. Each system is associated with a self trigger that computes at the current communication instant when it needs to communicate the next time. This information is piggybacked onto the message it sends. The network manager uses this information to compute a communication schedule respecting these demands and, if possible, saves energy or reallocates bandwidth to additional systems or data sources.

- Using experiments on a real CPS testbed with five inverted pendulums, we are the first to demonstrate distributed STC over wireless multi-hop networks with update intervals below 100 ms, while showing energy savings of up to 87 % compared to the periodic baseline.

5.2 Problem Setting

We consider N physical systems connected over a wireless multi-hop network, as shown in Figure 5.1. Each agent is modeled as a stochastic, linear, and time-invariant system

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k) + v_i(k), \tag{5.1}$$

with state $x_i(k) \in \mathbb{R}^n$, input $u_i(k) \in \mathbb{R}^m$, and $v_i(k) \in \mathbb{R}^n$ a Gaussian random variable with zero mean and variance Σ_i, capturing process noise. We assume each agent has a local controller that receives local observations directly, but also needs information from other agents for distributed control.

There are various methods to design distributed controllers (see, e.g., [105]). In this work, we adopt an approach based on the LQR [7]. Using augmented states $\tilde{x}(k) = (x_1(k), \ldots, x_N(k))^\mathsf{T}$ and inputs $\tilde{u}(k) = (u_1(k), \ldots, u_N(k))^\mathsf{T}$, we define the cost function

$$J = \lim_{k\to\infty} \frac{1}{K} \mathbb{E}[\tilde{x}^\mathsf{T}(k) Q \tilde{x}(k) + \tilde{u}^\mathsf{T}(k) R \tilde{u}(k)], \tag{5.2}$$

with positive definite weight matrices Q and R. The optimal stabilizing controller that minimizes (5.2) is of the form $u_i(k) = \sum_j F_{ij} x_j(k)$, where F_{ij} denotes entry (i, j) of the feedback matrix F. That is, to implement this controller, each agent needs information from all other agents, which is sent over the wireless multi-hop network. To provide high-performance control while efficiently using limited network bandwidth and energy resources, the system must meet several requirements:

- For coordination, the agents need to exchange data; in particular, for optimal control according to (5.2), all agents need to communicate with one another (all-to-all).
- Wireless multi-hop communication must be reliable and fast to support feedback control of physical systems with fast dynamics; we target mechanical systems requiring update intervals on the order of tens of milliseconds [4].
- The network must arbitrate among multiple types of data traffic as determined by the communication schedule, while always giving highest priority to control traffic.
- If some fraction of the bandwidth is not allocated to any entity, this resource should be shut down to save energy.

5.3 Co-Design Approach

The main goal of this work is to facilitate high-performance distributed control across multi-hop wireless networks with highly adaptive resource arbitration and allocation to support multiple traffic types and save unused resources. Prior work failed to reach this goal because the many imperfections of wireless systems, such as time-varying end-to-end delays and limited throughput, complicate the control design and make it difficult to quickly coordinate the system-wide operation and resource usage based on the current control-traffic demands.

To tackle this issue, we propose a novel co-design approach that integrates the control and communication systems in two ways. First, the *design* of the communication system tames network imperfections as much as possible, and the control system accounts for the emerging key properties and remaining imperfections. Second, during *operation*, the control system reasons about its future communication demands and informs the communication system accordingly. The communication system, on the other hand, adapts to these demands by arbitrating the available bandwidth among different types of traffic and by shutting down resources completely to save energy when neither the control system nor any other participant needs the full bandwidth. We call this concept *control-guided communication*, which we detail in the following two sections.

In addition, our wireless communication system provides fast and reliable many-to-all communication among any set of agents, even when the agents are mobile and thereby causing the network topology to change continuously. This feature is a key difference to traditional wireless communication systems, such as WirelessHART, and makes our co-design approach directly applicable to solve various kinds of distributed control problems that may be stated in the form of a cost function (5.2).

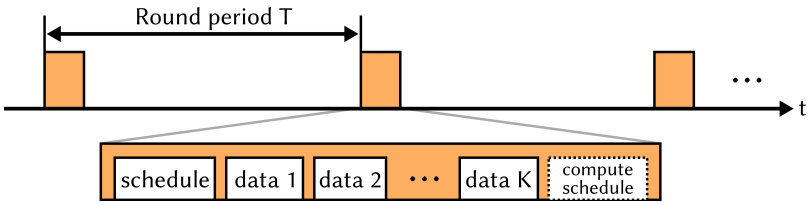


FIGURE 5.2: Time-triggered operation of the low-power wireless multi-hop protocol. Communication occurs in rounds with a constant period T. Each round consists of a schedule slot and up to K data slots. The schedule slot serves to inform all nodes about subsequent data slots in the round, including the number and allocation of control or other messages. The network manager computes the next schedule after the last data slot.

5.4 Wireless Communication System Design

We first describe the design of the wireless communication system, and detail the control design based on the emerging properties in the next section. The wireless system builds on the periodic design presented in Chapter 4 and consists of three elements, where 2) is significantly modified and 3) is a new component:

1) a *hardware platform* enabling a predictable and efficient execution of all control tasks and message transfers;

2) a *multi-hop wireless protocol* that provides many-to-all communication with minimal, bounded end-to-end delay;

3) an *online scheduler* that dynamically assigns bandwidth to each agent based on its communication requirements.

Hardware platform. We use a DPP where sensing, actuation, and control execute on an AP (MSP432P401R, 32-bit, 48 MHz) and the wireless multi-hop protocol executes on a CP (CC430F5147, 16-bit, 13 MHz). The processors communicate through the Bolt interconnect [166], which provides bounded worst-case execution times for the bidirectional exchange of messages between both processors. In this way, control and communication can efficiently execute in parallel and never interfere with each other, providing timing predictability.

Multi-hop wireless protocol. The CP of every DPP in the network runs a multi-hop protocol, whose design is inspired by a new breed of protocols that exploit ST-based flooding for highly reliable and efficient communication. As shown in Figure 5.2, using our protocol, communication occurs in *rounds* of equal duration that repeat with a constant period T. Each round consists of a sequence of non-overlapping *slots*. In each slot, one node is allowed to initiate a Glossy flood [48] to send a message to all other nodes. Glossy achieves the theoretical minimum latency for flooding a message in a multi-hop network using half-duplex radios, and provides a reliability above 99.9 % in real-world scenarios [48, 49]. In fact, Glossy's reliability can be pushed beyond 99.9999 % by letting nodes transmit more often during a flood, and it time-synchronizes all nodes to within sub-microsecond accuracy at no additional cost [48].

Any node in the network can serve as the designated *network manager* that uses the first slot in a round to flood the *schedule*. The schedule informs all other nodes about the number of data slots in the round (up to K) and the allocation of nodes to these data slots. The transmitted messages carry, for example, high-priority control information from agents or lower-priority data from other nodes, such as measurements from a remote sensor or information about a node's health status (e.g., its battery's state of charge). When sending a message, a node also piggybacks information about its future communication demands; if the network manager does not receive a message, it assumes that the respective node needs to transmit in the next round. Based on all demands, the network manager computes the schedule for the next round after the last data slot.

Online scheduler. To this end, the network manager maintains a list of unserved communication demands, and allocates up to K nodes to the data slots in the next round according to a *scheduling policy*. The scheduling policy can be adjusted to meet different application requirements. As an illustrative example, we design in this work a new policy that aims to strike a balance between resource efficiency and accommodating lower-priority messages next to control traffic. Specifically, if there are free data slots after assigning all nodes with pending control messages in the next round, we allocate one of the free data slots to a node for sending some other message, such as sensor or status information. The next node to send such message is chosen in a round-robin fashion. Any other free slot is left empty. Since nodes have their radios on only during allocated slots and off otherwise, this example policy illustrates that our wireless communication system allows for both arbitrating bandwidth among different traffic types and not allocating resources at all to save energy, as demonstrated in Section 5.6.

Key properties. Our wireless system design provides highly reliable, efficient many-to-all communication, system-wide time synchronization, and adapts at run-time to the nodes' communication demands. Due to the time synchronization, we can schedule control and communication tasks such that the jitter on the update interval and end-to-end delay is less than $\pm 50\,\mu s$, as formally and experimentally validated in Chapter 4.

5.5 Self-Triggered Control Design

We now detail the control design, first our approach to distributed control and then our self-triggered design.

5.5.1 Distributed Control

The wireless communication system provides a constant update interval T as the jitter is negligible for the considered scenarios. We thus set one discrete time step in (5.1) to T and data that is sent over the network is delayed by one time step. Moreover, the

many-to-all communication scheme ensures that information can be received by all agents in the network. This greatly facilitates control design as essentially arbitrary information patterns can be implemented. For example, this allows for implementing a (centralized) optimal controller in a distributed fashion as we show in this chapter. Given the high reliability of the wireless embedded system, we assume that data that are sent over the network are received by all agents.

As an example for distributed control, we consider synchronization of multiple agents through an LQR design as in (5.2). For ease of presentation, we outline the approach for the two-agent case, but it also extends to multiple agents as shown in Section 5.6. We choose the quadratic cost function

$$J = \lim_{K\to\infty} \frac{1}{K} \mathbb{E}\Big[\sum_{k=0}^{K-1} \sum_{i=1}^{2} \Big(x_i^{\mathsf{T}}(k) Q_i x_i(k) + u_i^{\mathsf{T}}(k) R_i u_i(k) \Big) + (x_1(k) - x_2(k))^{\mathsf{T}} Q_{\text{sync}} (x_1(k) - x_2(k)) \Big], \tag{5.3}$$

that is, we penalize deviations between $x_1(k)$ and $x_2(k)$ through the positive definite weight matrix Q_{sync}, as well as deviations from the equilibrium ($Q_i > 0$) and high control inputs ($R_i > 0$). Using augmented states as in (5.2), the term in the summation over k becomes

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

As discussed in Section 5.2, solving the optimal control problem then leads to a feedback controller that has the form $u_1(k) = F_{11}x_1(k) + F_{12}x_2(k)$, that is, agent 1 needs information from agent 2. We account for this by letting agent 2 send $u_{12}(k) = F_{12}x_2(k)$ over the network. Thus, agent 1's control input consists of $u_{11}(k) = F_{11}x_1(k)$, which it can compute using its local observations, and $u_{12}(k)$, which it receives over the network. We can thus define the closed-loop matrix $\tilde{A}_1 = A_1 + B_1F_{11}$ and (5.1) then reads as follows

$$x_1(k+1) = \tilde{A}_1 x_1(k) + B_1 u_{12}(k) + v_1(k). \tag{5.4}$$

5.5.2 Self-Triggered Approach

Different STC designs have been proposed and are conceivable to realize control-guided communication. We use a design that exploits ideas from previous work on state estimation [170]. Instead of sending states as in [170], we consider the communication of control inputs. Specifically, rather than sending its entire state, agent 2 only sends the input $u_{12}(k)$ that is needed by agent 1. In case of no communication, agent 1 keeps applying $u_{12}(k_\ell)$, where k_ℓ is the last time step at which the input $u_{12}(k)$ was sent. We trigger communication based on the error $e_{12}(k) := u_{12}(k) - u_{12}(k_\ell)$ as follows

$$\gamma_2(k) = 1 \iff (e_{12}(k))^{\mathsf{T}} e_{12}(k) > \delta. \tag{5.5}$$

Here, $\gamma_2(k)$ is a binary variable, denoting whether agent 2 communicates $u_{12}(k)$ ($\gamma_2(k) = 1$) or not ($\gamma_2(k) = 0$), while δ defines the designer's trade-off between saving communication (large δ) and keeping the error to a minimum (small δ).

If we directly implement (5.5), agent 2 instantaneously decides on whether to transmit $u_{12}(k)$ to agent 1. In case of a negative triggering decision, there is no possibility to reallocate bandwidth and hence freed resources remain unused. To overcome this problem, we use a self-triggered strategy. Whenever an agent communicates, it already decides when to communicate next. To this end, we predict the evolution of the error and look for the smallest $M > 1$ such that

$$\mathbb{E}\big[(e_{12}(k+M))^\mathsf{T} e_{12}(k+M)|\mathcal{D}_2(k)\big] > \delta \tag{5.6}$$

and set $\gamma(k+M-1) = 1$. Here, $\mathcal{D}_2(k)$ describes the data agent 2 collected until time step k, that is, its local states x_2 and the inputs u_2 and u_{12} that it has applied and sent so far, respectively. The rationale behind this triggering rule is as follows: Information that is sent over the network is delayed by one discrete time step. The inequality in (5.6) tells us that the error exceeds, in expectation, the threshold δ in M time steps. We thus seek to communicate next in $M-1$ time steps such that the new input arrives in M time steps, which is exactly when we expect the error to exceed the threshold.

The exact computation of (5.6) is complicated by the fact that the input $u_{21}(k)$ is not available at all times at agent 2. To derive the triggering law, we assume $u_{21}(k)$ is known and then comment on how we approximate it to yield a tractable implementation. Based on this, we get the error distribution

$$f(e_{12}(k+M)|\mathcal{D}_2(k)) = \mathcal{N}(\hat{e}_{12}(k+M|k), P_2(k+M|k)), \tag{5.7}$$

with mean $\hat{e}_{12}$ and variance P_2 given as

$$\begin{aligned}&\hat{e}_{12}(k+M|k) = \\ &F_{12}(\tilde{A}_2^M x_2(k) + \sum_{i=0}^{M} \tilde{A}_2^{M-i} B_2 u_{21}(k+i)) - u_{12}(k)\end{aligned} \tag{5.8a}$$

$$P_2(k+1|k) = F_{12}^\mathsf{T}(\tilde{A}_2^\mathsf{T} P_2(k|k)\tilde{A}_2 + \Sigma_2)F_{12}. \tag{5.8b}$$

Equations (5.8) are standard open-loop state and covariance predictions of the system in (5.4), so the derivations follow from Kalman filter theory [8, p. 111].

Given this error distribution, we can now, using $\mathbb{E}[e^\mathsf{T}e] = \|\mathbb{E}[e]\|^2 + \mathrm{Tr}(\mathrm{Var}[e])$, solve for the triggering rule (5.5): At every communication instant, find the smallest $M > 1$

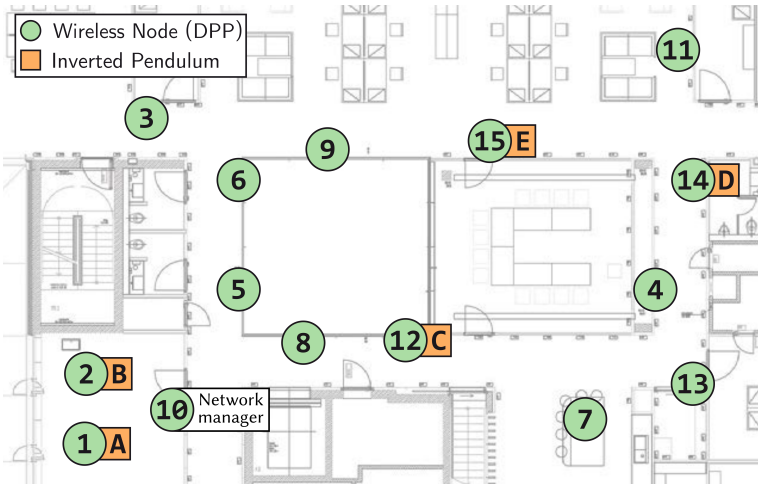


FIGURE 5.3: CPS testbed with 15 wireless DPP nodes and five cart-pole systems (A and B are real systems; C, D, and E are simulated systems). The network has a diameter of three hops. Node 10 is the network manager.

such that

$$\|F_{12}(\tilde{A}_2^M x_2(k) + \sum_{i=0}^{M} \tilde{A}_2^{M-i} B_2 u_{21}(k+i)) - u_{12}(k)\|^2 + \mathrm{Tr}(F_{12}^\intercal(\tilde{A}_2^\intercal P_2(k+M|k)\tilde{A}_2 + \Sigma_2)F_{12}) > \delta, \tag{5.9}$$

with Tr the trace of a matrix.

So far, we assumed that agent 2 has knowledge about the future development of $u_{21}(k+i)$, which does not hold in practice. Because agent 2 has no information about the current state of agent 1 and hence cannot infer the future development of $u_{21}(k+i)$, it approximates $u_{21}(k+i)$ as $u_{21}(k+i) = u_{21}(k)\,\forall i \in [0, M)$. With this, the input $u_{21}(k+i)$ in (5.8a) and (5.9) effectively becomes a constant.

We note that one way to let agent 1 reason about agent 2's state would be to send the entire state $x_2(k)$ instead of the control input $u_{12}(k)$. Agent 1 could use this state to compute $u_{12}(k)$ and to predict the evolution of agent 2's state. This, however, incurs higher communication demands at each instant as the state is typically of higher dimension than the input.

5.6 Experimental Evaluation

We evaluate our approach using experiments on a real CPS testbed [15] shown in Figure 5.3. It consists of 15 wireless DPP nodes and five cart-pole systems (or pendulums), where A and B are real systems and C, D, and E are simulated systems. The nodes are distributed in an office space of about 15 m by 20 m, and transmit at −6 dBm and 250 kbps in the 868 MHz band, forming a three-hop wireless network.

5.6.1 Scenario and Metrics

Scenario. The control task of each pendulum is to locally stabilize itself and to synchronize its cart position with all others. Since each system has access to its local state $x_i(k)$, we can run the local feedback loop at a faster update interval than communication over the network occurs. Here, we choose an update interval of 10 ms for the local loop. Control inputs $u_{ij}(k)$ of the other agents are communicated over the wireless multi-hop network, where the exchange of all control inputs takes 50 ms (i.e., one communication round with up to $K = 5$ data slots and 4 B per agent). We use the scheduling policy outlined in Section 5.4. To challenge the synchronization of the cart positions, we apply a sine distortion signal (3.6 s period with an amplitude of ±5 V) to the control input of pendulum B.

The controllers are designed as described in Section 5.5. We use the same model for the cart-pole system as in Chapter 4 and also adopt the Q_i matrices used for periodic synchronization. For Q_{sync}, we set the first diagonal entry to 20 and all other entries to zero to express our desire to synchronize the cart positions. Further, we choose $R_i = 0.01$ for all systems.

Metrics. Our evaluation uses the following metrics:

- *root mean square of the synchronization error (RMSE)* computed based on the cart positions of all pendulums in an experiment as a measure of control performance;
- *utilization* of the available data slots during each round, broken down into free slots (radio off), slots used for control traffic, and slots used for additional (other) traffic;
- *radio duty cycle*, the fraction of time a node has its radio on, which is a widely used metric in the low-power wireless networking literature (see, e.g., [49, 56]) for quantifying communication energy cost.

In the following, we first illustrate the run-time operation of our co-designed wireless control system in a real experiment, and then evaluate the trade-off among control performance, communication energy cost, and serving additional traffic as a function of the triggering threshold.

5.6.2 Efficient Resource Arbitration and Allocation

Figure 5.4 shows a real trace of the control performance (top) and the slot utilization in each communication round (bottom) over time for a triggering threshold of $\delta = 0.03$. Looking at the utilization, we see that, on average, less than one third of the available bandwidth is needed for control traffic. Our co-design approach effectively uses the freed bandwidth to schedule additional traffic (here at most one slot per round according to the example scheduling policy from Section 5.4) *and* to shut down the remaining bandwidth completely. During the many free slots all nodes have their radios turned

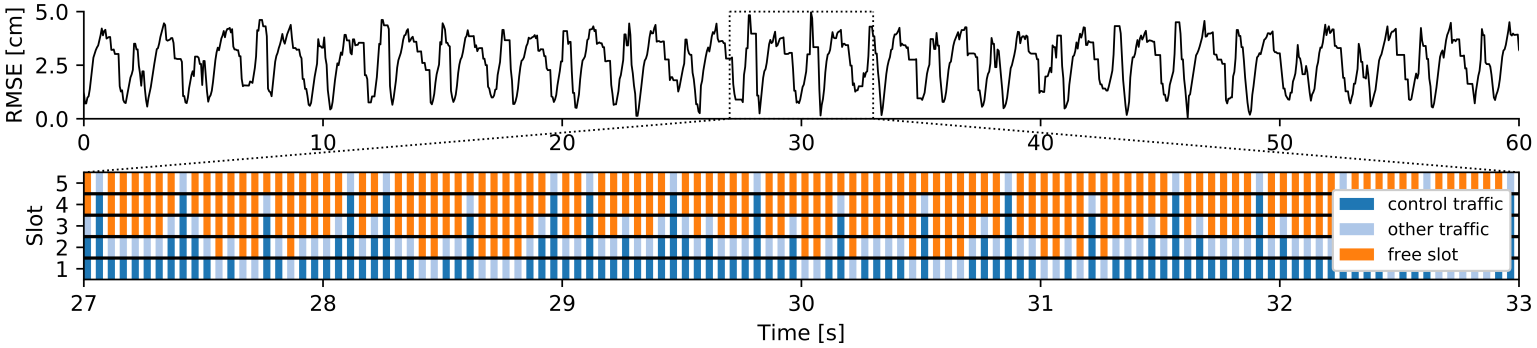
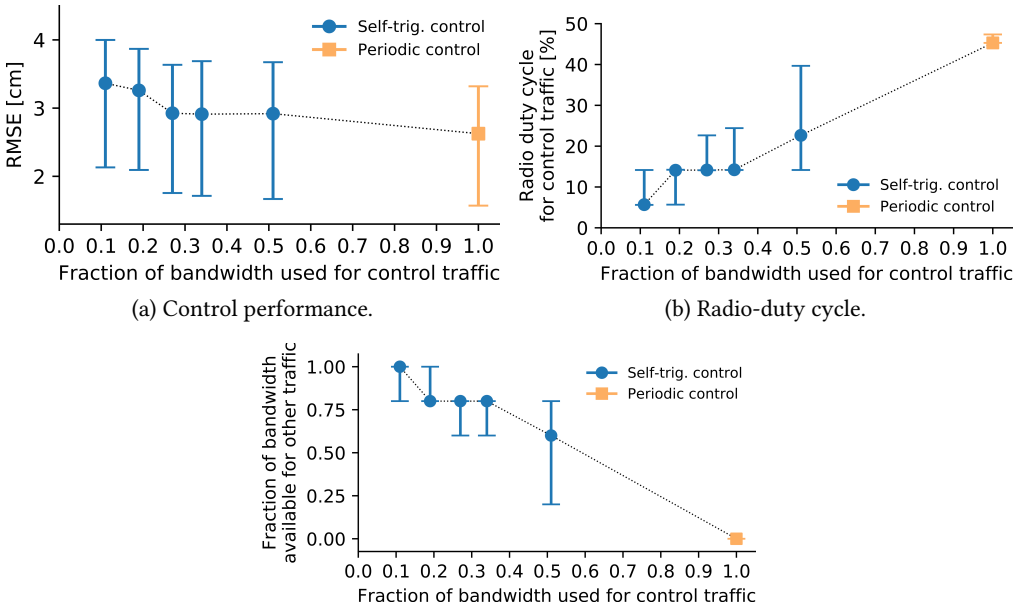


FIGURE 5.4: Control performance and bandwidth utilization over time, recorded during one of our experiments. The scheduling policy described in Section 5.4 is used but applications can also specify any other policy. Each vertical line in the lower figure represents a communication round. The control traffic demands vary over time between 0 and 4 slots. One slot is always used for other traffic and the remaining free slots are shut down to save communication energy.



(a) Control performance.

(b) Radio-duty cycle.

(c) Bandwidth available for other traffic.

FIGURE 5.5: Trade-off between control performance, communication energy efficiency, and flexibility in serving other traffic for different fractions of control traffic, reported in terms of the median and 25th/75th percentiles. Control performance decreases when less bandwidth is used for control traffic. Conversely, freed resources that are not needed for control traffic result in considerable energy savings or allow to serve other traffic (e.g., status, sensors).

off, which saves significant amounts of energy. Due to the sine distortion signal, the RMSE at the top exhibits a similar shape.

5.6.3 Control Performance vs. Efficiency vs. Flexibility

The triggering threshold δ allows a user to trade control performance for communication energy efficiency and flexibility in serving other traffic. To evaluate this trade-off, we consider six different thresholds and perform for each threshold three 2-minute experiments. In addition, we perform experiments with $\delta = 0$ to obtain results for periodic control, where all agents communicate in every time step requiring all bandwidth for control traffic. For each threshold, we report the median and 25th/75th percentiles across the three experiments.

Figure 5.5 shows RMSE, radio duty cycle for control traffic, and fraction of bandwidth available for other traffic against the fraction of bandwidth used for control traffic. We use this intuitive unit for the x-axis instead of the triggering threshold δ because our measurements reveal that each δ corresponds to a certain fraction of bandwidth used for control traffic with negligible variance across experiments with the same δ.

Looking at Figure 5.5, we observe that the more bandwidth is used for control traffic, the better the control performance and the less bandwidth is available for other traffic. As expected, higher bandwidth demands result in a higher radio duty cycle. Using 25 % of the available bandwidth for control traffic, the control performance is still

comparable to the periodic baseline. Further bandwidth reductions lead to a noticeable decrease in control performance compared with the periodic baseline of up to 22 % when only 11 % of the available bandwidth is used for control traffic. At the same time, up to 87 % of communication energy can be saved, while the vast majority of the bandwidth is available for other traffic. Overall, these experimental results demonstrate that our control-guided communication approach allows for exploiting this trade-off to meet a wide range of requirements of emerging CPS applications.

Postscript

The limited communication bandwidth is an inevitable bottleneck in emerging CPS applications, using more agents and generating larger traffic volumes. Modern ETC and STC designs can reduce the generated traffic without sacrificing control performance. However, transferring those savings from the control side to the communication side to gain a real end-to-end benefit is non-trivial as communication resources are allocated in advance. In particular, there is no solution for wireless multi-hop networks. We have addressed this shortcoming by proposing control-guided communication, a new co-design approach where the control system predicts and informs the communication system about future resource demands. As a result, bandwidth and energy can be saved or reallocated to additional agents or other kinds of traffic. Experiments on a CPS testbed demonstrate the effectiveness of our approach.

6

Scaling Beyond Bandwidth Limitations: Wireless Control With Stability Guarantees Under Overload

Preface

Our control-guided communication approach in the previous chapter integrates a modern control design (STC), thereby improving the interaction between a growing number of agents and enabling more adaptive, scalable, and efficient CPS. However, while ETC and STC designs can effectively reduce the generated control traffic *on average*, they cannot prevent situations in which many or potentially all agents want to communicate simultaneously. This leads to overload situations, where the communication demand exceeds the available bandwidth, resulting in unpredictable message losses and, thus, the inability to give stability guarantees or performance bounds.

In this chapter, we overcome the overload problem and propose a CPS design that can guarantee stability while scaling beyond bandwidth limitations. Based on a novel co-design, combining control via predictive triggering and our wireless communication primitive Mixer (see Chapter 2), we can efficiently distribute communication needs and dynamically allocate the available bandwidth to agents with the greatest need. We validate our system with experiments on a real CPS testbed featuring 20 cart-pole systems that synchronize their movements over a multi-hop network. The results show that our solution achieves significantly better control performance under overload than the state of the art. Moreover, we formally prove that our co-design guarantees closed-loop stability for physical systems with stochastic LTI dynamics.

6.1 Introduction

Distributed control over wireless networks is essential for CPS in which multiple agents work on a common task. Examples include mobile robots jointly manufacturing a product [17, 177] and drones flying in formation in a rescue mission [61]. To support emerging multi-agent CPS, a tight integration and co-design of wireless communication and control is needed that:

- *Facilitates distributed control.* To coordinate their activities, each agent must be capable of exchanging messages with every other agent. In this way, each agent can drive a local control loop based on local sensor readings (e.g., a drone can stabilize its flight), while in addition communication with other agents allows to solve a distributed control task (e.g., drone swarm keeping a desired formation). This is commonly referred to as *multi-agent systems* [107].
- *Tames and accounts for network imperfections.* Control of dynamical systems like drone swarms requires information exchange every few hundred milliseconds across large distances [140]. Thus, multi-hop communication with bounded latency and high reliability is crucial. Moreover, because wireless communication is notoriously unreliable, occasional message losses and communication delays must be accounted for by the control design.
- *Caters for small cost, weight, form factor, and energy consumption.* Depending on the application scenario it can be beneficial, if not necessary, to deploy the entire multi-agent CPS on low-cost, low-power embedded hardware with small weight and form factor, for example, to not exceed the maximum payload of a drone or to support remote energy-harvesting sensors [17].

While meeting these requirements is challenging in itself, multi-agent CPS also face an *overload problem.* To illustrate, let us consider the scenario shown in Figure 6.1. The bandwidth M available per time step (update interval) can be used to transmit up to M_{C} control messages and up to M_{A} messages carrying other application data, such as photos and video streams [17, 61]. However, as applications become ever more sophisticated—requiring, for example, more agents, shorter update intervals, and higher-volume data streams—the generated traffic inevitably exceeds the available bandwidth M. Specifically, a system is *overloaded*, when the control traffic generated by periodic control exceeds M_{C}. Periodic control in an overloaded system causes additional message loss equal to the amount of bandwidth exceedance, which may make it impossible to guarantee closed-loop stability and to achieve the required control performance.

Advances in wireless communication technology cannot solve the overload problem: The network bandwidth remains a limited resource that is ultimately outrun by increasing application demands. On the other hand, as illustrated in Figure 6.1 and detailed in Section 6.2, existing approaches such as ETC and STC can only reduce the generated control traffic *on average* compared to periodic control. However, in

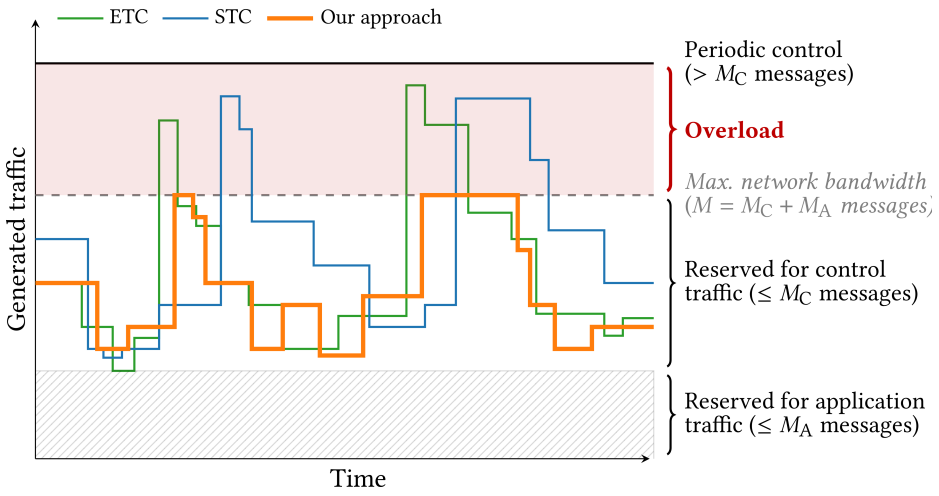


FIGURE 6.1: Illustration of the problem and approaches. The network bandwidth M available per time step (update interval) can be used to transmit up to M_C control messages and up to M_A application messages. A system is overloaded if the control traffic generated by periodic control exceeds M_C and therefore M. Using ETC or STC in an overloaded system cannot prevent temporary bandwidth exceedances, resulting in unpredictable behavior. Our approach ensures that the generated control traffic never exceeds M_C, and closed-loop stability can be provably guaranteed.

an overloaded system they typically cannot prevent situations where the bandwidth is temporarily exceeded. The behavior of the system during such situations (e.g., in terms of closed-loop stability) is unpredictable, which is unacceptable for critical CPS applications requiring a priori guarantees [144].

Contributions. We present the design, analysis, and real-world evaluation of a wireless CPS that addresses the overload problem, while meeting all of the above-mentioned requirements. Using our approach, the generated control traffic never exceeds the fraction M_C of the bandwidth reserved for control (see Figure 6.1), and we derive stability guarantees for the entire multi-agent CPS.

As described in Section 6.3 to Section 6.6, our approach is based on a novel co-design and tight integration of wireless communication and control. The key idea is to determine how urgent each agent needs to transmit control data, and to assign the available control bandwidth M_C in every update interval to those agents that currently have the highest need. Although the communication system we design is highly reliable, occasional message loss cannot be avoided due to the limited time for communication. Our control design accounts for such message loss as well as communication delays. By tightly integrating communication and control, we reduce the jitter caused by imperfect synchronization of distributed hardware components in real CPS to the point where it can be neglected. As a result, our overall solution is amenable to a formal end-to-end analysis of all relevant CPS components (communication, control, and physical system), which allows us to prove closed-loop stability for heterogeneous agents with stochastic LTI dynamics.

We evaluate our approach on a 20-agent CPS testbed. Each agent consists of a low-power wireless embedded device and a cart-pole system, whose dynamics are representative of mechanical systems found in real-world applications [11, 171]. The 20 agents form a 3-hop network, exchanging control traffic every 100 ms to synchronize the movement of their carts. Our experiments demonstrate that, in the scenarios we tested, the overall CPS is stable as predicted by our theoretical analysis despite external disturbance. The experimental results further show that our approach synchronizes the carts better than a highly optimized periodic baseline, while using fewer control messages.

In summary, this work makes the following contributions:

- We present the first practical wireless CPS design that addresses the overload problem. With this, we improve the scalability of CPS toward future applications with increasing demands.
- We formally prove that our wireless CPS design guarantees closed-loop stability for heterogeneous agents (i.e., physical systems) with potentially different stochastic LTI dynamics.
- Real-world experiments on a 20-agent CPS testbed confirm our theoretical results and demonstrate an improved control performance while using fewer control messages.

6.2 Problem and Related Work

This section defines the research problem we tackle in this work and reviews relevant prior work.

6.2.1 Problem Formulation

Scenario. Motivated by emerging applications in search and rescue, manufacturing, or construction [17, 61, 177], we consider wireless CPS consisting of N heterogeneous agents that jointly work on a distributed control task. Each agent runs a controller that computes actuator commands based on local sensor readings and information received from other agents. While local readings allow each agent to, for instance, stabilize itself, communication is essential to solve the distributed task, such as flying in formation. To this end, the agents are equipped with radio frequency (RF) transceivers to exchange messages over a wireless multi-hop network.

The agents' physical dynamics and the required control performance govern the update interval at which control information is to be exchanged in a many-to-many fashion among the agents. In this work, we target distributed control of mechanical systems requiring update intervals on the order of tens to hundreds of milliseconds [4, 140]. Conversely, the bandwidth of the wireless network determines the number M of messages that can be exchanged within each update interval. Out of these, as illustrated

in Figure 6.1, only $M_C < M$ messages can carry control information. This is because the wireless network is also used to transmit other application data (e.g., video streams, photos, status and configuration data), which occupy $M_A = M - M_C$ messages per update interval.

Overload problem. The state of the art with respect to the outlined application scenario supports at most $N = 5$ agents at an update interval of 50 ms when no application traffic is transmitted (see Chapter 4 and Chapter 5). This is insufficient for many envisioned CPS applications requiring tens to hundreds of agents, ever shorter update intervals to realize more sophisticated control tasks, and the continuous collection of high-volume data streams, for example, to feed machine-learning models [17, 61, 177].

The bottleneck is the limited network bandwidth M. While advances in wireless technology can increase M, the required infrastructure costs may not be economically viable [52]. Moreover, the traffic volumes of machine-to-machine communication to enable monitoring and control are expected to see annual growth rates of up to 50 % over the next ten years [38], quickly outrunning any increase in M. We refer to a system as overloaded when the control traffic generated by periodic control exceeds the available control bandwidth M_C. As a result, it becomes impossible to guarantee stability and achieve the desired control performance with periodic control methods.

6.2.2 Related Work

How to achieve high-performance control under limited communication resources has been widely studied. However, as discussed below, most prior approaches cannot solve the overload problem. A few theoretical control concepts can in principle address the problem, but none of these works considers the challenges of integrating control with a real network, neither wired nor wireless.

Event- and self-triggered control. ETC and STC methods aim to efficiently use the limited communication bandwidth [63, 124]. To this end, they only let agents transmit control information when needed (e.g., some error exceeds a threshold) instead of letting all agents transmit control information in every update interval as in standard periodic control. However, while ETC and STC can reduce the control traffic *on average* compared to periodic control, they cannot solve the overload problem: At any point in time, it can happen that more agents signal communication needs than the network can support, as illustrated in Figure 6.1. How to resolve such situations and provide stability guarantees under overload is an unsolved problem.

Further, using ETC, agents make communication decisions instantaneously, which leaves the communication system no time to reallocate unused bandwidth to other agents, wasting precious resources. Using STC, an agent decides about the next time it needs to communicate at the current communication instant, so unused bandwidth can be reallocated. However, there is no way to react between two communication instants; that is, the agent cannot react to unforeseen disturbances, which negatively affects control performance and stability.

Predictive triggering. Predictive triggering can handle disturbances by letting agents decide at every time step if they need to communicate some time in the future [169, 170]. Moreover, Mastrangelo et al. extend predictive triggering toward non-binary communication decisions, where a priority measure based on the probability of exceeding a threshold is used to schedule communication [120]. Unlike the binary communication decisions in ETC and STC, this approach can in principle address the overload problem, which is why we adopt it. However, compared with all prior work on predictive triggering, we a) propose an improved priority measure that is efficiently computable on resource-constrained hardware, b) address the challenges of integrating predictive triggering with a real wireless communication system, c) conduct a formal stability analysis, and d) validate our overall co-design on a real-world CPS testbed.

Contention resolution. Although it has been shown [120] that predictive triggering yields better performance than contention resolution [117, 128], we discuss it here as an alternative theoretical concept that can in principle address the overload problem. Besides the inability of some contention resolution algorithms to support heterogeneous agents [117], which is a common requirement in practice, none of the existing algorithms (see, e.g., [13, 37, 117, 128, 145]) has been integrated with a real network. Instead, the algorithms are exclusively evaluated in simulation, making assumptions about a potential communication system that are not backed up through real-world experiments.

Online scheduling. Recently, a few distributed [126, 192] and autonomous [125] scheduling approaches for wireless control systems have been proposed. The goal of these approaches is to adjust sampling periods and communication schedules in response to unexpected external disturbances and varying wireless link qualities. Although their distributed operation resembles our scheduling approach and also shares the goal of adapting to external disturbances, the scheduling criterion and techniques are fundamentally different. For instance, rather than adjusting the schedules to link quality changes, such changes are effectively accounted for by our ST and network coding based communication system, thereby hiding them from the scheduler.

Practical control-communication co-designs. Table 6.1 qualitatively compares our and prior practical co-designs that predict communication demands and validate the integration of control with wireless communication against the dynamics of real physical systems and real wireless networks.

Araujo et al. use STC to control a quadruple tank process over a single-hop network with update intervals of a few seconds [10]. Saifullah et al. present a multi-hop solution for power management in data centers, using update intervals of 20 s or longer [150]. While their control design is not explicitly based on STC, it exhibits similar properties, including the inability to cope with overload. The same holds for our co-design presented in Chapter 5 and the work by Santos et al. [152]. Both employ an STC approach and demonstrate control of fast physical systems; however, only our

TABLE 6.1: Comparison to prior practical co-designs of control and wireless communication that predict communication demands and have been validated using experiments on real physical systems and wireless networks. With fast physical systems we refer to the ability of a co-design to support update intervals of a few hundreds of milliseconds, which is typically required to control mechanical systems, such as a quadcopter swarm [140]. Co-designs that can quickly react to disturbances are here defined as those that can spontaneously react based on current measurements in contrast to, for example, STC designs, which decide about their next communication instant at the current one and cannot react in-between.

Work	Fast physical systems	Multi-hop networks	Quickly react to disturbances	Stability guarantees	Addresses overload
[10]	✗	✗	✗	✓	✗
[150]	✗	✓	✓	✗	✗
[152]	✓	✗	✗	✗	✗
Chapter 5	✓	✓	✗	✗	✗
This	✓	✓	✓	✓	✓

solution supports control over multi-hop networks. None of the works based on STC can spontaneously react to disturbances, and only [10] provides stability guarantees.

We also note a few other recent control-communication co-designs that use STC [110, 112] or ETC [20, 172] to reduce communication and that have been evaluated on real wireless networks. In contrast to our work, these co-designs target slow physical systems (e.g., water distribution networks [20, 172]) requiring update intervals on the order of seconds and have only been evaluated on simulated physical systems. Most importantly, none of them addresses the overload problem.

In summary, the co-design proposed in this chapter is the first to address the overload problem, while providing several other properties (see Table 6.1) essential for emerging CPS applications.

6.3 Overview of Co-Design Approach

To overcome the overload problem, we propose a novel co-design approach that integrates control and communication at *design-time* (i.e., prior to operation) and at *run-time* (i.e., during operation). The design of the communication system tames network imperfections to the extent possible, and the design of the control system takes the remaining imperfections in terms of message loss and delay into account. During operation, the control system continuously reasons about the future need for communication of each of the N agents. The communication system time-synchronizes the agents and adapts to communication needs by dynamically assigning the M_C messages available for control traffic in each update interval to agents with the highest need. As illustrated in Figure 6.1, our co-design approach ensures that the generated control traffic never exceeds M_C, and closed-loop stability can be provably guaranteed.

Figure 6.2 shows the overall system architecture and the data flow between the building blocks inside each agent. Our novel control system consists of three building blocks:

B1 The *controller* performs the desired control task for the physical system connected to agent i. It locally stabilizes the individual system, while at the same time it exploits many-to-all communication capabilities to realize the distributed control task.

B2 The *priority measure* indicates an agent's future need for communication to meet the required control performance. The measure makes it possible to compare communication needs independent of the specific control tasks and for different physical dynamics of heterogeneous agents.

B3 The *estimator* provides the controller and the priority measure with estimates of the current state of all agents based on the available information received over the network.

We jointly design the control system with a novel wireless communication system, which consists of the following three building blocks:

B4 The *state exchange* distributes, in each update interval k, M_C control messages over the wireless multi-hop network to all N agents. The set of at most M_C agents that are allowed to transmit their messages may change from one update interval to the next. Because of unavoidable packet losses when communicating over wireless channels, it is possible that an agent receives only a subset of the transmitted messages; however, our experiments and empirical evidence from Chapter 2 show that the probability is extremely low (i.e., less than 0.01 %).

B5 The *priority exchange* builds on the mechanism from **B4** to distribute each agent's current priority measure among all N agents in the system.

B6 The distributed *scheduler* dynamically assigns the M_C control messages available in each update interval k to the agents with the highest priorities. Every agent computes the global communication schedule locally. If an agent does not have full information about the latest priorities, it is guaranteed that it does not disturb the communication of the other agents.

Road map. In Section 6.4 and Section 6.5, we detail the co-design of the control and wireless communication systems along the different building blocks **B1** to **B6**. Then, we describe the careful integration of control and communication, and formally prove stability of the entire CPS for physical systems with stochastic LTI dynamics in Section 6.6. Finally, Section 6.7 complements our analytical results through real-world experiments on a CPS testbed.

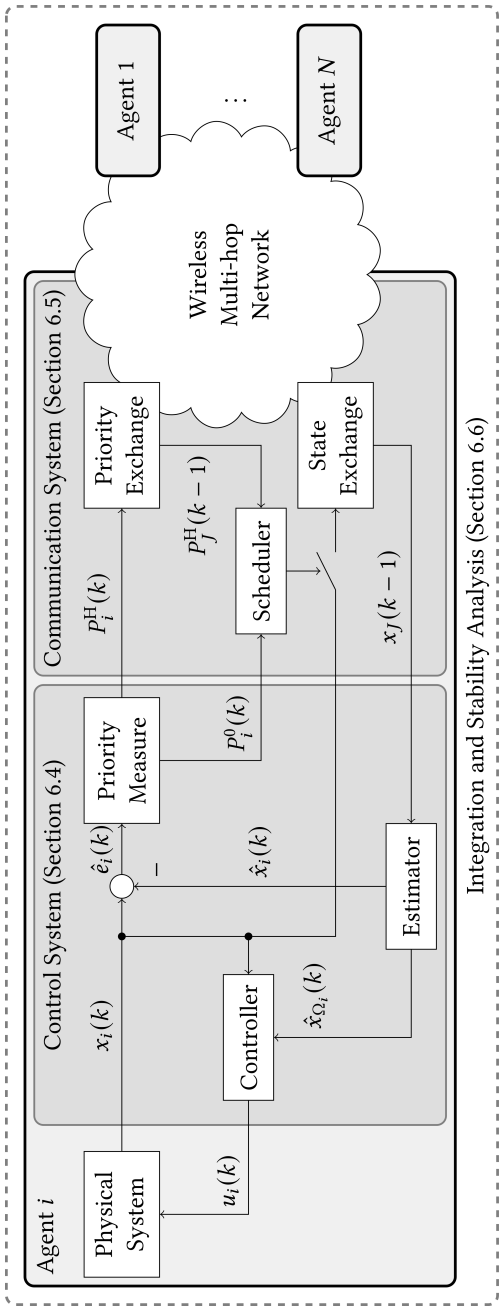


FIGURE 6.2: System architecture of our proposed co-design approach. The CPS consists of N heterogeneous agents working on a common task. Each agent controls a physical system. To coordinate their actions, all agents communicate through a wireless multi-hop network. Ω_i denotes the set of agents relevant for the control task, and Jdenotes all agents whose information has been successfully exchanged in the previous time step.

6.4 Predictive Triggering and Control System

In this section, we present the design of the control system. We first provide a model of the physical system and then show how the different building blocks of Figure 6.2 are designed. In particular, we introduce the controller (**B1**) and the estimator (**B3**) in Section 6.4.2, and derive the priority measure (**B2**) in Section 6.4.3.

6.4.1 Control System Model

We consider a collection of N stochastic LTI systems. The dynamics of the physical system i (cf. Figure 6.2) of this collection are given by

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k) + v_i(k), \tag{6.1}$$

with discrete time index $k \in \mathbb{N}$, state $x_i(k) \in \mathbb{R}^n$, input $u_i(k) \in \mathbb{R}^m$, A_i and B_i matrices of appropriate dimensions, and process noise $v_i(k) \in \mathbb{R}^n$, which we assume to follow a normal distribution with zero mean and variance Σ_{v_i}. Further, we assume the noise processes of individual agents to be uncorrelated, that is, $\mathbb{E}[v_i(k)v_j(k)] = 0$ for all $i \neq j$, and k. The time needed by the communication system to distribute up to M messages determines the update interval, which represents the fixed length of a discrete time step k.

We consider a distributed control problem, that is, the input $u_i(k)$ of agent i at time k depends not only on its own state but also on the state of (possibly all) other agents. Assuming static linear feedback, the control law for agent i becomes

$$u_i(k) = F_{ii} x_i(k) + \sum_{j \in \Omega_i} F_{ij} \hat{x}_{ij}(k), \tag{6.2}$$

with feedback matrices $F_{ii}, F_{ij} \in \mathbb{R}^{m \times n}$, Ω_i the set of all agents whose state is relevant to agent i (i.e., those for which F_{ij} is non-zero), and $\hat{x}_{ij}(k)$ agent i's estimate of agent j's state. The estimate of other agents' states is based on the information transmitted over the wireless network (cf. Figure 6.2). However, we account for the case that only $M_C < N$ agents can transmit information in every update interval k, that is, an agent i may not receive state updates from all other agents $j \in \Omega_i$ at time step k.

Due to the clearer presentation, we assume throughout the chapter that all agents operate with the same update interval k. Nevertheless, our design also supports heterogeneous agents with different update intervals as long as the communication system can provide the required update interval. Intuitively, agents that are stable at longer update intervals, that is, agents with slower dynamics, can also be stabilized at shorter update intervals. We evaluate different types of systems in Section 6.7, where the heterogeneity is reflected in different communication needs.

6.4.2 Control Architecture

Based on the model (6.1), we now introduce the estimator (**B3**) that lets agent i estimate agent j's current state. Afterward, we discuss the feedback matrices F_{ii} and F_{ij} of the controller (**B1**).

State estimation. As stated in the previous section, agent i needs information of the other agents' states to compute its control input. However, only $M_C < N$ agents can transmit their state information over the communication network. Thus, for all agents from which agent i did not receive an update, it needs to estimate the current state based on the data it received so far. Further, information sent over the network is delayed by one time step and subject to message loss. An estimate of agent j's current state, compensating sporadic message exchange, transmission delays, and message loss, can be obtained via

$$\hat{x}_{ij}(k) = \begin{cases} (A_j + B_j F_{jj}) x_j(k-1) + B_j \sum_{\ell \in \Omega_j} F_{j\ell} \hat{x}_{i\ell}(k-1) & \text{if } \kappa_j(k) = 1 \wedge \phi_j(k) = 1 \\ (A_j + B_j F_{jj}) \hat{x}_{ij}(k-1) + B_j \sum_{\ell \in \Omega_j} F_{j\ell} \hat{x}_{i\ell}(k-1) & \text{otherwise,} \end{cases} \tag{6.3}$$

where $\kappa_j(k)$ denotes, whether ($\kappa_j(k) = 1$) or not ($\kappa_j(k) = 0$) agent j transmitted its state in the current round. Possible message loss is captured by $\phi_j(k)$ ($\phi_j(k) = 0$ if the message was lost and $\phi_j(k) = 1$ otherwise). The intuition behind (6.3) is as follows: in case of successful communication, agent i receives agent j's state. To compensate for the transmission delay, agent i makes a one step ahead prediction. In case of no communication (either intentionally or due to message loss), it propagates its last estimate of agent j's state ($\hat{x}_{ij}$).

Control objective. While we consider distributed control, the individual agents may be unstable, that is, the A_i in (6.1) may have eigenvalues with absolute value greater than one. Thus, each agent needs to locally stabilize itself *and* solve the distributed control task. To make the controller design precise, we consider synchronization as an example of distributed control. We here understand synchronization as trying to have the states of all agents evolve as close as possible. That is, we want to keep the error $e_{ij}(k) \coloneqq x_i(k) - x_j(k)$ between the states of any two agents i and j small. Alternatively, it is also possible to only synchronize a subset of the states. Synchronizing the states (or parts thereof) of multiple agents is a frequently considered problem setting in distributed control and also known under the terms consensus or coordination [106].

Controller design. For ease of presentation, we assume a two-agent setting in the following. However, as we also show in the experimental evaluation, the design straightforwardly extends to more agents. Based on the control objective, we start by

formulating a cost function

$$J = \lim_{K\to\infty} \frac{1}{K} \mathbb{E}\Bigg[\sum_{k=0}^{K-1}\sum_{i=1}^{2}\big(x_i(k)^\mathsf{T} Q_i x_i(k) + u_i(k)^\mathsf{T} R_i u_i(k)\big) \\ +(x_1(k)-x_2(k))^\mathsf{T} Q_{\text{sync}}(x_1(k)-x_2(k))\Bigg], \tag{6.4}$$

with positive semidefinite matrices Q_i and Q_{sync}, which penalize deviations from the equilibrium state and from the synchronization objective, and positive definite matrix R_i, which penalizes high control inputs. Since both agents have an estimate of the other agent's current state, obtained from (6.3), we can rewrite the term in the summations in (6.4), using the augmented state $\tilde{x}(k) = [x_1(k), x_2(k)]^\mathsf{T}$ and augmented input $\tilde{u}(k) = [u_1(k), u_2(k)]^\mathsf{T}$, as

$$\tilde{x}^\mathsf{T}(k)\begin{pmatrix} Q_1 + Q_{\text{sync}} & -Q_{\text{sync}} \\ -Q_{\text{sync}} & Q_2 + Q_{\text{sync}} \end{pmatrix}\tilde{x}(k) + \tilde{u}^\mathsf{T}(k)\begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix}\tilde{u}(k). \tag{6.5}$$

We now seek to find the optimal feedback controller that minimizes the cost function J. Since the cost function J puts an emphasis on the stability of each individual agent (through Q_i) and on the synchronization objective (through Q_{sync}), solving the optimization problem results in finding a trade-off between both objectives. This trade-off can be influenced through the choice of Q_i and Q_{sync}. Using the formulation in (6.5), this problem can be solved using standard tools from linear optimal control [7] and yields a static linear feedback controller as in (6.2). Note that this implies that the agents do not need to solve the optimization problem at run-time. Instead, the optimal controller can be computed offline before the system operation commences.

6.4.3 Priority Measure

We now discuss how to derive a priority measure (**B2**) that enables the communication system to allocate the M_C available control messages to the agents with the highest needs. If state estimates (6.3) were perfect, there would be no need for any communication. However, since we consider noisy dynamics (cf. (6.1)), the estimates will, over time, start to deviate from the true state. Thus, we seek to transmit information as soon as the error between estimated and true state becomes too large. Therefore, agent i computes the same[1] estimate other agents have of its state and derives the estimation error $\hat{e}_i(k) \coloneqq x_i(k) - \hat{x}_{ii}(k)$. Ideally, we would now like agent i to transmit its state whenever $\hat{e}_i(k)$ exceeds some threshold. However, we need to a) announce communication needs in advance, that is, we cannot use the current $\hat{e}_i(k)$ to decide about communication, and b) we need a measure that quantifies agent i's communication demand in a generic way, instead of a binary decision as it is the case with ETC and STC.

[1] Estimates may diverge in case of message loss. We discard impact of message loss for the triggering design and analyze its impact in the stability analysis in Section 6.6.2.

We first introduce the measure that we use to quantify whether the estimation error is "too large." This measure is the squared Mahalanobis distance [115] of the estimation error from the origin,

$$\|\hat{e}_i(k)\|_{\mathrm{M}}^2 \coloneqq \mathbb{E}[(\hat{e}_i(k))]^{\mathsf{T}} \operatorname{Var}[\hat{e}_i(k)]^{-1} \mathbb{E}[(\hat{e}_i(k))]. \tag{6.6}$$

The distance between an observation and a random variable is in one dimensional settings often measured by assessing how many standard deviations it is away from the mean. The Mahalanobis distance extends this idea to multiple dimensions. For $\operatorname{Var}[\hat{e}_i(k)] = I_n$, with I_n the $n \times n$ identity matrix, the Mahalanobis distance reduces to the Euclidean distance.

Directly using the squared Mahalanobis distance to schedule agents would yield a viable triggering strategy for homogeneous agents. Yet, for heterogeneous agents, the estimation errors may be in different orders of magnitude and, thus, not comparable. Further, instead of an instantaneous decision, we need to announce communication needs in advance. As shown in Figure 6.2, at time step k, each agent receives states that were sent at $k-1$, that is, there is a delay of one time step. Thus, we consider the probability of the squared Mahalanobis distance exceeding a predefined threshold δ_i in $H+1$ time steps. The parameter H denotes how many time steps the communication system needs to reschedule resources. To compute the Mahalanobis distance, we need the expected value and the variance of the error $\hat{e}_i(k+H+1)$. Since the error is the difference between current state and estimated state, its expected value and variance can be calculated using expected value and variance of the state at time $k+H+1$. Assuming no communication between k and $k+H+1$ and exploiting that the noise sequences are uncorrelated, those are given by

$$\mathbb{E}[x_i(k+\mathrm{H}+1)] = A_i^{\mathrm{H}+1} x_i(k) + \sum_{s=0}^{\mathrm{H}} A_i^s B_i u_i(k-1-s) \tag{6.7a}$$

$$\operatorname{Var}[x_i(k+\mathrm{H}+1)] = \sum_{s=0}^{\mathrm{H}} A_i^s \operatorname{Var}[v_i(k-1-s)](A_i^s)^{\mathsf{T}}, \tag{6.7b}$$

where $\operatorname{Var}[v_i(k)] = \Sigma_{\mathrm{v_i}}$ for all k. Note that (6.7b) is constant, that is, it does not depend on current data. Thus, the inverse needed in (6.6) can be computed *a priori*, leaving only matrix multiplications that need to be done at run-time.

We define our priority measure as

$$P_i^{\mathrm{H}}(k) = \frac{\gamma\left(\frac{n}{2}, \frac{\delta_i - \|\hat{e}_i(k+\mathrm{H}+1)\|_{\mathrm{M}}^2}{2}\right)}{\Gamma\left(\frac{n}{2}\right)}, \tag{6.8}$$

with $\gamma(\cdot)$ the lower incomplete gamma function, $\Gamma(\cdot)$ the gamma function, and n the dimensionality of the physical system. This priority measure is the closed-form expression of the probability that the squared Mahalanobis distance of a normally distributed random variable with zero mean exceeds $\delta_i - \|\hat{e}_i(k+\mathrm{H}+1)\|_{\mathrm{M}}^2$ [53]. Technically, it is

therefore the probability that the Mahalanobis distance of the estimation error will either exceed δ_i or shrink by more than $\delta_i - \|\hat{e}_i(k+\mathrm{H}+1)\|_{\mathrm{M}}^2$ and not exactly the sought-after probability. However, this does not change the qualitative result (systems with higher probability of exceeding δ_i will have a higher priority measure) and, in that way, allows to rank systems by priority. Further, we do not need the exact probability of a system exceeding the threshold for any of our technical results. Note that (6.8) can only be evaluated if the estimation error is smaller than the threshold. If the threshold is already exceeded, we can reverse the arguments of $\gamma(\cdot)$ to compute the probability that the estimation error falls again below the threshold.

6.5 Adaptive Communication System

The second part of our CPS co-design is the communication system. As outlined in Section 6.3, we consider the problem that the communication bandwidth is not sufficient to meet the demands of application and periodic control. We have M_{C} control messages for N agents with $N > M_{\mathrm{C}}$, and a potentially high number of application messages M_{A}. To quickly and reliably exchange all $M = M_{\mathrm{A}} + M_{\mathrm{C}}$ messages (**B4**), we design a wireless protocol based on our communication primitive Mixer (Chapter 2). Section 6.5.1 details how we increase the scalability of Mixer to support larger multi-agent systems, whereas Section 6.5.2 describes novel extensions to support distributed control based on our predictive triggering approach. Specifically, we conceive a scheme for the efficient and reliable exchange of priorities among agents (**B5**), which we then leverage for our distributed scheduling design (**B6**) to adaptively allocate messages to agents with the highest communication needs.

6.5.1 Communication Support for Scalable Multi-Agent Systems

In prior approaches capable of distributed control over wireless multi-hop networks (see Chapter 4, Chapter 5, and [16]) the latency to exchange M messages scales as $O(MT)$, with T the time required for the dissemination of a single message. With this scaling behavior the number of messages that can be exchanged in a given update interval is rather small. For instance, the experiments in [16] support a maximum of five agents with a 50 ms update interval an no application traffic. To significantly improve on this number, we adopt and enhance our many-to-all communication primitive Mixer (Chapter 2), which will form the basis of our wireless protocol. Mixer approaches the order-optimal scaling of $O(M+T)$, resulting in a significant improvement over the $O(MT)$ scaling for larger M and T.

Below we give a brief overview of how Mixer works and describe how we improve its performance by a) porting it from IEEE 802.15.4 to the faster BLE PHY to reduce T, and b) making use of the most recent network state information to communicate more effectively. Section 6.5.2 details how we enhance Mixer's functionality for a systematic co-design with the predictive triggering based control system.

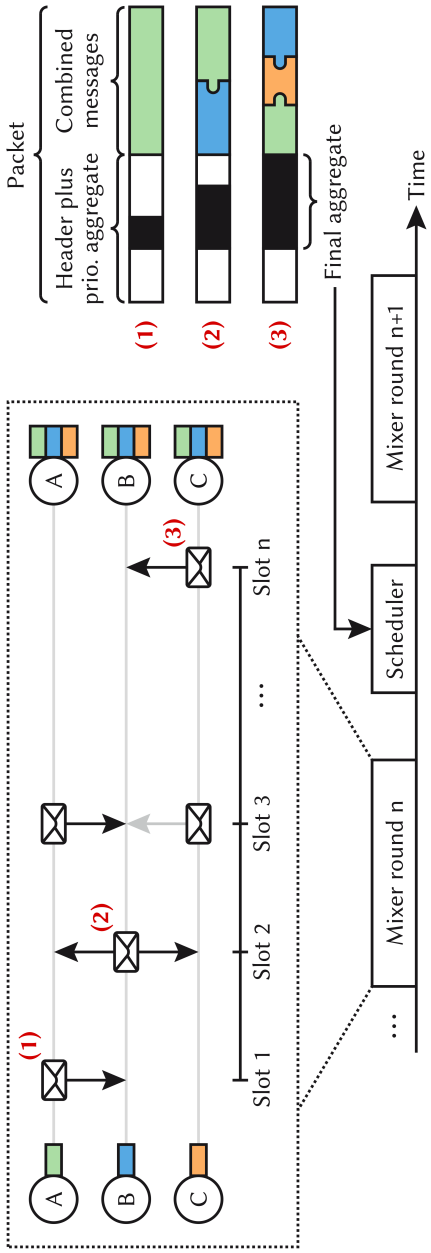


FIGURE 6.3: Overview of the wireless communication protocol. MIXER realizes the many-to-all message exchange in a network (colors represent messages). A MIXER round consists of synchronized time slots, where in each slot nodes (A, B, C) decide whether and what to transmit. Nodes build packets by randomly combining already received information, for example, node B sends a combination of blue and green in slot 2. Priorities are exchanged and aggregated as part of MIXER's header. At the end of the round all nodes have all messages. The scheduler runs locally on each node between rounds and uses the final aggregate to determine the global message allocation for the next round.

Mixer protocol operation. Mixer is a fast, efficient, and reliable communication primitive that disseminates a set of M messages among all nodes in a network by using ST [196] and RLNC [70]. As shown in Figure 6.3, a Mixer *round* consists of synchronized time *slots*, and in each slot the nodes decide independently whether to transmit or listen for a *packet*. Mixer packets consist of a protocol header and a combination of 1 to M messages as *payload*. Such a combination is created by choosing a random subset of already received packets and combining their payloads via XOR. This coding-based approach is the main reason for the improved scaling behavior, as it makes better use of the communication channel. The slotted communication continues until all M messages ($M = 3$ for the example in Figure 6.3) can be decoded by all nodes, or the predefined maximum duration of a round has elapsed.

ST exploit the capture effect [48] describing that, under certain conditions, multiple transmissions can overlap at a receiver, but one of the packets is likely to be received successfully. This is comparable to a conversation between two people while at the same time other people are talking from a greater distance. The voice signals interfere in the air, however, it is likely possible to understand the person nearby. For example, in slot 3 in Figure 6.3, node B successfully receives the packet from node A, despite the interfering transmission of node C, due to the capture effect.

Mixer using Bluetooth Low Energy. The original Mixer implementation targets IEEE 802.15.4-compliant low-power RF transceivers with a limited data rate of 250 kbps. Another PHY in the low-power wireless domain is BLE, which supports data rates of up to 2 Mbps in the Bluetooth 5 standard, which is 8× faster than IEEE 802.15.4. To benefit from the faster PHY, we implemented Mixer using BLE on the Nordic Semiconductor nRF52840 platform.

This platform also supports the IEEE 802.15.4 PHY, which makes it easier to compare the end-to-end performance of both variants. For this purpose, we run experiments involving several thousand Mixer rounds on a 31-node testbed at Graz University of Technology [157]. The nodes were deployed across several rooms and hallways in an office building. With a transmit power of 8 dBm, the nodes form networks with a diameter of 2 hops (IEEE 802.15.4) and 3 hops (BLE). Nodes exchange messages with a size of 18 B in an all-to-all fashion, that is, in every round all nodes start with one message ($M = 31$) that should be disseminated to all other nodes.

The left plot in Figure 6.4 shows the latency distribution of both Mixer variants. We see that our BLE implementation of Mixer outperforms the original version in terms of communication latency by 2.5–4.7× with a median of 4.2×. The main reason why the theoretical speed-up of 8× is unattainable in practice are differences in the robustness of the two PHYs: while the IEEE 802.15.4 PHY features a spreading code, the 2 Mbps BLE mode is uncoded and thus more susceptible to interference. Consequently, the number of slots needed with BLE to exchange M messages increases compared to IEEE 802.15.4, as visible in the right plot of Figure 6.4. At the same time, the higher data rate reduces the length of each slot, which leads to the overall latency improvements.

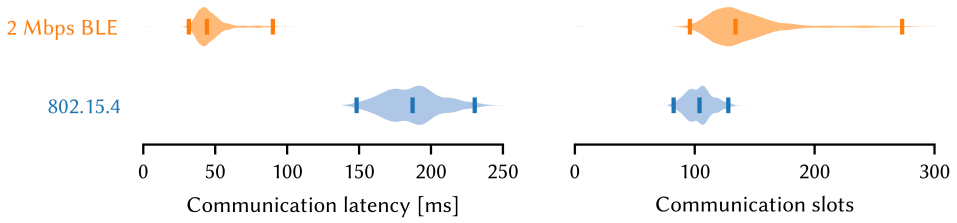


Figure 6.4: Distribution of communication latency using Mixer with the IEEE 802.15.4 and 2 Mbps BLE PHY. Although communication using BLE requires more slots, because this PHY has lower robustness, the higher data rate reduces the length of each slot and thus the resulting latency by 4.2× (median). The strokes mark 1st, 50th, and 99th percentiles.

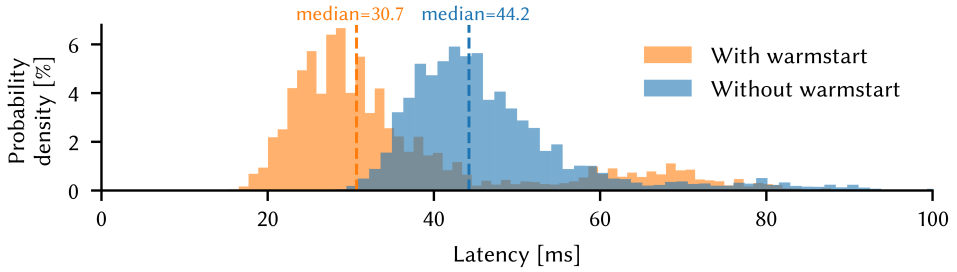


Figure 6.5: Histogram of Mixer's latency using the 2 Mbps BLE PHY with and without warmstart. Warmstarting nodes helps them communicate more effectively at the beginning of a round and reduces latency by 31 % (median).

Warmstart. Wireless control with fast physical systems requires short update intervals. For instance, drones need to exchange their positions every 100 ms to fly in formation or to avoid crashes [30]. In general and from a network perspective, the topology changes continuously in such dynamic scenarios. However, viewed from one Mixer round to the next, the topology changes are relatively small due to the short update interval. We leverage this observation to further improve Mixer's performance without sacrificing communication reliability.

Mixer rounds are independent of each other. Each node has no information about its neighborhood at the beginning of a round. This changes during a round with each packet received, allowing better and better transmit decisions to be made, and thus the message exchange to be finished more quickly. Instead of acquiring this information from scratch in each round, we extend Mixer to allow neighborhood information to carry over from one round to the next. Such *warmstarted* nodes can communicate more effectively right from the beginning of a round. Experiments with the same testbed setup as before show that warmstarting nodes reduces latency by 12–40 %, with a median of 31 % (see Figure 6.5).

6.5.2 Support for Predictive Triggering

Having established an efficient and reliable communication support for wireless control, we now turn to the core co-design questions from the perspective of the communication system:

1) How to efficiently exchange the priorities computed by each agent within a Mixer round?
2) How to schedule communication based on the priorities such that the network bandwidth is used in the most effective way by allocating messages to the agents with the highest need?
3) How to communicate efficiently when facing varying communication demands?

We propose novel Mixer protocol functionality to answer these three questions, as discussed next.

Efficient exchange of priorities. With respect to their storage format the priorities $P_i^{\mathrm{H}}(k)$ are generally much smaller in size than the full state information $x_i(k)$. The key idea is to exploit this size advantage and use a part of the available network bandwidth for the exchange of all priorities in each Mixer round, to enable using the remaining network bandwidth in the most effective way.

The problem with this is that Mixer's coding operations require that all messages exchanged in a round must have the same size, thus, transmitting priorities and state information as separate messages would completely eliminate the size advantage of priorities. Instead, we extend Mixer's packet header and reserve space for a priority aggregate, as shown in Figure 6.3. At the beginning of a round, each agent knows only its own priority. During a round, agents aggregate priorities from all received packets and embed the current aggregate into the packets they transmit. Since the aggregate is part of every packet, it distributes independently of the M individual messages and propagates quickly and reliably through the network.

To minimize the overhead introduced by the priority aggregate, a compact representation is needed. Importantly, the aggregate must contain information about the M_{C} agents with the highest priority and whether all agents contributed to the aggregate, as required by the distributed scheduling algorithm described below. We propose two representations that satisfy these requirements. The first option is to concatenate all N priorities in the aggregate. With W_P being the number of bits used for each priority value, the required number of bytes S for the aggregate of N priority values is given by

$$S = \lceil NW_P/8 \rceil. \tag{6.9a}$$

Alternatively, if $N \gg M_{\mathrm{C}}$, we achieve a more compact representation using

$$S = \lceil (M_{\mathrm{C}} W_P + M_{\mathrm{C}} \lceil \log_2 N \rceil + N)/8 \rceil, \tag{6.9b}$$

which includes the M_C highest priorities ($M_C W_P$), together with the corresponding agent IDs ($M_C \lceil \log_2 N \rceil$), and one bit per node to verify that all agents have contributed (N). Based on the concrete scenario parameters (i.e., W_P, M_C, and N), our system automatically choses the most compact representation.

Distributed scheduling. A schedule in our communication protocol is the mapping of agents to the M_C available control messages, hence, it describes which agents are allowed to send a control message in the next round. Agents derive the schedule based on the final priority aggregate to which all agents have contributed, by selecting the M_C agents with the highest need to send their state information. More formally, we have that agent i can send a message if its priority $P_i^H(k) \in \mathcal{P}_L$, where $\mathcal{P}_L$ denotes the set of the M_C highest priorities, *and* if $P_i^H(k) > P_\delta$ for some user-defined threshold P_δ. How we deal with equality of priorities, a result of using quantization in a real implementation, is described in Section 6.7.2.

As described in Section 6.4.3, the scheduling horizon H depends on the number of time steps the communication system needs to reschedule resources. In our system the priority aggregate always determines the schedule for the next round, so $H = 1$. In the rare event that agent i does not have the final aggregate at the end of a round, it is not guaranteed that it can compute the correct global schedule and therefore the agent must not send a message. Nevertheless, the agent participates in the communication round like all other agents that have no message assigned to them. However, if $P_i^H(k) \in \mathcal{P}_L$, the message assigned to agent i is not used in the next round. Since messages and priorities spread independently of each other, message loss has little to no effect on the final priority aggregate. In our experiments, we did not record a single priority loss and only one message loss, which shows the reliability of our communication system.

Reducing communication costs. A Mixer round ends when all nodes have received all M messages or the maximum duration of a round is reached. Since M is a parameter set at compile-time, nodes expect the same number of messages in every round, which suits periodic traffic. If the communication demand varies from round to round, then M determines the maximum number of messages. Since nodes do not finish the round before receiving M messages, the difference to the actual number of messages has to be filled with empty messages. As a result, the original Mixer protocol (Chapter 2) cannot translate a lower communication demand into energy savings.

To achieve energy savings if less than M messages need to be sent in a round, we extend Mixer so that nodes can mark unused messages instead of sending empty messages. The idea is similar to the way the priority aggregate works. We aggregate the information about unused messages and quickly distribute them in the network independent of individual messages. To realize this, we can reuse a certain part of Mixer's packet header (i.e., the InfoVector) and switch each slot between its original content and the information about unused messages, thus, adding no further overhead to the transmitted packets.

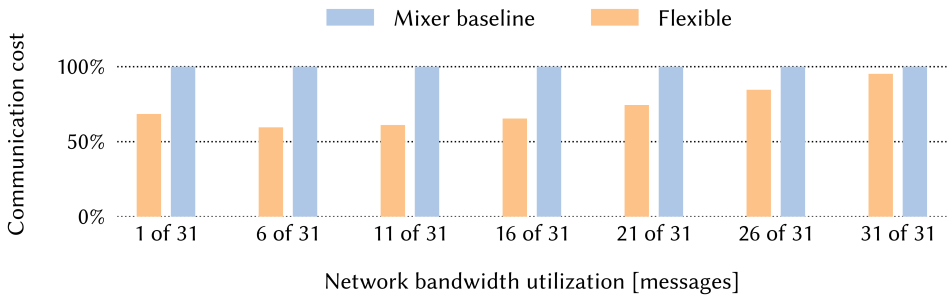


FIGURE 6.6: Communication costs at different network bandwidth utilizations. The original MIXER (baseline) cannot convert lower bandwidth utilization into energy savings. Enabling nodes to mark unused messages (flexible) saves up to 40 % of the communication costs.

We evaluate the savings in terms of communication costs enabled by this modification using the experimental setup from Section 6.5.1. Now, messages also contain the priority aggregate, and we enable the warmstart feature. We determine the communication costs by measuring the average radio-on time across all nodes during a communication round. Figure 6.6 plots for the original MIXER baseline and our approach, labeled *Flexible*, the resulting communication costs when different fractions of the available network bandwidth are used, from 1 used message out of 31 possible messages up to 31 used messages out of 31 possible messages. We can see that, by marking unused messages, our Flexible approach achieves significant savings of up to 40 % compared with the original MIXER baseline. As expected, the savings increase as the network bandwidth utilization decreases.

The possibility to save energy can also be taken into account by the scheduling algorithm. Since the communication need is announced one round in advance ($H = 1$), the estimation error $\hat{e}$ can also decrease until the next time step, whereby allocated messages might no longer be needed. For this, every agent that is allocated a message in the schedule derives $P_i^0(k)$ (as shown in Figure 6.2) by setting $H = 0$ in (6.6) and (6.8). If $P_i^0(k) < P_\delta$, agent i can skip the transmission and mark the message as unused to save energy. The combined scheduling rule can be summarized as

$$\kappa_i(k+\mathrm{H}+1) = 1 \iff P_i^{\mathrm{H}}(k) \in \mathcal{P}_{\mathrm{L}} \wedge P_i^{\mathrm{H}}(k) > P_\delta \wedge P_i^0(k+\mathrm{H}) > P_\delta. \tag{6.10}$$

Overall, this mechanism keeps the energy costs incurred by communication in proportion to the communication demand.

6.6 Integration and Stability Analysis

The integration of our co-designed control and communication systems with respect to their timely interaction and execution is key to meet CPS application requirements. We briefly characterize implementation and timing aspects of our wireless embedded

system in Section 6.6.1, followed by a formal stability analysis of our integrated system in Section 6.6.2.

6.6.1 Implementation Aspects

Hardware platform. We leverage a DPP that integrates an AP and a CP together with the processor interconnect Bolt [166]. The AP executes all application tasks (sensing, actuation, control) while the CP performs all communication tasks. Both processors run in parallel, and the interconnect permits data exchange within formally verified timing bounds [166]. AP and CP use ARM Cortex-M4F CPU cores running at 48 MHz (MSP432P401R) and 64 MHz (nRF52840), respectively. The CP features a low-power RF transceiver that implements the IEEE 802.15.4 and BLE PHYs.

Time synchronization. We use the start of a communication round as a global reference time, which is estimated by all CPs in the network during communication. Each CP propagates this reference time to its respective AP over a dedicated connection between GPIO pins. By aligning all application tasks to the global reference time, similar to Chapter 4, we ensure that the end-to-end delay and jitter among application tasks (i.e., from sensing through control to actuation) is minimized [76]. The slightly different estimates of the reference time by the CPs lead to jitter, which is, however, upper bounded to the nominal length of a communication slot due to the design of Mixer. For our BLE implementation and typical payload sizes, the jitter is on the order of a few hundred microseconds.

6.6.2 Stability Analysis

While the individual agents can locally stabilize themselves, they are coupled through a common control objective. Introducing couplings can destabilize otherwise stable subsystems [105, Ch. 8], even under periodic communication. Moreover, we also consider larger CPS where the limited number of control messages leads to an overloaded system. All these aspects imply that stability of the overall system does not follow straightforwardly from the stability of individual systems and, thus, needs to be analyzed. For the stability analysis, we make the following assumptions.

Jitter. The system model (6.1) implicitly assumes that the time step $k \longrightarrow k+1$ (i.e., the update interval) is constant, implying that jitter is neglected. One time step corresponds to the time between the start of consecutive communication rounds. The CPS scenarios described in Section 6.1 use update intervals of tens to hundreds of milliseconds. As a rule of thumb, jitter that is below 10 % of the nominal update interval does not need to be compensated for [28, p. 48]. As discussed in Section 6.6.1, the jitter of our CPS architecture is way below that threshold, justifying our assumption.

Message loss. For the following analysis, we assume that message loss is i.i.d.. Given the extremely high reliability of Mixer with hardly any message losses during hundreds

of hours of indoor and outdoor experiments (see Chapter 2 [2]), the temporal correlation among those very few message losses is practically negligible. This also justifies our second assumption: we assume that the message loss probability is below 100 %, a theoretical corner case that would prohibit any stability analysis.

System and control design. For notational convenience, we assume homogeneous systems with equal choices of Q and R matrices for controller design, that is, $A_i = A_j$, $B_i = B_j$, and $F_{ij} = F_{ji}$ for all i, j. We, thus, drop the index i for A and B in this section and comment on how the analysis would need to be adapted for the heterogeneous case in the end of the section. The controller matrix is designed such that both the overall system and the individual closed-loop systems $A + BF_{ii}$ would be stable under periodic communication without message loss. For homogeneous systems, the system with the largest Mahalanobis distance from the origin will also have the highest priority.

We define stability of the system in terms of boundedness of its second moment (cf. [146]):

Definition 2 (Mean square boundedness)**.** *The system (6.1) is mean square bounded (MSB) if there exist $\bar{\epsilon}$ and $\rho(\epsilon)$ such that for a given $\epsilon > \bar{\epsilon} > 0$, $\|x(0)\|_2 < \rho$ implies $\sup_{k\geq 0} \mathbb{E}[\|x(k)\|_2^2] \leq \epsilon$.*

With this definition, we can state the following theorem:

Theorem 4. *Consider a system consisting of N homogeneous agents with stochastic LTI dynamics as in (6.1), a transmission channel with $0 < M_C < N$ resources per communication round, the control law (6.2), estimation strategy (6.3), and scheduling law (6.10) with $H = 1$. Then, the system is MSB.*

The proof mainly follows the proofs in [116, 117]. We here provide some intuition and comment on the main differences to [116, 117] while we defer the formal proof to Appendix 6.A.

As the controller is designed so that the overall closed-loop system is stable under reliable, periodic communication, potential instability stems from the estimation error. Thus, we show that we can derive upper bounds for this error in a mean-squared sense, which ensures MSB of the overall system, as we show in Appendix 6.A.

Compared to [116, 117], we assume a deterministic scheduling policy. While in [116, 117] the agent with the largest error might not receive a slot due to the probabilistic scheduling policy, this cannot happen in our design. Therefore, we can obtain tighter upper bounds on the estimation error. Further, since we assume message loss to be i.i.d., we do not need to guarantee that a certain amount of messages is delivered successfully within a given time interval.

Remark 1. *For homogeneous agents, we arrange agents according to their errors, since the system with the largest error has the largest Mahalanobis distance. This simplifies*

[2] In Chapter 2, this has been demonstrated with the IEEE 802.15.4 PHY. We have recorded only one message loss in about 50 000 messages in our experiments with the BLE 2 Mbps PHY.

notation since we can directly consider the errors. However, for heterogeneous agents, where the errors may not be comparable, we can derive an upper bound on the probability of exceeding the threshold, which implies an upper bound on the error. Thus, considering heterogeneous agents comes at no additional complexity, except for notation.

Remark 2. *Theorem 4 guarantees that the second moment is bounded but makes no statements about the actual bound. Explicitly computing this bound would enable to assess the performance of the considered setup, which clearly depends on various parameters, such as the number of control messages M_C. This can be used to derive adjustable performance bounds. For a more detailed discussion, we refer the reader to [117]. How to extend the analysis to scenarios without local stabilization is discussed in Remark 3 in Appendix 6.A.*

6.7 Testbed Experiments

This section provides measurements from a 20-agent wireless CPS testbed used to study both the effectiveness of our co-design approach and the interaction of predictive triggering with the communication system. Evaluating such approaches on real-world testbeds is essential to build trust in the proposed solutions [101]. In our experiments, we find:

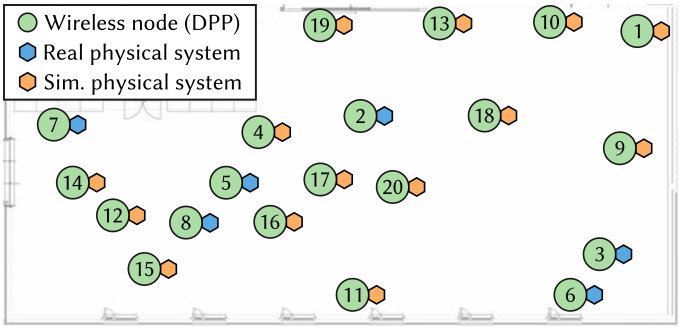
- The priorities determined by the predictive triggering framework reasonably reflect the need for communication depending on the difference between estimated and actual system states. The communication system effectively allocates the available network bandwidth to the agents with the highest need.
- Compared to a baseline using periodic control, our approach reduces control costs as defined in (6.4) by up to ≈19 %, leading to a better synchronization of the 20 agents in our CPS testbed.

6.7.1 Wireless Cyber-Physical Systems Testbed

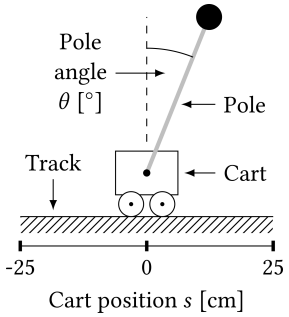
Characteristics. We use a wireless CPS testbed with 20 agents that spans an area of about 20 m by 30 m in our robotic lab. Figure 6.7a shows the locations of the agents in our testbed. Each agent consists of a wireless node (see Section 6.6.1 and Figure 6.7c) and a physical system.

Wireless network. We use the 2 Mbps BLE mode and set the transmission power to −8 dBm. With these settings, paths between wireless nodes are between 1 and 3 hops in length. Communication in our testbed is subject to interference from other wireless transmitters operating in the 2.4 GHz band and radiation of electrical components from co-located robotic experiments.

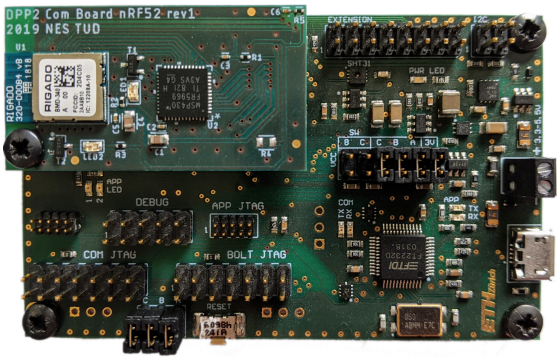
Physical systems. We use *cart-pole systems* as physical systems (see Figure 6.7b and Figure 6.7d) because they exhibit fast dynamics and are widely studied in control theory [23]. In particular, we aim at stabilizing the pole attached to the cart at a pole angle of $\theta = 0°$. This is known as the *inverted pendulum*, which can be well



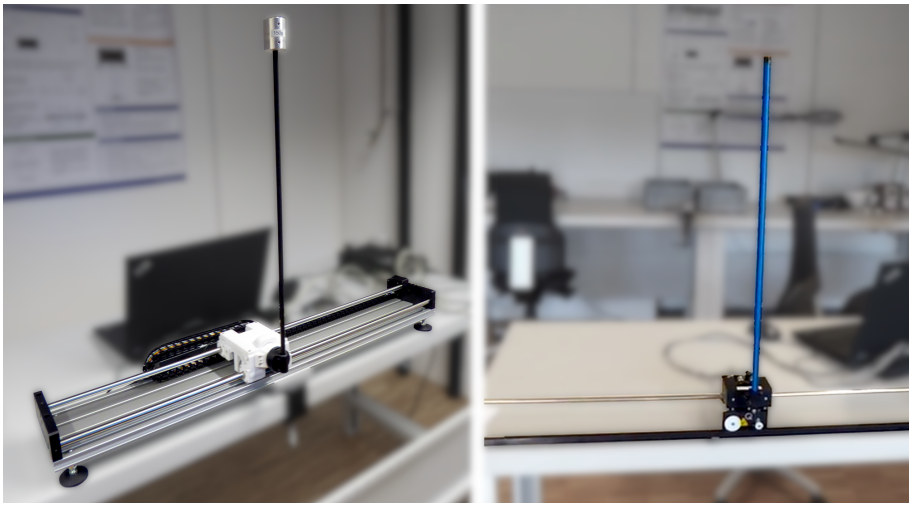
(a) Testbed deployment (20 m by 30 m) including 6 agents with different types of real physical systems and 14 agents with simulated physical systems. Heterogeneity makes distributed control more challenging.



(b) Cart-pole system. Controlled cart movements can stabilize the pole in an upright position.



(c) Wireless node. Dual-processor platform (DPP).



(d) Self-built system on the left side and off-the-shelf system on the right side. The two types of real physical systems offer different dynamics.

FIGURE 6.7: Wireless CPS testbed with 20 agents. Each agent consists of a wireless node (DPP) and a real or simulated cart-pole system as physical system. The wireless nodes form a three-hop network.

approximated by an LTI system for small deviations from the unstable equilibrium (i.e., $\sin(\theta) \approx \theta$). The state $x(k)$ of the system consists of the pole angle $\theta(k)$, the cart position $s(k)$, and their respective derivatives. We measure $\theta(k)$ and $s(k)$ using angle sensors, and estimate $\dot{\theta}(k)$ and $\dot{s}(k)$ using finite differences and low-pass filtering. The control input $u(k)$ is the voltage applied to the motor of the cart, steering the cart's movement speed and direction.

As shown in Figure 6.7a, our testbed features 6 real cart-pole systems connected to the AP of nodes 2, 3, 5–8, and 14 simulated cart-pole systems by running simulation models on the AP of nodes 1, 4, 9–20. The simulated systems allow us to scale up the number of agents in our testbed, given the hefty price tag of real cart-pole systems. In particular, we use 2 off-the-shelf cart-pole systems from Quanser (agent 7 and 8) and 4 cart-pole systems we built in our lab (agent 2, 3, 5, and 6). As a result, we have a testbed with three types of physical systems with different properties and communication demands. In general, heterogeneity makes distributed control more challenging compared to homogeneous systems [106].

6.7.2 Scenario and Settings

Control task. We consider a scenario where the objective and timescales of the distributed control task are similar to, for example, coordination in drone swarms. Drones have to locally stabilize their flight and, additionally, exchange information to coordinate their movements in order to keep a certain formation or avoid crashing into each other. Similarly, all $N = 20$ agents in our testbed have to locally stabilize their poles in an upright position at cart position $s = 0$ cm (i.e., in the middle of their tracks) *and* to synchronize their cart positions. Synchronizing their cart position requires that all agents know the states of all others. Stabilization, however, depends only on information every node has locally. Thus, local update intervals are not bound to the communication delay and can be shorter, making stability less critical. In our setup, we run the local control loop with an update interval of 10 ms. For synchronization, agents exchange information over the multi-hop wireless network with a communication period of 100 ms.

As discussed in Section 6.4.3, systems need to exchange their states because the state predictions using the deterministic model will deviate from the real-world over time. However, in reality, disturbances, such as a gust of wind affecting a subset of drones, can create even more significant deviations. To include this aspect in our experiments, we let all carts start at $s = 0$ cm but purposely fix the cart position of one of the simulated systems (agent 11) at $s = 20$ cm half-way through an experiment.

Comparison approaches. In addition to the control task, high-priority application traffic, such as image data recorded by drones in remote surveillance [61], occupies most of the available wireless communication bandwidth. Specifically, every communication round, $M_{\mathrm{A}} = 18$ messages are reserved for the application. The remaining network bandwidth is used by the agents to exchange the information required for

synchronization. All messages have a payload size of 32 B. We compare our approach against *periodic* control. These two differ in the number of M_C messages and the way these messages are assigned to the agents. Further details regarding the calculation of M_C is provided in Appendix 6.B. To allow for a fair comparison between the two approaches, both use MIXER with the enhancements described in Section 6.5.1. The mechanisms from Section 6.5.2, designed to support predictive triggering, are only used in our *predictive* approach. ETC and STC approaches are not considered here, because it is unclear how they can systematically address overload situations.

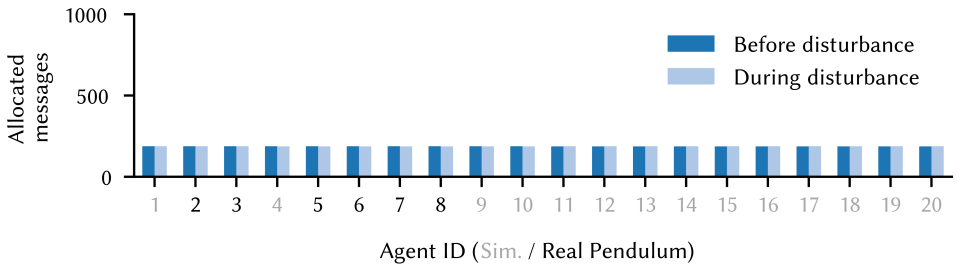
Periodic: Periodic control introduces no overhead. Thus, the remaining network bandwidth can be used for $M_C = 3$ messages per communication round in our scenario. Since there are $N = 20$ agents that want to communicate each round, the system is overloaded. The best we can do with periodic control is to allocate the M_C messages in a predetermined round-robin fashion; that is, each agent is allocated a control message every 7th communication round, effectively sharing its state every 700 ms.

Predictive: This represents the novel predictive triggering approach presented in this work, where in each communication round the M_C agents with the highest priorities are allowed to share their states. To efficiently compute priorities on our embedded hardware we approximate (6.8) with Chernoff bounds [34, Lemma 2.2]. With a continuous priority measure, the probability that two nodes transmit the same priority is 0. In an implementation, however, priorities must be quantized. We choose $W_P = 4$ bit wide priorities, which lets us distinguish 16 different values. If there is a tie, we sort the nodes by their IDs in ascending order. With the priority aggregate in the packet header (see Figure 6.3) the size of all packets increases, which reduces the network bandwidth available for the control messages. In our experiments, this reduces M_C to 2 control messages per round with an aggregate overhead of $S = 5$ B (see (6.9)). Further, we use a threshold of $P_\delta = 0.5$ and choose $\hat{e}_{\max} = (0.03, 0.03, 0.1, 0.3)^\mathsf{T}$ as the maximum error to derive the priorities (cf. Section 6.4.3). In case any individual error state grows above the corresponding entry in $\hat{e}_{\max}$ in terms of its absolute value, we transmit the highest priority.

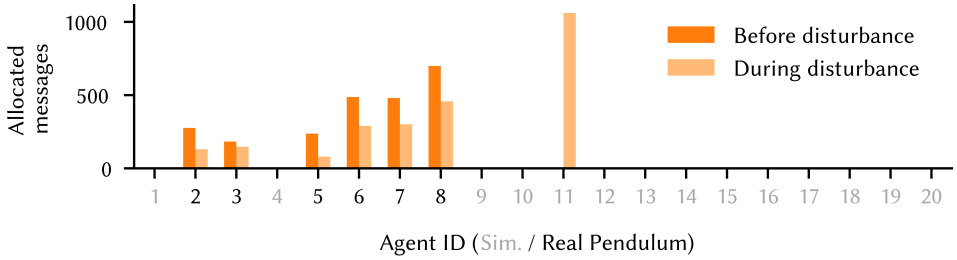
Controller design. For both approaches, we set all Q_i matrices as suggested in [141], $R = 0.1$, and set the first entry of Q_{sync} to 10 and all others to 0 to express our desire to synchronize the cart positions. We adopt the system matrices provided in [141] for the off-the-shelf systems and the ones from [120] for the self-built systems. The respective matrices are used for controller design and estimation. The simulated systems also use the matrices from [141], but we add normally distributed noise with a standard deviation of 10^{-4}.

6.7.3 Results

In the following, we discuss the results for our predictive triggering approach and periodic control. The results are representative for our concrete scenario and may differ in other cases. Due to the design of the physical systems used, in particular the



(a) Periodic. Each agent is allocated the same number of messages regardless of the disturbance.



(b) Predictive. Messages are allocated according to the agents' needs, addressing the disturbance.

FIGURE 6.8: Total number of control messages allocated to each agent throughout the experiments for the predictive and periodic approach. A disturbance is added to agent 11 in the middle of the experiments. Compared to the periodic approach, the predictive approach assigns the control messages based on the current state of the agents.

limited track length of the carts, the estimation error $\hat{e}$ is restricted to a relatively small value, which favors the periodic control approach. In scenarios where $\hat{e}$ can grow larger or is potentially unbounded, we expect higher control performance gains with our predictive approach.

Resource allocation. In Figure 6.8 we show the total number of control messages allocated to each agent throughout one characteristic experiment for each approach. We begin by discussing the first half of the experiment, before the disturbance, and continue with the second half afterwards.

Before the disturbance: With the *periodic* approach, every agent gets the same share of messages. Using the *predictive* approach, instead, some agents get to communicate significantly more or less often than others. This is because the *predictive* approach dynamically allocates messages to agents based on their actual needs. We see that agents with simulated systems (1, 4, 9–20) rarely get messages allocated ($\leq$ 12 messages per agent) because their states can be predicted more accurately. The states of agents with real systems (2, 3, 5–8) cannot be predicted as accurately as those of the simulated systems, so they need to communicate their states more often to achieve a better synchronization. This higher need is effectively accounted for by the *predictive* approach, as visible from the many messages allocated to these agents.

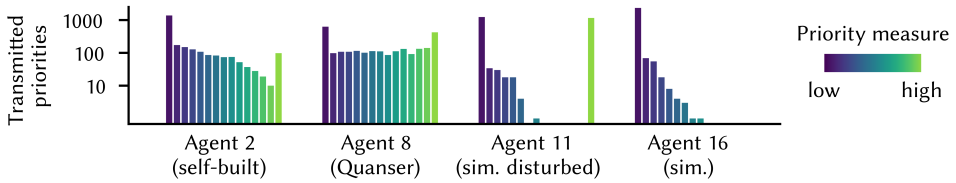


FIGURE 6.9: Per agent distribution of the transmitted priorities for a selected set of agents using the predictive approach. Systems that are more difficult to predict will transmit high priorities more often. Simulated systems transmit mostly low priorities, except agent 11 that is disturbed during half of the experiment.

During the disturbance: In the middle of the experiment the position of agent 11 is fixed at $s = 20\,\text{cm}$, which suddenly leads to very inaccurate state predictions for this agent. This does not affect the message allocation in the *periodic* approach and every agent still gets the same amount of messages. The *predictive* approach, on the other hand, adapts to the situation by allocating agent 11 the most messages among all agents (> 1000 messages) during the second half of the experiment. The instantaneous trigger, that is, $P_i^0(k + \mathrm{H}) > P_\delta$ in (6.10), reduces the number of control messages by about 1–2 %. More savings are expected when the ratio M_C/N increases.

Priority distribution. In the *predictive* approach, the decision about which agent is allocated a message is based on the communication needs, which is reflected by the transmitted priorities. Figure 6.9 shows how often each priority was transmitted during the experiment, for a selected set of agents. The selection includes one representative agent for each type of pendulum (self-built, Quanser, simulated), and additionally the disturbed agent 11. Each histogram indicates how often the respective agent has transmitted low to high priorities. Agent 2 and 8 have different types of real physical systems (self-built vs. off-the-shelf). Although both are cart-pole systems and, thus, have similar dynamics, their individual physical characteristics are different, resulting in different communication needs. Agent 8 transmits higher priorities more often compared to agent 2, implying that the state of agent 8 is less predictable. Agents 11 and 16 have simulated systems, so they show very similar communication needs. However, as soon as we manipulate the cart position of agent 11, its predictions become inaccurate, and it chooses higher priorities more often; in fact, during the disturbance, it chooses the highest priority most often among all agents. Consequently, the *predictive* approach allocates a message to agent 11 in almost every communication round, helping the other agents to better synchronize the cart positions. The priority distribution of the different pendulum types is so characteristic across all our experiments that the type of each pendulum can be identified from this alone. Moreover, these distributions reflect the different communication needs which also shows the heterogeneity of our physical systems.

Control performance. Figure 6.10 shows the control performance by evaluating the cost function (6.4) in each time step. The results are computed using a moving

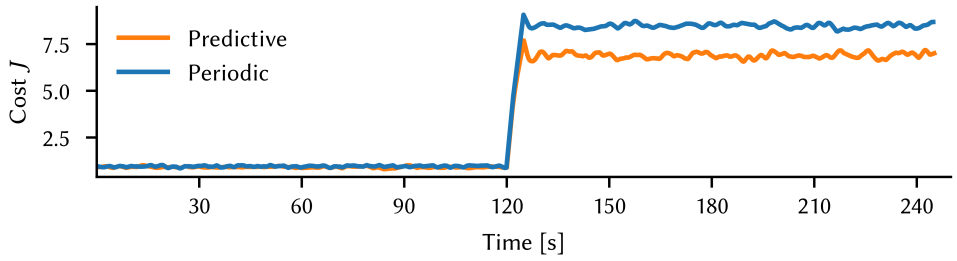


FIGURE 6.10: Control cost over time, averaged across multiple runs per approach. During the disturbance (starting at 120 s), the predictive approach decreases the control cost by around 19 % due to the adaptive allocation of communication resources.

average with a window size of 50 time steps (5 s), and averaged over 3 different runs per approach. In the first half of the experiment, both approaches are close together with ≈5 % lower cost for the *predictive* approach. The distance increases in the second half of the experiments, during the disturbance, where the control cost of our approach is ≈19 % lower, because it adaptively allocates more messages to the disturbed agent 11, thus enabling a better synchronization. This improvement is achieved even though the *periodic* approach can transmit one control message more per communication round ($M_C = 3$) compared to the *predictive* approach ($M_C = 2$).

In our experiments, the network bandwidth reserved for application traffic is relatively high (≈85 %). Conversely, the bandwidth remaining for control messages is small. As a result, the overhead of our *predictive* approach reduces M_C from 3 (with *periodic* control) to 2, which is 33 % less. However, the overhead depends on the concrete experiment settings. For example, in the same scenario but with $M_A = 10$ messages, the periodic approach could use $M_C = 11$ control messages and the predictive approach $M_C = 9$ control messages, with an aggregate overhead of $S = 10$ bytes. In this case, the number of control messages is only reduced by 18 %.

6.A Proof of Theorem 4

Here, we provide the proof of Theorem 4. We start with some preliminaries that will be needed to ensure that the estimation errors $\hat{e}_i(k)$ and $\hat{e}_{ij}$ are well-behaved, what lets us conclude MSB of the overall system.

6.A.1 Preliminaries

To study the behavior of the estimation error, we employ the concept of f-ergodicity. Generally, a stochastic process as (6.1) is said to be ergodic if its sample average and time average coincide. The notion of f-ergodicity is stronger a stronger notion and is used in the context of Markov chains. Intuitively, if a process is f-ergodic, the Markov chain is stationary and the process itself converges to an invariant finite-variance measure over the entire state-space. More formally:

Definition 3 ([123, Ch. 10]). *Let the Markov chain $\Phi = (\Phi(0), \Phi(1), \ldots)$ evolve in the state-space $\mathcal{X}$, which is equipped with some known σ-algebra $\mathcal{B}(\mathcal{X})$. The Markov chain Φ is said to be positive Harris recurrent if*

1) *a non-trivial measure $\nu(B) > 0$ exists for a set $B \in \mathcal{B}$ such that for all $\Phi(0) \in \mathcal{X}$, $P(\Phi(k) \in B, k < \infty) = 1$ holds.*
2) *Φ admits a unique invariant probability measure.*

Definition 4 ([123, Ch. 14]). *Let $f \geq 1$ be a real-valued function in $\mathbb{R}^n$. A Markov chain Φ is said to be f-ergodic, if*

1) *Φ is positive Harris recurrent with unique invariant measure π,*
2) *the expectation $\pi(f) = \int f(\Phi(k))\pi(\mathrm{d}\,\Phi(k))$ is finite,*
3) *$\lim_{k\to\infty} \|P^k(\Phi(0), \cdot) - \pi\|_f = 0$ for every initial value $\Phi(0) \in \mathcal{X}$, where $\|\nu\|_f = \sup_{|g|\leq f} |\nu(g)|$.*

We further define a drift function:

Definition 5 ([117]). *Let $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, and Φ be a Markov chain. For any measurable function V, the drift $\Delta V(\cdot)$ is*

$$\Delta V(\Phi(k)) := \mathbb{E}[V(\Phi(k+1)) \mid \Phi(k)] - V(\Phi(k)), \tag{6.11}$$

with $\Phi(k) \in \mathbb{R}^n$.

We can then establish f-ergodicity.

Theorem 5 (f-Norm Ergodic Theorem [123, Ch. 15]). *Suppose that the Markov chain Φ is ψ-irreducible and aperiodic and let $f(\Phi(k)) \geq 1$ a real-valued function in $\mathbb{R}^n$. If a small set $\mathcal{D}$ and a non-negative real-valued function V exist such that $\Delta V(\Phi(k)) \leq -f(\Phi(k))$, for every $\Phi(k) \in \mathbb{R}^n \setminus \mathcal{D}$, and $\Delta V < \infty$ for $\Phi(k) \in \mathcal{D}$, the Markov chain Φ is f-ergodic.*

The Markov chain defined through the dynamical system (6.1) is both aperiodic and ψ-irreducible. This is the case since the noise distribution $v_i(k)$ is assumed to be absolutely continuous with an everywhere-positive density function. Thus, every subset of the state space $\mathcal{X}$ is reachable within one time-step. Here, ψ is a non-trivial measure on $\mathbb{R}^n$. Further, all compact subsets of an LTI system are small sets [123, Ch. 5].

6.A.2 Stochastic Stability

Equipped with the concepts from the previous section, we now present the stability proof. For this, we first show that the Markov chain induced by the estimation errors $\hat{e}_i(k)$ and $\hat{e}_{ij}$ is f-ergodic and then show that this implies MSB of the overall system.

For analyzing the ergodicity of the error Markov chain we start with defining the function V as the sum of all squared Mahalanobis distances from the equilibrium, i.e.,

$$V(\hat{e}(k)) = \sum_{i=0}^{N} \|\hat{e}_i(k)\|_{\mathrm{M}}^2 + \sum_{i,j=0, i\neq j}^{N} \left\|\hat{e}_{ij}(k)\right\|_{\mathrm{M}}^2. \tag{6.12}$$

Ideally, we would now seek to guarantee $\Delta V(\hat{e}(k))$ to be negative at every time step to invoke f-ergodicity. However, for an event-triggered strategy and a lossy communication channel, this is not possible. We may have rounds, in which none of the agents seeks to communicate as all their errors are low enough. Also, we may have agents that want to communicate but messages being lost. In both cases, the drift may become positive. Thus, we modify the drift definition from (6.11),

$$\Delta V(\hat{e}(k), K) = \mathbb{E}[V(\hat{e}(k+K)) \mid \hat{e}(k)] - V(\hat{e}(k)). \tag{6.13}$$

As discussed in [123, Ch. 19], Theorem 5 can also be used to show f-ergodicity for such modified drift definitions. Here, we choose $K = \mathrm{ceil}(2N/M_{\mathrm{C}})$, where ceil is a function rounding its argument to the next higher integer. The incentive of this is to have an interval that allows each agent to communicate once. Since agents announce their communication needs in advance and only learn about whether they received a slot in the round they receive it, it might happen that an agent is awarded two slots in a row. To also allow for communication of all agents in such cases, we take two times the minimum interval.

In the absence of message loss, we have $\hat{e}_i(k) = \hat{e}_{ij}(k)$ for all k and j. In the following, we, thus, first derive bounds for $\hat{e}_i(k)$ and then show how we deal with errors due to message loss. Note that the estimation error $\hat{e}_i(k+K)$ can be written as a function of $\hat{e}_i(k)$,

$$\begin{aligned}\hat{e}_i(k+K) = \tilde{A}^{K-k}\hat{e}_i(k) \prod_{n=k}^{K}(1-\kappa_i(n)\phi_i(n)) \\ + \sum_{n=k}^{K-1} \tilde{A}^{K-k-1} v_i(K-n) \prod_{r=n+2}^{K}(1-\kappa_i(r)\phi_i(r)),\end{aligned} \tag{6.14}$$

where $\tilde{A} = A + BF_{ii}$.

We now seek to upper bound the expected squared Mahalanobis distance of $\hat{e}_i(k)$. Generally, such bounds can be derived as follows:

Lemma 4. *We can upper bound* $\mathbb{E}[\|\hat{e}_i(k+1)\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)]$ *in case of no communication by*

$$\mathbb{E}[\|\hat{e}_i(k+1)\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)] \leq \left\|\tilde{A}\right\|_2^2 \|\hat{e}_i(k)\|_{\mathrm{M}}^2 + \mathrm{Tr}(I_n). \tag{6.15}$$

Proof. Following (6.14), we have[3]

$$\mathbb{E}[\|\hat{e}_i(k+1)\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)] = \mathbb{E}[\left\|\tilde{A}\hat{e}_i(k) + v_i(k)\right\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)]. \tag{6.16}$$

Leveraging that the noise has zero mean and using the Cauchy-Schwarz inequality, we arrive at

$$\begin{aligned}
&\mathbb{E}[\left\|\tilde{A}\hat{e}_i(k) + v_i(k)\right\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)] \\
&= \mathbb{E}[\left\|\tilde{A}\hat{e}_i(k)\right\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)] + \mathbb{E}[\|v_i(k)\|_{\mathrm{M}}^2] \\
&\leq \left\|\tilde{A}\right\|_2^2 \mathbb{E}[\|\hat{e}_i(k)\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)] + \mathrm{Tr}(\Sigma_{\mathrm{v_i}}^{-1}\Sigma_{\mathrm{v_i}}) \\
&= \left\|\tilde{A}\right\|_2^2 \|\hat{e}_i(k)\|_{\mathrm{M}}^2 + \mathrm{Tr}(I_n).
\end{aligned}$$

□

We now prove f-ergodicity of the Markov chain induced by $\hat{e}_i$.

Theorem 6. *Consider the setting from Theorem 4. Then, for any $\delta \in \mathbb{R}_{\geq 0}$, the Markov chain $\hat{\tilde{e}}_i(k) = [\hat{e}_1(k), \dots, e_{\mathrm{N}}(k)]^\intercal$ is f-ergodic.*

Proof. We let the dynamical system evolve over the time interval $[k, k+K]$ and study all possible outcomes. For this, we partition the agents in mutual exclusive subsets:

- c_1 The agent has or has not transmitted over $[k, k+K]$ and $\|\hat{e}_i(k+K-1)\|_{\mathrm{M}}^2 \leq \delta$;
- c_2 The agent has transmitted successfully at least once over $[k, k+K]$ and $\|\hat{e}_i(k+K-1)\|_{\mathrm{M}}^2 > \delta$;
- c_3 The agent has never been assigned a slot over $[k, k+K]$ and $\|\hat{e}_i(k+K-1)\|_{\mathrm{M}}^2 > \delta$.

We now derive upper bounds for all cases. For case c_1, we know that the estimation error at the last time instant was smaller than the threshold δ. Thus, we can use this and Lemma 4 to arrive at

$$\mathbb{E}[\|\hat{e}_i(k+K)\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)] \leq \delta\left\|\tilde{A}\right\|_2^2 + \mathrm{Tr}(I_n). \tag{6.17}$$

For case c_2, assume that the last transmission instant was at $k = r$. Thus, we know that at $k = r$ the error was reset to 0 and obtain

$$\mathbb{E}[\|\hat{e}_i(k+K)\|_{\mathrm{M}}^2 \mid \hat{e}_i(k)] \leq \sum_{s=r}^{K} \mathrm{Tr}(I_n)\left\|\tilde{A}^{K-s}\right\|_2^2. \tag{6.18}$$

For case 3, we consider two subcases:

- $c_{3\mathrm{a}}$ Over $[k, k+K]$, for at least one k' we had $\|\hat{e}_i(k')\|_{\mathrm{M}}^2 \leq \delta$;

[3] Note that here we ignore the additional information that if the system does not receive a communication slot, its error must have been lower than that of the others. This "negative information" [159] could be used to arrive at sharper bounds.

c_{3b} Over $[k, k+K]$, we had $\|\hat{e}_i(k')\|_M^2 > \delta$ for all k'.

For case c_{3a}, assume that $k = r$ was the last time we had $\|\hat{e}_i(k')\|_M^2 \leq \delta$. Then, we have, again using Lemma 4,

$$\mathbb{E}[\|\hat{e}_i(k+K)\|_M^2 \mid \hat{e}_i(k)] \leq \delta \left\| \tilde{A}^{K-r} \right\|_2^2 + \sum_{s=r}^{K-1} \mathrm{Tr}(I_n) \left\| \tilde{A}^{K-s-1} \right\|_2^2. \tag{6.19}$$

For case c_{3b}, we have agents whose error was above the threshold δ in every time-step, but never got a slot. As the scheduling rule always awards resources to the agents with the highest error, we can upper bound the error of agents in c_{3b} with the worst-case error of the agents in c_2,

$$\mathbb{E}[\|\hat{e}_i(k+K)\|_M^2 \mid \hat{e}_i(k)] \leq \max_{i \in c_2} \mathbb{E}[\|\hat{e}_i(k+K)\|_M^2 \mid \hat{e}_i(k)]. \tag{6.20}$$

The drift can now be upper bounded as

$$\begin{aligned}
\Delta V(\hat{\hat{e}}_i(k), K) \leq & \sum_{i \in c_1, c_2, c_3} \mathbb{E}[\|\hat{e}_i(k+K)\|_M^2 \mid \hat{e}_i(k)] - V(\hat{e}(k)) \\
& \leq \sum_{i \in c_1} \left(\delta \left\| \tilde{A} \right\|_2^2 + \mathrm{Tr}(I_n) \right) + \sum_{i \in c_2} \sum_{n=r_i}^{K} \mathrm{Tr}(I_n) \left\| \tilde{A}^{K-n} \right\|_2^2 \\
& + \sum_{i \in c_{3a}} \left(\delta \left\| \tilde{A}^{K-r} \right\|_2^2 + \sum_{n=r}^{K-1} \mathrm{Tr}(I_n) \left\| \tilde{A}^{K-n-1} \right\|_2^2 \right) \\
& + \sum_{i \in c_{3b}} \max_{j \in c_2} \left(\sum_{n=r_j}^{K} \mathrm{Tr}(I_n) \left\| \tilde{A}^{K-n} \right\|_2^2 \right) - V(\hat{e}(k)) \\
& = \zeta - V(\hat{\hat{e}}_i(k)),
\end{aligned} \tag{6.21}$$

where ζ represents all bounded terms stemming from cases c_1, c_2, and c_3. We can then define $f(\hat{\hat{e}}(k)) = 1 + \epsilon V(\hat{\hat{e}}(k))$, with $\epsilon \in (0, 1)$. Since $V(\hat{\hat{e}}(k)) \geq 0$, it follows that $f(\hat{\hat{e}}(k)) \geq 1$. As $v(\hat{\hat{e}}(k))$ grows with $\hat{\hat{e}}(k)$ while ζ is constant, we can further find a small set $\mathcal{D}$ and an ϵ such that $\Delta V(\hat{e}(k), K) \leq -f$, which proves f-ergodicity according to Theorem 5 in the absence of message loss. □

With this, we have shown f-ergodicity of $\hat{\hat{e}}_i(k)$. However, there might, due to message loss, still be a divergence between $\hat{e}_i(k)$ and $\hat{e}_{ij}(k)$, that is, the estimates that other agents have about agent i's state.

Corollary 1. *Consider the same setting as in Theorem 6. Then, the Markov chain* $\hat{e}_= [\hat{e}_1(k), \ldots, \hat{e}_{1N}, \hat{e}_{21}, \ldots, e_N(k)]^\mathsf{T}$ *is f-ergodic under i.i.d. message loss.*

Proof. First, assume that only messages of agent i might be lost, but not those of other agents. Then, we can account for lost messages, following the strategy of [116], by extending the horizon until the next successful message transmission of agent i. That

is, we consider $k \to k+K+m^*$ instead of $k \to k+K$. We then have

$$\mathbb{E}[\left\|\hat{e}_{ij}(k+K+m^*)\right\|_{\mathrm{M}}^2 \mid \hat{e}_{ij}(k)] \leq \mathrm{Tr}(I_n). \tag{6.22}$$

Thus, for any finite m^*, the estimation error is bounded. Now, consider the case $m \to \infty$ without a single message being transmitted successfully. Due to the i.i.d. property of message loss, the probability of losing m_{c} messages in a row is $(1-p_{\mathrm{c}})^{m_{\mathrm{c}}}$, where p_{c} is the probability of successfully transmitting a message. Thus, we have

$$\begin{aligned}&\mathbb{E}[\left\|\hat{e}_{ij}(k+K+m)\right\|_{\mathrm{M}}^2 \mid \hat{e}_{ij}(k))] \leq \\ &(1-p_{\mathrm{c}})^{m_{\mathrm{c}}}\left(\left\|\hat{e}_{ij}(k)\right\|_{\mathrm{M}}^2\left\|\tilde{A}^{K+m}\right\|_2^2 + \sum_{n=k}^{K+m-1}\mathrm{Tr}(I_n)\left\|\tilde{A}^{K-n-1}\right\|_2^2\right).\end{aligned} \tag{6.23}$$

The variable m_{c} here denotes the amount of times agent i tried to transmit its state in the interval $k \to k+K+m$. Agent i competes for a slot, if $\|\hat{e}_i(k)\|_{\mathrm{M}} > \delta$ for some k and if it gets a slot the error is reset. Thus, the time at which $\hat{e}_i(k)$ exceeds the threshold δ can be interpreted as a stopping time, which has finite expected value [57, Section 2]. Therefore, agent i will infinitely often compete for slots as $m \to \infty$. The same holds for all other agents and since we assume homogeneous agents, each will be assigned the same amount of slots as $m \to \infty$ due to the law of large numbers. In conclusion, we have $m_{\mathrm{c}} \to \infty$ as $m \to \infty$. As we further have $\left\|\tilde{A}\right\|_2 < 1$ and $p_{\mathrm{c}} > 0$ (cf. the assumptions stated at the beginning of Section 6.6.2), (6.23) goes to zero as m^* goes to infinity.

In practice, also other agents may lose messages. Then, the estimation error $\hat{e}_{ij}$ might grow not only because agent j is missing information about agent i, but also because their information about other agents states might differ. Since the control input depends also on these other states, this might lead to diverging predictions. In this case, we need to add another term

$$\sum_{\ell \in \Omega_i \setminus \{i,j\}} \left\|\hat{e}_{i\ell}(k+K+m^*-1) - \hat{e}_{j\ell}(k+K+m^*-1)\right\|_{\mathrm{M}}^2\left\|\tilde{A}\right\|_2^2 \tag{6.24}$$

to (6.22). Also these terms can be bounded. In the beginning, we assume agents i and j to start with the same initial guess about agent ℓ, i.e., (6.24) is zero. We can now, similar as above, extend the horizon such that at $k+K+m^*$, i successfully transmitted a message to j and at some k' within $k \to k+K+m^*$ both i and j successfully received an update from ℓ. If then i receives another update within at some $k'' > k'$ while j did not, the error is bounded by

$$\begin{aligned}&\mathbb{E}[\left\|\hat{e}_{i\ell}(k+K+m^*) - \hat{e}_{j\ell}(k+K+m^*)\right\|_{\mathrm{M}}^2 \mid \hat{e}_{ij}(k)] \leq \\ &\sum_{s=k'}^{K+m^*}\mathrm{Tr}(I_n)\left\|\tilde{A}^{K+m^*-s}\right\| + \sum_{s=k''}^{K+m^*}\mathrm{Tr}(I_n)\left\|\tilde{A}^{K+m^*-s}\right\|.\end{aligned} \tag{6.25}$$

The only possibility for the error to grow without bounds is for agent i to receive an

update of ℓ at some point while j never receives one. For the same arguments as above, the probability of that happening converges to zero faster than the error grows as m goes to infinity.

Thus, also in the case of message loss, the Markov chain is f-ergodic. □

Remark 3. *Under additional assumptions, the results can be extended to systems for which* $\left\|\tilde{A}\right\|_2 > 1$. *In that case, it needs to be ensured that the probability of successfully delivering a message grows faster than the estimation error.*

With this, we can now prove stability of the overall system. For MSB following Definition 2, we need $\mathbb{E}[\|x(k)\|_2^2]$ to be bounded. The individual system without estimation error is exponentially stable, that is, its second moment would go to 0 at an exponential rate. The second moment of the estimation error can be upper bounded, independent of the state of the system, and also the noise variance is constant and independent of the system's state. That is, we essentially have an exponentially stable system with a constant disturbance. Thus, also the second moment of the system state is bounded [25, Ch. 7.6d].

6.B Usage of the Network Bandwidth for Control

The network bandwidth available for control traffic (M_C) depends on the communication demand of the application (M_A) and other scenario parameters such as the required update interval. In our experiments in Section 6.7.2 the communication period is 100 ms, and in each round $M_A = 18$ application messages are sent. In between two communication rounds, each agent does computations based on the received data, and provides the current priority and state information for the next communication round. For these calculations and some additional buffer time we need to reserve 24 ms, thus, 76 ms remain for the communication system to exchange data. Based on our BLE experiments in Section 6.5.1, we can derive that Mixer needs on average about 9.5 slots per message. In the 2 Mbps BLE mode it takes 4 µs to transmit one byte. With respect to the packet size and other unavoidable delays (see Chapter 2), for example, switching the radio from receive to transmit mode and vice versa, the resulting duration of one Mixer slot is 380 µs. We can now calculate M_C, first for the *periodic* approach and then for our *predictive* approach.

6.B.1 Periodic

The *periodic* approach has no additional overhead and M_C can be straightforwardly calculated with

$$communication\_time \geq (M_A + M_C) * slots\_per\_message * slot\_time. \tag{6.26}$$

Using the specific values of our experiment, we have

$$76\,000\,\mu s \geq (18 + M_C) * 9.5 * 380\,\mu s,$$

resulting in a value of $M_C = 3$ control messages per communication round.

6.B.2 Predictive

The relationships are a bit more complex in our *predictive* approach, because M_C and the duration of a slot in Mixer are interrelated. The slot_time term in (6.26) additionally depends on the aggregate overhead S as calculated in (6.9). This requires M_C to be calculated iteratively by choosing a value for M_C, then calculating S, and finally checking if this is below the maximum communication time. In our experiments, with $N = 20$, $W_P = 4$, and $M_C = 2$ we get $S = 5$ additional bytes per packet. As a result, slot_time increases from 380 µs to 400 µs and we get

$$76\,000\,\mu s \geq (18 + 2) * 9.5 * 400\,\mu s.$$

Postscript

This chapter has presented a novel co-design of distributed control and wireless communication that successfully handles overload situations, where more agents want to transmit information than the communication system can support. Such situations are inevitable as emerging CPS applications become increasingly sophisticated, featuring more agents and higher traffic demands. Prior work using ETC or STC designs can mitigate the overload problem by reducing the generated traffic on average. However, they cannot avoid overload situations as potentially all agents want to transmit simultaneously, leading to unpredictable message losses. By contrast, our approach efficiently distributes the agents' communication needs with Mixer (see Chapter 2) to allocate the available bandwidth to agents with the highest need. Furthermore, prioritizing traffic demands enables us to provide formal stability guarantees for the entire CPS, including control and communication systems. Experiments on a CPS testbed successfully demonstrate the synchronization of 20 cart-pole systems over a multi-hop network with severely limited communication resources.

7

Conclusion and Outlook

CPS are considered a driving force behind the next industrial revolution [144, 184]. They integrate computing elements and communication into our environment to monitor and control physical processes through sensors and actuators. As a result, CPS will transform how humans and machines interact with the world. In this context, many envisioned CPS applications rely on the enormous potential of wireless communication, which offers unmatched flexibility, substantial cost savings, and opens up completely new possibilities.

However, many of these applications also have high dependability, efficiency, and adaptivity requirements. Specifically, CPS should operate as expected even if individual parts fail; they should adapt to continuously changing conditions caused by variations within the system or the environment; and they should efficiently support dense and large-scale deployments. These requirements are in stark contrast to the characteristics of wireless communication. For instance, frequent and correlated message losses, as well as long and varying delays significantly complicate the control design. In addition, node mobility and other network dynamics lead to time-varying communication links that create a large coordination overhead.

Existing works that have practically addressed the challenges of wireless CPS can only satisfy a few of the many requirements mentioned above. For example, slow physical systems with update intervals in the order of seconds have loose requirements and thus massively relieve timing and reliability constraints. By contrast, the control of fast physical systems is limited to small single-hop deployments. A major problem is the properties of state-of-the-art wireless communication protocols. Their performance largely depends on up-to-date information about the network state, which becomes an unmanageable effort in dynamic networks, leading to an unreliable and unpredictable message exchange and, thus, deteriorating the control performance.

7.1 Contributions

This thesis addresses the shortcomings of prior approaches and makes contributions in two areas. In the first part, we developed the networking foundation to meet the communication demands of emerging CPS. Then, based on the co-design of communication and control, we built and evaluated practical wireless CPS with theoretical guarantees, using classical time-triggered control as well as modern event-based approaches. The achievements of this thesis are summarized below.

Efficient and scalable communication with Mixer. We presented the many-to-all communication primitive Mixer, which suits the diverse needs of emerging wireless CPS. Compared to prior approaches, which rely on routing or flooding mechanisms, Mixer provides a novel combination of ST and RLNC to operate largely independently from the network topology and to achieve order-optimal scaling behavior. This combination makes Mixer widely applicable, supporting dynamic networks and any communication pattern, from one-to-one to all-to-all, an essential requirement in distributed CPS.

Highly dependable wireless communication with Butler. With our work on Butler, we addressed the weak point of Mixer and other ST-based protocols and significantly improved their dependability. Without exception, all ST-based protocols rely on tight time synchronization and use a dedicated node for this task. Unfortunately, a failure of this node results in a complete communication outage. However, nodes can fail for various reasons, such as depleted batteries, and failures will be common in large-scale systems. Therefore, a dependable communication service should enable the message exchange between functioning nodes, irrespective of individual failures. To achieve this goal, we developed Butler, a lightweight and distributed synchronization mechanism that can synchronize the network on demand, eliminating the need for a dedicated node and, thus, the single point of failure. In addition, we have formally proven Butler's correctness and evaluated its behavior and performance in testbed experiments.

Wireless feedback control with stability guarantees. We were the first to demonstrate fast feedback control over low-power wireless multi-hop networks with update intervals of 20–50 ms. Existing solutions are either limited to single-hop networks or physical processes with slow dynamics and update intervals of several seconds, leaving a large gap to the requirements of envisioned CPS. We proposed a novel co-design to fill this gap, realizing a tight integration of communication and control. Our approach tames wireless imperfections to the extent possible and addresses the resulting key properties in the control design. The careful integration of communication and control tasks on a wireless embedded platform provides real-time operation and enables a thorough end-to-end analysis of the entire CPS. As a result, we could formally guarantee closed-loop stability and also validate these guarantees through real-world experiments on a CPS testbed featuring real physical systems and wireless networks. Thus, our work is an important step toward realizing the CPS vision.

End-to-end energy savings via control-guided communication. We have demonstrated for the first time distributed, STC over wireless multi-hop networks with energy savings and reallocation of resources at fast update intervals of tens of milliseconds. While prior works could achieve significant savings on the control side, they could not transfer those savings over to the communication side in order to gain an end-to-end benefit. Our control-guided communication approach targets this deficit with a new co-design, where the control system predicts future resource demands and informs the communication system ahead of time. The practical evaluation on a CPS testbed validates that the communication system can use its resources most efficiently and save energy whenever possible.

Scaling beyond bandwidth limitations. Finally, we presented a novel co-design using distributed control and Mixer to cope with the ever-increasing bandwidth requirements. Our design can successfully handle overload situations where the available communication bandwidth is insufficient to meet the demand. Existing works using ETC or STC can reduce the communication demand on average; however, they cannot prevent overload situations that result in message loss and unpredictable control performance. In contrast, we use state predictions to derive priorities and efficiently distribute them among all agents using Mixer. Thus, we can prioritize communication demands in a distributed way and serve the agents with the greatest needs first while respecting the available bandwidth at each time step. Furthermore, because of the prioritization, we can guarantee closed-loop stability for heterogeneous physical systems. We demonstrated the improved control performance in experiments with 20 cart-pole systems and heavily constrained bandwidth.

7.2 Future Directions

A comprehensive understanding of synchronous transmissions. ST provide a radically different perspective on low-power wireless communication, away from unflexible routing-based approaches, and are an essential concept in this thesis. However, ST still need to be wholly understood, although significant progress has been made in recent years, such as in analyzing beating patterns [182]. Nevertheless, with a thorough understanding, PHYs could be adapted using, for example, more suitable modulation schemes and transceiver designs to make better use of the capture effect and other PHY phenomena. In addition, the unused potential could be tapped by developing specialized hardware, such as radios that can receive multiple packets simultaneously [85].

Despite the significantly better performance of ST-based protocols compared to link-based protocols in many scenarios, their behavior often relies on simple measures, such as the number of discovered neighbors or randomness. Unfortunately, many of these measures provide only a very rough estimate of the actual situation in the real world. The overall packet reception rate and bandwidth utilization could be further improved with better measures, which are often closely intertwined with how we characterize

networks. Typical properties such as diameter, node degree, and link quality are static metrics that describe the network state at a specific time but ignore their change rate caused by network dynamics. Measures that can accurately capture network dynamics and that are specifically tailored to the characteristics of ST could improve on the state of the art.

Event-triggered and learning-based wireless control. In general, communication resources have to be allocated in advance to use the available communication bandwidth efficiently. The allocation presupposes that the different communication needs are known (see periodic control in Chapter 4) or predictable (see STC in Chapter 5, and predictive triggering in Chapter 6). In contrast, truly ETC makes instantaneous communication decisions, providing the highest flexibility and shortest reaction time [63]. However, combining instantaneous decisions with wireless communication, which requires planning, remains an open challenge.

Based on recent advances in machine learning and data science, learning-based approaches can improve various components of CPS, such as communication and control. Learning enables continuously adapting to changing conditions and dealing with, for example, uncertainty or changes in the physical system or control task at run-time. Moreover, communication protocols such as Mixer that rely on simple measures and heuristics can benefit significantly from learning and understanding the characteristics of the network and its environment. Learning the system's behavior and key indicators is also valuable for fault-tolerance mechanisms. The early detection of abnormal behavior enables appropriate actions before failures may deteriorate the functionality and performance of CPS. However, learning is a resource-intensive task, and appropriate models that can run on low-power embedded devices and strike the balance between computational complexity and accuracy are needed. Furthermore, the challenge of obtaining enough and also useful training data is essential to learn correct behavior.

From research to industry. The potential of wireless communication in control applications is well known, and the industry is willing to exploit it. Nevertheless, the current situation is far from the envisioned CPS scenarios, and wireless solutions are neither found in time- nor safety-critical systems [17]. Sporadically, there are wireless approaches, such as demonstrated in a paper mill [2], but this remains rare. Mainly dependability concerns slow down the adoption in the rather conservative industrial sector [184]. A necessary step toward a broader acceptance is the evaluation of realistic CPS, an important contribution in this thesis; however, setting up such testbeds is a time- and cost-intensive endeavor. Therefore, public CPS testbeds that enable comparing different approaches in a reproducible environment would greatly benefit the research community, accompanied by advances in wireless technology, such as faster and more robust PHYs. In addition to experimental demonstrations, accurate formal models of wireless CPS are needed to bridge theory and practice. In that context, this thesis is a first step toward emerging wireless CPS, showing the feasibility and focussing on fundamental requirements and properties.

Bibliography

[1] 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)*, pages 283–288, Marrakech, Morocco, 2017. doi:10.1145/3019612.3019710.

[2] Anders Ahlén, Johan Åkerberg, Markus Eriksson, Alf J. Isaksson, Takuya Iwaki, Karl Henrik Johansson, Steffi Knorn, Thomas Lindh, and Henrik Sandberg. Toward Wireless Control in Industrial Process Automation: A Case Study at a Paper Mill. *IEEE Control Systems Magazine*, 39(5):36–57, 2019. doi:10.1109/MCS.2019.2925226.

[3] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000. doi:10.1109/18.850663.

[4] Johan Åkerberg, Mikael Gidlund, and Mats Björkman. Future Research Challenges in Wireless Sensor and Actuator Networks Targeting Industrial Automation. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 410–415, Lisbon, Portugal, 2011. doi:10.1109/INDIN.2011.6034912.

[5] Beshr Al Nahas, Simon Duquennoy, and Olaf Landsiedel. Network-Wide Consensus Utilizing the Capture Effect in Low-Power Wireless Networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1:1–1:14, Delft, Netherlands, 2017. doi:10.1145/3131672.3131685.

[6] Rajeev Alur, Alessandro D'Innocenzo, Karl Henrik Johansson, George J. Pappas, and Gera Weiss. Compositional Modeling and Analysis of Multi-Hop Control Networks. *IEEE Transactions on Automatic Control*, 56(10):2345–2357, 2011. doi:10.1109/TAC.2011.2163873.

[7] Brian D. O. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Books on Engineering. Dover Publications, 2007. ISBN 9780486457666. URL `https://books.google.de/books?id=PZToAAAACAAJ`.

[8] Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Dover Books on Electrical Engineering. Dover Publications, 2012. ISBN 9780486136899. URL `https://books.google.de/books?id=iYMqLQp49UMC`.

[9] José Araújo, Adolfo Anta, Manuel Mazo Jr., João Faria, Aitor Hernandez, Paulo Tabuada, and Karl Henrik Johansson. Self-Triggered Control over Wireless Sensor and Actuator Networks. In *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–9, Barcelona, Spain, 2011. doi:10.1109/DCOSS.2011.5982135.

[10] José Araújo, Manuel Mazo Jr., Adolfo Anta, Paulo Tabuada, and Karl Henrik Johansson. System Architectures, Protocols and Algorithms for Aperiodic Wireless Control Systems. *IEEE Transactions on Industrial Informatics*, 10(1):175–184, 2014. doi:10.1109/TII.2013.2262281.

[11] Karl Johan Åström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008. ISBN 9780691135762. URL `https://books.google.de/books?id=G8UGmQEACAAJ`.

[12] Karl Johan Åström and Björn Wittenmark. *Computer-Controlled Systems: Theory and Design*. Prentice-Hall Information and System Sciences Series. Prentice Hall, 3rd edition, 1997. ISBN 9780133148992. URL `https://books.google.de/books?id=5Z0QAQAAMAAJ`.

[13] M. Hadi Balaghi I, Duarte J. Antunes, Mohammad Hossein Mamduhi, and Sandra Hirche. A Decentralized Consistent Policy for Event-triggered Control over a Shared Contention-based Network. In *IEEE Conference on Decision and Control (CDC)*, pages 1719–1724, Miami, Florida, USA, 2018. doi:10.1109/CDC.2018.8619584.

[14] 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):65–86, 2014. doi:10.1109/MCS.2014.2320394.

[15] 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 (CPS-Bench)*, pages 13–18, Porto, Portugal, 2018. doi:10.1109/CPSBench.2018.00009.

[16] Dominik Baumann, Fabian Mager, Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. Fast Feedback Control over Multi-Hop Wireless Networks with Mode Changes and Stability Guarantees. *ACM Transactions on Cyber-Physical Systems*, 4(2):18:1–18:32, 2019. doi:10.1145/3361846.

[17] Dominik Baumann, Fabian Mager, Ulf Wetzker, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. Wireless Control for Smart Manufacturing: Recent Approaches and Open Challenges. *Proceedings of the IEEE*, 109(4):441–467, 2021. doi:10.1109/JPROC.2020.3032633.

[18] İlker Bekmezci, Ozgur Koray Sahingoz, and Şamil Temel. Flying Ad-Hoc Networks (FANETs): A survey. *Ad Hoc Networks*, 11(3):1254–1270, 2013. doi:10.1016/j.adhoc.2012.12.004.

[19] Bart Besselink, Valerio Turri, Sebastian H. van de Hoef, Kuo-Yun Liang, Assad Alam, Jonas Mårtensson, and Karl Henrik Johansson. Cyber-Physical Control of Road Freight Transport. *Proceedings of the IEEE*, 104(5):1128–1141, 2016. doi:10.1109/JPROC.2015.2511446.

[20] Laksh Bhatia, Ivana Tomić, Anqi Fu, Michael Breza, and Julie A. Mccann. Control Communication Co-Design for Wide Area Cyber-Physical Systems. *ACM Transactions on Cyber-Physical Systems*, 5(2):18:1–18:27, 2021. doi:10.1145/3418528.

[21] Naveed Anwar Bhatti, Muhammad Hamad Alizai, Affan A. Syed, and Luca Mottola. Energy Harvesting and Wireless Transfer in Sensor Network Applications: Concepts and Experiences. *ACM Transactions on Sensor Networks*, 12(3): 24:1–24:40, 2016. doi:10.1145/2915918.

[22] Vincent D. Blondel and John N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000. doi:10.1016/S0005-1098(00)00050-9.

[23] Olfa Boubaker. The inverted Pendulum: A fundamental Benchmark in Control Theory and Robotics. In *International Conference on Education and e-Learning Innovations (ICEELI)*, Sousse, Tunisia, 2012. doi:10.1109/ICEELI.2012.6360606.

[24] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, 1994. ISBN 9780898714852. doi:10.1137/1.9781611970777.

[25] Frank M. Callier and Charles A. Desoer. *Linear System Theory*. Springer Science & Business Media, 2012. ISBN 9781461209577. URL `https://books.google.de/books?id=9xTSBwAAQBAJ`.

[26] Berta Carballido Villaverde, Susan Rea, and Dirk Pesch. InRout – A QoS aware route selection algorithm for industrial wireless sensor networks. *Ad Hoc Networks*, 10(3):458–478, 2012. doi:10.1016/j.adhoc.2011.07.015.

[27] Matteo Ceriotti, Michele Corrà, Leandro D'Orazio, Roberto Doriguzzi, Daniele Facchin, Stefan Guna, Gian Paolo Jesi, Renato Lo Cigno, Luca Mottola, Amy L. Murphy, Massimo Pescalli, Gian Pietro Picco, Denis Pregnolato, and Carloalberto Torghele. 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)*, pages 187–198, Chicago, Illinois, USA, 2011. URL `https://ieeexplore.ieee.org/abstract/document/5779037`.

[28] Anton Cervin. *Integrated Control and Real-Time Scheduling*. PhD thesis, Department of Automatic Control, Lund Institute of Technology (LTH), 2003. URL https://portal.research.lu.se/en/publications/integrated-control-and-real-time-scheduling.

[29] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 169–180, Kyoto, Japan, 2007. doi:10.1145/1282380.1282400.

[30] Timothy H. Chung, Michael R. Clement, Michael A. Day, Kevin D. Jones, Duane Davis, and Marianna Jones. Live-Fly, Large-Scale Field Experimentation for Large Numbers of Fixed-Wing UAVs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1255–1262, Stockholm, Sweden, 2016. doi:10.1109/ICRA.2016.7487257.

[31] James R. Clapper, John J. Young, James E. Cartwright, and John G. Grimes. Unmanned Systems Roadmap 2007-2032. Technical report, US Department of Defense, 2007. URL https://www.globalsecurity.org/intell/library/reports/2007/dod-unmanned-systems-roadmap_2007-2032.pdf.

[32] Peter Corke, Tim Wark, Raja Jurdak, Wen Hu, Philip Valencia, and Darren Moore. Environmental Wireless Sensor Networks. *Proceedings of the IEEE*, 98 (11):1903–1917, 2010. doi:10.1109/JPROC.2010.2068530.

[33] Nikolaus Correll, Prabal Dutta, Richard Han, and Kristofer Pister. Wireless Robotic Materials. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 24:1–24:6, Delft, Netherlands, 2017. doi:10.1145/3131672.3131702.

[34] Sanjoy Dasgupta and Anupam Gupta. An Elementary Proof of a Theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. doi:10.1002/rsa.10073.

[35] Brian Dawkins. Siobhan's Problem: The Coupon Collector Revisited. *The American Statistician*, 45(1):76–82, 1991. doi:10.1080/00031305.1991.10475772.

[36] Supratim Deb, Muriel Médard, and Clifford Choute. Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering. *IEEE Transactions on Information Theory*, 52(6):2486–2507, 2006. doi:10.1109/TIT.2006.874532.

[37] Burak Demirel, Arunselvan Ramaswamy, Daniel E. Quevedo, and Holger Karl. DeepCAS: A Deep Reinforcement Learning Algorithm for Control-Aware Scheduling. *IEEE Control Systems Letters*, 2(4):737–742, 2018. doi:10.1109/LCSYS.2018.2847721.

[38] Statista Research Department. M2M (machine-to-machine) – Statistics & Facts, 2021. URL https://www.statista.com/topics/1843/m2m-machine-to-machine/. Accessed: 2022-12-13.

[39] Patricia Derler, Edward A. Lee, and Alberto Sangiovanni Vincentelli. Modeling Cyber-Physical Systems. *Proceedings of the IEEE*, 100(1):13–28, 2012. doi:10.1109/JPROC.2011.2160929.

[40] Manjunath Doddavenkatappa, Mun Choon Chan, and A. L. Ananda. Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed. In *Testbeds and Research Infrastructure. Development of Networks and Communities (TridentCom)*, pages 302–316, Shanghai, China, 2012. doi:10.1007/978-3-642-29273-6_23.

[41] Manjunath Doddavenkatappa, Mun Choon Chan, and Ben Leong. Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 269–282, Lombard, Illinois, USA, 2013. URL `https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/doddavenkatappa`.

[42] Wan Du, Jansen Christian Liando, Huanle Zhang, and Mo Li. When Pipelines Meet Fountain: Fast Data Dissemination in Wireless Sensor Networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 365–378, Seoul, South Korea, 2015. doi:10.1145/2809695.2809721.

[43] 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)*, pages 337–350, Seoul, South Korea, 2015. doi:10.1145/2809695.2809714.

[44] Johan Eker, Anton Cervin, and Andreas Hörjel. Distributed Wireless Control Using Bluetooth. *IFAC Proceedings Volumes*, 34(22):360–365, 2001. doi:10.1016/S1474-6670(17)32965-8.

[45] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 263–270, Seattle, Washington, USA, 1999. doi:10.1145/313451.313556.

[46] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart Grid – The New and Improved Power Grid: A Survey. *IEEE Communications Surveys & Tutorials*, 14(4):944–980, 2012. doi:10.1109/SURV.2011.101911.00087.

[47] William Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, 3rd edition, 1968. ISBN 9780471257080. URL `https://books.google.de/books?id=wYkQAQAAIAAJ`.

[48] 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)*, pages 73–84, Chicago, Illinois, USA, 2011. URL `https://ieeexplore.ieee.org/abstract/document/5779066`.

[49] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. Low-Power Wireless Bus. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–14, Toronto, Ontario, Canada, 2012. doi:10.1145/2426656.2426658.

[50] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. Virtual Synchrony Guarantees for Cyber-physical Systems. In *IEEE International Symposium on Reliable Distributed Systems (SRDS)*, pages 20–30, Braga, Portugal, 2013. doi:10.1109/SRDS.2013.11.

[51] Christina Fragouli, Jörg Widmer, and Jean-Yves Le Boudec. Efficient Broadcasting Using Network Coding. *IEEE/ACM Transactions on Networking*, 16(2):450–463, 2008. doi:10.1109/TNET.2007.901080.

[52] Bob Frankston. Consumer Technology Versus 5G. *IEEE Consumer Electronics Magazine*, 10(2):43–50, 2021. doi:10.1109/MCE.2020.3037418.

[53] Guillermo Gallego, Carlos Cuevas, Raúl Mohedano, and Narciso García. On the Mahalanobis Distance Classification Criterion for Multidimensional Normal Distributions. *IEEE Transactions on Signal Processing*, 61(17):4387–4396, 2013. doi:10.1109/TSP.2013.2269047.

[54] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync Protocol for Sensor Networks. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 138–149, Los Angeles, California, USA, 2003. doi:10.1145/958491.958508.

[55] Konstantinos Gatsis, Alejandro Ribeiro, and George J. Pappas. Optimal Power Management in Wireless Control Systems. *IEEE Transactions on Automatic Control*, 59(6):1495–1510, 2014. doi:10.1109/TAC.2014.2305951.

[56] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection Tree Protocol. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–14, Berkeley, California, USA, 2009. doi:10.1145/1644038.1644040.

[57] Denis S. Grebenkov. First exit times of harmonically trapped particles: a didactic review. *Journal of Physics A: Mathematical and Theoretical*, 48(1), 2014. doi:10.1088/1751-8113/48/1/013001.

[58] Pulkit Grover. Information Structures, the Witsenhausen Counterexample, and Communicating Using Actions. *Encyclopedia of Systems and Control*, pages 567–571, 2015. doi:10.1007/978-1-4471-5058-9_148.

[59] Bernhard Haeupler. Analyzing Network Coding Gossip Made Easy. In *ACM Symposium on Theory of Computing (STOC)*, pages 293–302, San Jose, California, USA, 2011. doi:10.1145/1993636.1993676.

[60] Bernhard Haeupler. Analyzing Network Coding (Gossip) Made Easy. *Journal of the ACM*, 63(3):26:1–26:22, 2016. doi:10.1145/2629696.

[61] 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):2624–2661, 2016. doi:10.1109/COMST.2016.2560343.

[62] Jianping He, Peng Cheng, Ling Shi, Jiming Chen, and Youxian Sun. Time Synchronization in WSNs: A Maximum-Value-Based Consensus Approach. *IEEE Transactions on Automatic Control*, 59(3):660–675, 2014. doi:10.1109/TAC.2013.2286893.

[63] W.P.M.H. Heemels, Karl Henrik Johansson, and Paulo Tabuada. An Introduction to Event-triggered and Self-triggered Control. In *IEEE Conference on Decision and Control (CDC)*, pages 3270–3285, Maui, Hawaii, USA, 2012. doi:10.1109/CDC.2012.6425820.

[64] Janus Heide, Morten V. Pedersen, Frank H. P. Fitzek, and Muriel Médard. On Code Parameters and Coding Vector Representation for Practical RLNC. In *IEEE International Conference on Communications (ICC)*, pages 1–5, Kyoto, Japan, 2011. doi:10.1109/icc.2011.5963013.

[65] Hermann Hellwagner and Christian Bettstetter. Networking research challenges in multi-UAV systems, 2016. URL `https://bettstetter.com/uav-networking-challenges/`. Accessed: 2022-12-13.

[66] Mario Hermann, Tobias Pentek, and Boris Otto. Design Principles for Industrie 4.0 Scenarios. In *Hawaii International Conference on System Sciences (HICSS)*, pages 3928–3937, Koloa, Hawaii, USA, 2016. doi:10.1109/HICSS.2016.488.

[67] Aitor Hernandez, João Faria, José Araújo, Pangun Park, Henrik Sandberg, and Karl Henrik Johansson. Inverted Pendulum Control over an IEEE 802.15.4 Wireless Sensor and Actuator Network. In *European Conference on Wireless Sensor Networks (EWSN)*, Bonn, Germany, 2011. URL `http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A504854`.

[68] Shahab Heshmati-Alamdari, Alina Eqtami, George C. Karras, Dimos V. Dimarogonas, and Kostas J. Kyriakopoulos. A Self-triggered Visual Servoing Model Predictive Control Scheme for Under-actuated Underwater Robotic Vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3826–3831, Hong Kong, China, 2014. doi:10.1109/ICRA.2014.6907414.

[69] João P. Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A Survey of Recent Results in Networked Control Systems. *Proceedings of the IEEE*, 95(1):138–162, 2007. doi:10.1109/JPROC.2006.887288.

[70] Tracey Ho, Muriel Médard, Ralf Koetter, David R. Karger, Michelle Effros, Jun Shi, and Ben Leong. A Random Linear Network Coding Approach to Multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006. doi:10.1109/TIT.2006.881746.

[71] IEEE Standard for Low-Rate Wireless Networks. IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015), 2020. doi:10.1109/IEEESTD.2020.9144691.

[72] Timofei Istomin, Csaba Kiraly, and Gian Pietro Picco. Is RPL Ready for Actuation? A Comparative Evaluation in a Smart City Scenario. In *European Conference on Wireless Sensor Networks (EWSN)*, pages 291–299, Porto, Portugal, 2015. doi:10.1007/978-3-319-15582-1_22.

[73] 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 Networked Sensor Systems (SenSys)*, pages 83–95, Stanford, California, USA, 2016. doi:10.1145/2994551.2994558.

[74] 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)*, pages 167–178, Porto, Portugal, 2016. doi:10.1109/RTSS.2016.025.

[75] 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 Conference & Exhibition (DATE)*, pages 865–868, Dresden, Germany, 2018. doi:10.23919/DATE.2018.8342127.

[76] Romain Jacob, Licong Zhang, Marco Zimmerling, Jan Beutel, Samarjit Chakraborty, and Lothar Thiele. The Time-Triggered Wireless Architecture. In *Euromicro Conference on Real-Time Systems (ECRTS)*, volume 165, pages 19:1–19:25, Dagstuhl, Germany, 2020. doi:10.4230/LIPIcs.ECRTS.2020.19.

[77] Peng Jiang. A New Method for Node Fault Detection in Wireless Sensor Networks. *Sensors*, 9(2):1282–1294, 2009. doi:10.3390/s90201282.

[78] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized Rumor Spreading. In *Symposium on Foundations of Computer Science (FOCS)*, pages 565–574, Redondo Beach, California, USA, 2000. doi:10.1109/SFCS.2000.892324.

[79] Jens Karschau, Marco Zimmerling, and Benjamin M. Friedrich. Renormalization group theory for percolation in time-varying networks. *Scientific Reports*, 8 (8011):1–8, 2018. doi:10.1038/s41598-018-25363-2.

[80] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. XORs in the Air: Practical Wireless Network Coding. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 243–254, Pisa, Italy, 2006. doi:10.1145/1159913.1159942.

[81] Sachin Katti, Shyamnath Gollakota, and Dina Katabi. Embracing Wireless Interference: Analog Network Coding. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 397–408, Kyoto, Japan, 2007. doi:10.1145/1282380.1282425.

[82] Tai-hoon Kim, Carlos Ramos, and Sabah Mohammed. Smart City and IoT. *Future Generation Computer Systems*, 76:159–162, 2017. doi:10.1016/j.future.2017.03.034.

[83] Andrzej Kochut, Arunchandar Vasan, A. Udaya Shankar, and Ashok Agrawala. Sniffing out the correct Physical Layer Capture model in 802.11b. In *IEEE International Conference on Network Protocols (ICNP)*, pages 252–261, Berlin, Germany, 2004. doi:10.1109/ICNP.2004.1348115.

[84] Ralf Koetter and Muriel Médard. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, 2003. doi:10.1109/TNET.2003.818197.

[85] Linghe Kong and Xue Liu. mZig: Enabling Multi-Packet Reception in ZigBee. In *International Conference on Mobile Computing and Networking (MobiCom)*, pages 552–565, Paris, France, 2015. doi:10.1145/2789168.2790104.

[86] Linghe Kong, Xi Chen, Xue Liu, Qiao Xiang, Yi Gao, Noam Ben Baruch, and Guihai Chen. AdaSharing: Adaptive Data Sharing in Collaborative Robots. *IEEE Transactions on Industrial Electronics*, 64(12):9569–9579, 2017. doi:10.1109/TIE.2017.2708000.

[87] Dimitrios Koutsonikolas, Y. Charlie Hu, and Chih-Chun Wang. Pacifier: High-Throughput, Reliable Multicast Without "Crying Babies" in Wireless Mesh Networks. *IEEE/ACM Transactions on Networking*, 20(5):1375–1388, 2012. doi:10.1109/TNET.2011.2177274.

[88] A. Ajith Kumar S., Knut Øvsthus, and Lars Michael Kristensen. An Industrial Perspective on Wireless Sensor Networks – A Survey of Requirements, Protocols, and Challenges. *IEEE Communications Surveys & Tutorials*, 16(3):1391–1412, 2014. doi:10.1109/SURV.2014.012114.00058.

[89] Branislav Kusy, Prabal Dutta, Philip Levis, Miklós Maróti, Ákos Lédeczi, and David Culler. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services. *International Journal of Ad Hoc and Ubiquitous Computing*, 1(4):239–251, 2006. doi:10.1504/IJAHUC.2006.010505.

[90] Yann Labit, Dimitri Peaucelle, and Didier Henrion. SEDUMI INTERFACE 1.02: A tool for solving LMI problems with SEDUMI. In *IEEE International Symposium on Computer Aided Control System Design (CACSD)*, pages 272–277, Glasgow, UK, 2002. doi:10.1109/CACSD.2002.1036966.

[91] Olaf Landsiedel, Federico Ferrari, and Marco Zimmerling. Chaos: Versatile and Efficient All-to-All Data Sharing and in-Network Processing at Scale. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1:1–1:14, Roma, Italy, 2013. doi:10.1145/2517351.2517358.

[92] J. Nicholas Laneman, David N. C. Tse, and Gregory W. Wornell. Cooperative Diversity in Wireless Networks: Efficient Protocols and Outage Behavior. *IEEE Transactions on Information Theory*, 50(12):3062–3080, 2004. doi:10.1109/TIT.2004.838089.

[93] Jeongkeun Lee, Wonho Kim, Sung-Ju Lee, Daehyung Jo, Jiho Ryu, Taekyoung Kwon, and Yanghee Choi. An Experimental Study on the Capture Effect in 802.11a Networks. In *ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WinTECH)*, pages 19–26, Montreal, Quebec, Canada, 2007. doi:10.1145/1287767.1287772.

[94] Krijn Leentvaar and Jan H. Flint. The Capture Effect in FM Receivers. *IEEE Transactions on Communications*, 24(5):531–539, 1976. doi:10.1109/TCOM.1976.1093327.

[95] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. PulseSync: An Efficient and Scalable Clock Synchronization Protocol. *IEEE/ACM Transactions on Networking*, 23(3):717–727, 2015. doi:10.1109/TNET.2014.2309805.

[96] 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)*, Vienna, Austria, 2016. doi:10.1109/ICCPS.2016.7479131.

[97] Shuo-Yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear Network Coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003. doi:10.1109/TIT.2002.807285.

[98] Yao Li, Emina Soljanin, and Predrag Spasojevic. Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding. *IEEE Transactions on Information Theory*, 57(2):1111–1123, 2011. doi:10.1109/TIT.2010.2095111.

[99] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. In *International Conference on Information Processing in Sensor Networks (IPSN)*, pages 153–166, Philadelphia, Pennsylvania, USA, 2013. doi:10.1145/2461381.2461402.

[100] Roman Lim, Balz Maag, and Lothar Thiele. Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 149–158, Graz, Austria, 2016. URL `https://dl.acm.org/doi/10.5555/2893711.2893732`.

[101] 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(5):1013–1024, 2016. doi:10.1109/JPROC.2015.2497161.

[102] Jiakang Lu and Kamin Whitehouse. Flash Flooding: Exploiting the Capture Effect for Rapid Flooding in Wireless Sensor Networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 2491–2499, Rio de Janeiro, Brazil, 2009. doi:10.1109/INFCOM.2009.5062177.

[103] Michael Luby. LT Codes. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 271–282, Vancouver, British Columbia, Canada, 2002. doi:10.1109/SFCS.2002.1181950.

[104] Rogelio Luck and Asok Ray. An Observer-based Compensator for Distributed Delays. *Automatica*, 26(5):903–908, 1990. doi:10.1016/0005-1098(90)90007-5.

[105] Jan Lunze. *Feedback Control of Large-Scale Systems*. Prentice Hall International Series in Systems and Control Engineering. Prentice Hall, 1992. ISBN 013318353X. URL `https://books.google.de/books?id=U91SAAAAMAAJ`.

[106] Jan Lunze. Synchronization of Heterogeneous Agents. *IEEE Transactions on Automatic Control*, 57(11):2885–2890, 2012. doi:10.1109/TAC.2012.2191332.

[107] Jan Lunze, Frank Allgöwer, Mathias Bürger, Ozan Demir, Uwe Helmke, Anna von Heusinger, and René Schuh. Multi-agent Systems. In *Control Theory of Digitally Networked Dynamic Systems*, pages 263–324. Springer International Publishing, 2014. doi:10.1007/978-3-319-01131-8_6.

[108] Michele Luvisotto, Zhibo Pang, and Dacfey Dzung. Ultra High Performance Wireless Control for Critical Applications: Challenges and Directions. *IEEE Transactions on Industrial Informatics*, 13(3):1448–1459, 2017. doi:10.1109/TII.2016.2617459.

[109] Jerome Peter Lynch, Yang Wang, R. Andrew Swartz, Kung-Chun Lu, and Chin-Hsiung Loh. Implementation of a closed-loop structural control system using wireless sensor networks. *Structural Control and Health Monitoring*, 15(4):518–539, 2008. doi:10.1002/stc.214.

[110] Yehan Ma and Chenyang Lu. Efficient Holistic Control over Industrial Wireless Sensor-Actuator Networks. In *IEEE International Conference on Industrial Internet (ICII)*, pages 89–98, Seattle, Washington, USA, 2018. doi:10.1109/ICII.2018.00018.

[111] 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):3:1–3:25, 2018. doi:10.1145/3185510.

[112] Yehan Ma, Chenyang Lu, and Yebin Wang. Efficient Holistic Control: Self-awareness across Controllers and Wireless Networks. *ACM Transactions on Cyber-Physical Systems*, 4(4):41:1–41:27, 2020. doi:10.1145/3371500.

[113] Fabian Mager, Johannes Neumann, Carsten Herrmann, Marco Zimmerling, and Frank Fitzek. All-to-All Communication in Multi-Hop Wireless Networks with Mixer: Poster Abstract. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 360–361, Stanford, California, USA, 2016. doi:10.1145/2994551.2996706.

[114] Fabian Mager, Carsten Herrmann, and Marco Zimmerling. One for All, All for One: Toward Efficient Many-to-Many Broadcast in Dynamic Wireless Networks. In *ACM Workshop on Hot Topics in Wireless (HotWireless)*, pages 19–23, Snowbird, Utah, USA, 2017. doi:10.1145/3127882.3127884.

[115] Prasanta Chandra Mahalanobis. On the Generalised Distance in Statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49–55, 1936. doi:10.1007/s13171-019-00164-5.

[116] Mohammad Hossein Mamduhi, Domagoj Tolić, and Sandra Hirche. Robust Event-based Data Scheduling for Resource Constrained Networked Control Systems. In *American Control Conference (ACC)*, pages 4695–4701, Chicago, Illinois, USA, 2015. doi:10.1109/ACC.2015.7172069.

[117] Mohammad Hossein Mamduhi, Adam Molin, Domagoj Tolić, and Sandra Hirche. Error-Dependent Data Scheduling in Resource-Aware Multi-Loop Networked Control Systems. *Automatica*, 81:209–216, 2017. doi:10.1016/j.automatica.2017.03.005.

[118] Justin Manweiler, Naveen Santhapuri, Souvik Sen, Romit Roy Choudhury, Srihari Nelakuditi, and Kamesh Munagala. Order Matters: Transmission Reordering in Wireless Networks. In *International Conference on Mobile Computing and Networking (MobiCom)*, pages 61–72, Beijing, China, 2009. doi:10.1145/1614320.1614328.

[119] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The Flooding Time Synchronization Protocol. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 39–49, Baltimore, Maryland, USA, 2004. doi:10.1145/1031495.1031501.

[120] José Mario Mastrangelo, Dominik Baumann, and Sebastian Trimpe. Predictive Triggering for Distributed Control of Resource Constrained Multi-agent Systems. *IFAC-PapersOnLine*, 52(20):79–84, 2019. doi:10.1016/j.ifacol.2019.12.130.

[121] Manuel Mazo Jr., Adolfo Anta, and Paulo Tabuada. An ISS self-triggered implementation of linear controllers. *Automatica*, 46(8):1310–1314, 2010. doi:10.1016/j.automatica.2010.05.009.

[122] Greg McMillan. Is Wireless Process Control Ready for Prime Time?, 2009. URL https://www.controlglobal.com/home/article/11381703/is-wireless-process-control-ready-for-prime-time. Accessed: 2022-12-13.

[123] Sean P. Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Springer Science & Business Media, 2012. ISBN 9781447132677. URL https://books.google.de/books?id=LlTlBwAAQBAJ&dq.

[124] Marek Miskowicz. *Event-Based Control and Signal Processing*. CRC Press, 2018. ISBN 9781482256567. URL https://books.google.de/books?id=b_QYCwAAQBAJ&dq.

[125] Venkata Prashant Modekurthy and Abusayeed Saifullah. Online Period Selection for Wireless Control Systems. In *IEEE International Conference on Industrial Internet (ICII)*, pages 170–179, Orlando, Florida, USA, 2019. doi:10.1109/ICII.2019.00042.

[126] Venkata Prashant Modekurthy, Abusayeed Saifullah, and Sanjay Madria. DistributedHART: A Distributed Real-Time Scheduling System for WirelessHART Networks. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 216–227, Montreal, Quebec, Canada, 2019. doi:10.1109/RTAS.2019.00026.

[127] Mobashir Mohammad and Mun Choon Chan. Codecast: Supporting Data Driven In-network Processing for Low-power Wireless Sensor Networks. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 72–83, Porto, Portugal, 2018. doi:10.1109/IPSN.2018.00014.

[128] Adam Molin and Sandra Hirche. Optimal Design of Decentralized Event-triggered Controllers for Large-scale Systems with Contention-based Communication. In *IEEE Conference on Decision and Control (CDC)*, pages 4710–4716, Orlando, Florida, USA, 2011. doi:10.1109/CDC.2011.6161215.

[129] Damon Mosk-Aoyama and Devavrat Shah. Information Dissemination via Network Coding. In *IEEE International Symposium on Information Theory (ISIT)*, pages 1748–1752, Seattle, Washington, USA, 2006. doi:10.1109/ISIT.2006.261654.

[130] Luca Mottola and Gian Pietro Picco. Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art. *ACM Computing Surveys*, 43(3): 19:1–19:51, 2011. doi:10.1145/1922649.1922656.

[131] Bobak Nazer and Michael Gastpar. Compute-and-Forward: Harnessing Interference Through Structured Codes. *IEEE Transactions on Information Theory*, 57 (10):6463–6486, 2011. doi:10.1109/TIT.2011.2165816.

[132] RCR Wirelesss News. Amazon's ambitions will push delivery drones' battery lives to the limit and maybe beyond, 2016. URL http://www.rcrwireless.com/20160802/europe/five-challenges-drones-tag28. Accessed: 2022-12-13.

[133] Fredrik Österlind and Adam Dunkels. Approaching the Maximum 802.15.4 Multi-hop Throughput. In *Workshop on Embedded Networked Sensors (HotEmNets)*, Charlottesville, Virginia, USA, 2008. URL https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1042468&d.

[134] Pouya Ostovari, Jie Wu, and Abdallah Khreishah. *Network Coding Techniques for Wireless and Sensor Networks*, pages 129–162. Springer Berlin Heidelberg, 2014. ISBN 9783642400094. doi:10.1007/978-3-642-40009-4_5.

[135] Pangun Park, Sinem Coleri Ergen, Carlo Fischione, Chenyang Lu, and Karl Henrik Johansson. Wireless Network Design for Control Systems: A Survey. *IEEE Communications Surveys & Tutorials*, 20(2):978–1013, 2018. doi:10.1109/COMST.2017.2780114.

[136] Nicholas J. Ploplys, Paul A. Kawka, and Andrew G. Alleyne. Closed-Loop Control over Wireless Networks. *IEEE Control Systems Magazine*, 24(3):58–71, 2004. doi:10.1109/MCS.2004.1299533.

[137] Valentin Poirot, Beshr Al Nahas, and Olaf Landsiedel. Paxos Made Wireless: Consensus in the Air. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 1–12, Beijing, China, 2019. URL https://dl.acm.org/doi/10.5555/3324320.3324322.

[138] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling Ultra-Low Power Wireless Research. In *International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 364–369, Boise, Idaho, USA, 2005. doi:10.1109/IPSN.2005.1440950.

[139] Michal Prauzek, Jaromir Konecny, Monika Borova, Karolina Janosova, Jakub Hlavica, and Petr Musilek. Energy Harvesting Sources, Storage Devices and System Topologies for Environmental Wireless Sensor Networks: A Review. *Sensors*, 18(8):1–22, 2018. doi:10.3390/s18082446.

[140] James A. Preiss, Wolfgang Hönig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A Large Nano-Quadcopter Swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304, Singapore, 2017. doi:10.1109/ICRA.2017.7989376.

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

[142] Daniel E. Quevedo, Anders Ahlén, Alex S. Leong, and Subhrakanti Dey. On Kalman filtering over fading wireless channels with controlled transmission powers. *Automatica*, 48(7):1306–1316, 2012. doi:10.1016/j.automatica.2012.03.025.

[143] Hariharan Rahul, Haitham Hassanieh, and Dina Katabi. SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 171–182, New Delhi, India, 2010. doi:10.1145/1851182.1851204.

[144] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Anthony Stankovic. Cyber-Physical Systems: The Next Computing Revolution. In *Design Automation Conference (DAC)*, pages 731–736, Anaheim, California, 2010. doi:10.1145/1837274.1837461.

[145] Chithrupa Ramesh, Henrik Sandberg, and Karl Henrik Johansson. Performance Analysis of a Network of Event-Based Systems. *IEEE Transactions on Automatic Control*, 61(11):3568–3573, 2016. doi:10.1109/TAC.2016.2523422.

[146] Federico Ramponi, Debasish Chatterjee, Andreas Milias-Argeitis, Peter Hokayem, and John Lygeros. Attaining Mean Square Boundedness of a Marginally Stable Stochastic Linear System With a Bounded Control Input. *IEEE Transactions on Automatic Control*, 55(10):2414–2418, 2010. doi:10.1109/TAC.2010.2054850.

[147] Mohsin Raza, Nauman Aslam, Hoa Le-Minh, Sajjad Hussain, Yue Cao, and Noor Muhammad Khan. A Critical Analysis of Research Potential, Challenges, and Future Directives in Industrial Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials*, 20(1):39–95, 2018. doi:10.1109/COMST.2017.2759725.

[148] Matt Rich and Nicola Elia. Optimal mean-square performance for MIMO networked systems. In *American Control Conference (ACC)*, pages 6040–6045, Chicago, Illinois, USA, 2015. doi:10.1109/ACC.2015.7172288.

[149] Coen Roest. Enabling the Chaos Networking Primitive on Bluetooth LE. Master's thesis, TU Delft, 2015. URL `http://resolver.tudelft.nl/uuid:95f50b13-6af5-4bb3-83ef-84b065e13682`.

[150] 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)*, pages 334–345, Rome, Italy, 2014. doi:10.1109/RTSS.2014.35.

[151] Carlos Santos, Manuel Mazo Jr., and Felipe Espinosa. Adaptive self-triggered control of a remotely operated P3-DX robot: Simulation and experimentation. *Robotics and Autonomous Systems*, 62(6):847–854, 2014. doi:10.1016/j.robot.2014.01.010.

[152] Carlos Santos, Felipe Espinosa, Enrique Santiso, and Manuel Mazo Jr. Aperiodic Linear Networked Control Considering Variable Channel Delays: Application to Robots Coordination. *Sensors*, 15(6):12454–12473, 2015. doi:10.3390/s150612454.

[153] Luca Schenato and Federico Fiorentin. Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, 47 (9):1878–1886, 2011. doi:10.1016/j.automatica.2011.06.012.

[154] Luca Schenato and Giovanni Gamba. A distributed consensus protocol for clock synchronization in wireless sensor network. In *IEEE Conference on Decision and Control (CDC)*, pages 2289–2294, New Orleans, Louisiana, USA, 2007. doi:10.1109/CDC.2007.4434671.

[155] 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)*, pages 1–8, Paris, France, 2017. doi:10.23919/WIOPT.2017.7959925.

[156] Fred B. Schneider. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990. doi:10.1145/98163.98167.

[157] Markus Schuß, Carlo Alberto Boano, Manuel Weber, and Kay Römer. A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 54–65, Uppsala, Sweden, 2017. URL `https://dl.acm.org/doi/10.5555/3108009.3108018`.

[158] Fanrong Shi, Xianguo Tuo, Lili Ran, Zhenwen Ren, and Simon X. Yang. Fast Convergence Time Synchronization in Wireless Sensor Networks Based on Average Consensus. *IEEE Transactions on Industrial Informatics*, 16(2):1120–1129, 2020. doi:10.1109/TII.2019.2936518.

[159] Joris Sijs, Benjamin Noack, and Uwe D. Hanebeck. Event-based State Estimation with Negative Information. In *International Conference on Information Fusion (FUSION)*, pages 2192–2199, Istanbul, Turkey, 2013. URL `https://ieeexplore.ieee.org/abstract/document/6641279`.

[160] 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):1453–1464, 2004. doi:10.1109/TAC.2004.834121.

[161] Francesco Smarra, Alessandro D'Innocenzo, and Maria Domenica Di Benedetto. Optimal co-design of control , scheduling and routing in multi-hop control networks. In *IEEE Conference on Decision and Control (CDC)*, pages 1960–1965, Maui, Hawaii, USA, 2012. doi:10.1109/CDC.2012.6426249.

[162] Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. The β-Factor: Measuring Wireless Link Burstiness. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 29–42, Raleigh, North Carolina, USA, 2008. doi:10.1145/1460412.1460416.

[163] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. An Empirical Study of Low-Power Wireless. *ACM Transactions on Sensor Networks*, 6(2): 16:1–16:49, 2010. doi:10.1145/1689239.1689246.

[164] John Anthony Stankovic, Tarek Farouk Abdelzaher, Chenyang Lu, Lui Sha, and Jennifer C. Hou. Real-Time Communication and Coordination in Embedded Sensor Networks. *Proceedings of the IEEE*, 91(7):1002–1022, 2003. doi:10.1109/JPROC.2003.814620.

[165] Aleksandar Stanoev, Adnan Aijaz, Anthony Portelli, and Michael Baddeley. Demo: Closed-Loop Control over Wireless - Remotely Balancing an Inverted Pendulum on Wheels. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 195–197, Lyon, France, 2020. URL `https://dl.acm.org/doi/10.5555/3400306.3400342`.

[166] 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)*, pages 267–280, Seoul, South Korea, 2015. doi:10.1145/2809695.2809706.

[167] Shuo Tian, Wenbo Yang, Jehane Michael Le Grange, Peng Wang, Wei Huang, and Zhewei Ye. Smart healthcare: making medical care more intelligent. *Global Health Journal*, 3(3):62–65, 2019. doi:10.1016/j.glohj.2019.07.001.

[168] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A Macroscope in the Redwoods. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 51–63, San Diego, California, USA, 2005. doi:10.1145/1098918.1098925.

[169] Sebastian Trimpe. Predictive and Self Triggering for Event-based State Estimation. In *IEEE Conference on Decision and Control (CDC)*, pages 3098–3105, Las Vegas, Nevada, USA, 2016. doi:10.1109/CDC.2016.7798733.

[170] Sebastian Trimpe and Dominik Baumann. Resource-Aware IoT Control: Saving Communication Through Predictive Triggering. *IEEE Internet of Things Journal*, 6(3):5013–5028, 2019. doi:10.1109/JIOT.2019.2894628.

[171] 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):48–75, 2012. doi:10.1109/MCS.2012.2214135.

[172] Matteo Trobinger, Gabriel de Albuquerque Gleizer, Timofei Istomin, Manuel Mazo Jr., Amy L. Murphy, and Gian Pietro Picco. The Wireless Control Bus: Enabling Efficient Multi-Hop Event-Triggered Control with Concurrent Transmissions. *ACM Transactions on Cyber-Physical Systems*, 6(1):4:1–4:29, 2021. doi:10.1145/3485467.

[173] Roman Trüb, Reto Da Forno, Lukas Sigrist, Lorin Mühlebach, Andreas Biri, Jan Beutel, and Lothar Thiele. FlockLab 2: Multi-Modal Testing and Validation for Wireless IoT. In *Workshop on Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench)*, pages 1–7, London, UK, 2020. doi:10.3929/ethz-b-000442038.

[174] Manel Velasco, Josep M. Fuertes, and Pau Martí. The Self Triggered Task Model for Real-Time Control Systems. In *Work-in-Progress Session of the IEEE Real-Time Systems Symposium (RTSS)*, pages 67–70, Cancun, Mexico, 2003. URL `https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.164.3868`.

[175] Gregory C. Walsh, Hong Ye, and Linda G. Bushnell. Stability Analysis of Networked Control Systems. *IEEE Transactions on Control Systems Technology*, 10 (3):438–446, 2002. doi:10.1109/87.998034.

[176] Achim Walter, Robert Finger, Robert Huber, and Nina Buchmann. Smart farming is key to developing sustainable agriculture. *Proceedings of the National Academy of Sciences*, 114(24):6148–6150, 2017. doi:10.1073/pnas.1707462114.

[177] Shiyong Wang, Jiafu Wan, Di Li, and Chunhua Zhang. Implementing Smart Factory of Industrie 4.0: An Outlook. *International Journal of Distributed Sensor Networks*, 12(1):1–10, 2016. doi:10.1155/2016/3159805.

[178] Xiaofeng Wang and Michael D. Lemmon. Self-Triggered Feedback Control Systems With Finite-Gain $\mathcal{L}_2$ Stability. *IEEE Transactions on Automatic Control*, 54(3):452–467, 2009. doi:10.1109/TAC.2009.2012973.

[179] 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):1025–1038, 2016. doi:10.1109/JPROC.2015.2509186.

[180] Geoffrey Werner-Allen, Geetika Tewari, Ankit Patel, Matt Welsh, and Radhika Nagpal. Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 142–153, San Diego, California, USA, 2005. doi:10.1145/1098918.1098934.

[181] Kamin Whitehouse, Alec Woo, Fred Jiang, Joseph Polastre, and David Culler. Exploiting The Capture Effect For Collision Detection And Recovery. In *IEEE Workshop on Embedded Networked Sensors (EmNetS)*, pages 45–52, Sydney, New South Wales, Australia, 2005. doi:10.1109/EMNETS.2005.1469098.

[182] Matthias Wilhelm, Vincent Lenders, and Jens B. Schmitt. On the Reception of Concurrent Transmissions in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 13(12):6756–6767, 2014. doi:10.1109/TWC.2014.2349896.

[183] Björn Wittenmark, Johan Nilsson, and Martin Törngren. Timing Problems in Real-time Control Systems. In *American Control Conference (ACC)*, pages 2000–2004, Seattle, Washington, USA, 1995. doi:10.1109/ACC.1995.531240.

[184] Martin Wollschlaeger, Thilo Sauter, and Juergen Jasperneite. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017. doi:10.1109/MIE.2017.2649104.

[185] Junlin Xiong and James Lam. Stabilization of linear systems over networks with bounded packet loss. *Automatica*, 43(1):80–87, 2007. doi:10.1016/j.automatica.2006.07.017.

[186] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and Modelling of the Temporal Dependence in Packet Loss. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 345–352, New York, New York, USA, 1999. doi:10.1109/INFCOM.1999.749301.

[187] Tai C. Yang, Hongnian Yu, M. R. Fei, and L. X. Li. Networked control systems: a historical review and current research topics. *Measurement & Control*, 38(1): 12–16, 2005. doi:10.1177/002029400503800101.

[188] Hong Ye, Gregory C. Walsh, and Linda G. Bushnell. Real-Time Mixed-Traffic Wireless Networks. *IEEE Transactions on Industrial Electronics*, 48(5):883–890, 2001. doi:10.1109/41.954551.

[189] Dingwen Yuan and Matthias Hollick. Let's Talk Together: Understanding Concurrent Transmission in Wireless Sensor Networks. In *IEEE Conference on Local Computer Networks (LCN)*, pages 219–227, Sydney, New South Wales, Australia, 2013. doi:10.1109/LCN.2013.6761237.

[190] Pouria Zand, Supriyo Chatterjea, Kallol Das, and Paul Havinga. Wireless Industrial Monitoring and Control Networks: The Journey So Far and the Road Ahead. *Journal of Sensor and Actuator Networks*, 1(2):123–152, 2012. doi:10.3390/jsan1020123.

[191] Lixian Zhang, Huijun Gao, and Okyay Kaynak. Network-Induced Constraints in Networked Control Systems - A Survey. *IEEE Transactions on Industrial Informatics*, 9(1):403–416, 2013. doi:10.1109/TII.2012.2219540.

[192] Tianyu Zhang, Tao Gong, Song Han, Qingxu Deng, and Xiaobo Sharon Hu. Fully Distributed Packet Scheduling Framework for Handling Disturbances in Lossy Real-Time Wireless Networks. *IEEE Transactions on Mobile Computing*, 20(2): 502–518, 2021. doi:10.1109/TMC.2019.2950913.

[193] Wei Zhang, Michael S. Branicky, and Stephen M. Phillips. Stability of Networked Control Systems. *IEEE Control Systems Magazine*, 21(1):84–99, 2001. doi:10.1109/37.898794.

[194] 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 Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 546–555, San Francisco, California, USA, 2013. doi:10.1109/MASCOTS.2013.76.

[195] 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):8:1–8:29, 2017. doi:10.1145/3012005.

[196] Marco Zimmerling, Luca Mottola, and Silvia Santini. Synchronous Transmissions in Low-Power Wireless: A Survey of Communication Protocols and Network Services. *ACM Computing Surveys*, 53(6):121:1–121:39, 2021. doi:10.1145/3410159.

List of Publications

The following list contains all publications of the author, categorized by conference, journal, poster, demo, and video publications.

Conference publications

Fabian Mager[⋆], Carsten Herrmann[⋆], and Marco Zimmerling. "One for All, All for One: Toward Efficient Many-to-Many Broadcast in Dynamic Wireless Networks." In *ACM Workshop on Hot Topics in Wireless (HotWireless)*, pages 19–23, Snowbird, Utah, USA, 2017. doi:10.1145/3127882.3127884.

⋆ Both authors contributed equally to this work.

Carsten Herrmann[⋆], Fabian Mager[⋆], and Marco Zimmerling. "Mixer: Efficient Many-to-All Broadcast in Dynamic Wireless Mesh Networks." In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 145–158, Shenzhen, China, 2018. doi:10.1145/3274783.3274849.

⋆ Both authors contributed equally to this work.

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, Porto, Portugal, 2018. doi:10.1109/CPSBench.2018.00009.

⋆ Both authors contributed equally to this work.

Fabian Mager[⋆], Dominik Baumann[⋆], Romain Jacob, Lothar Thiele, Sebastian Trimpe, and Marco Zimmerling. "Feedback Control Goes Wireless: Guaranteed Stability over Low-power Multi-hop Networks." In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 97–108, Montreal, Quebec, Canada, 2019. doi:10.1145/3302509.3311046. **(Best paper award)**

⋆ Both authors contributed equally to this work.

Fabian Mager, Andreas Biri, Lothar Thiele, and Marco Zimmerling. "Butler: Increasing the Availability of Low-Power Wireless Communication Protocols." In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 108–119, Linz, Austria, 2022. https://dl.acm.org/doi/10.5555/3578948.3578958.

Andreas Biri, Reto Da Forno, Tobias Kuonen, Fabian Mager, Marco Zimmerling, and Lothar Thiele. "Hydra: Concurrent Coordination for Fault-tolerant Networking." In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 219–232, San Antonio, Texas, USA, 2023. doi:10.1145/3583120.3587047

Journal publications

Dominik Baumann*, Fabian Mager*, Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. "Fast Feedback Control over Multi-hop Wireless Networks with Mode Changes and Stability Guarantees." *ACM Transactions on Cyber-Physical Systems*, 4(2):18:1–18:32, 2019. doi:10.1145/3361846.

* Both authors contributed equally to this work.

Dominik Baumann*, Fabian Mager*, Marco Zimmerling, and Sebastian Trimpe. "Control-Guided Communication: Efficient Resource Arbitration and Allocation in Multi-Hop Wireless Control Systems." *IEEE Control Systems Letters*, 4(1):127–132, 2020. doi:10.1109/LCSYS.2019.2922188.

* Both authors contributed equally to this work.

Dominik Baumann*, Fabian Mager*, Ulf Wetzker, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. "Wireless Control for Smart Manufacturing: Recent Approaches and Open Challenges." *Proceedings of the IEEE*, 109(4):441–467, 2021. doi:10.1109/JPROC.2020.3032633.

* Both authors contributed equally to this work.

Fabian Mager*, Dominik Baumann*, Carsten Herrmann, Sebastian Trimpe, and Marco Zimmerling. "Scaling Beyond Bandwidth Limitations: Wireless Control With Stability Guarantees Under Overload." *ACM Transactions on Cyber-Physical Systems*, 6(3):20:1–20:30, 2022. doi:10.1145/3502299.

* Both authors contributed equally to this work.

Posters

Fabian Mager, Johannes Neumann, Carsten Herrmann, Marco Zimmerling, and Frank Fitzek. "All-to-All Communication in Multi-Hop Wireless Networks with Mixer: Poster Abstract." In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, pages 360–361, Stanford, California, USA, 2016. doi:10.1145/2994551.2996706.

Fabian Mager*, Dominik Baumann*, Sebastian Trimpe, and Marco Zimmerling. "Poster Abstract: Toward Fast Closed-Loop Control over Multi-Hop Low-Power Wireless Networks." In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 158–159, Porto, Portugal, 2018. doi:10.1109/IPSN.2018.00042.

* Both authors contributed equally to this work.

Demos

Fabian Mager*, Dominik Baumann*, Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. "Demo Abstract: Fast Feedback Control and Coordination with Mode Changes for Wireless Cyber-Physical Systems." In *International Conference on Information Processing in Sensor Networks (IPSN)*, pages 340–341, Montreal, Quebec, Canada, 2019. doi:10.1145/3302506.3312483. **(Best demo award)**

* Both authors contributed equally to this work.

Fabian Mager, Romain Jacob, Reto Da Forno, and Marco Zimmerling. "Competition: Low-Power Wireless Bus Baseline." In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 292–293, Beijing, China, 2019. https://dl.acm.org/doi/10.5555/3324320.3324386.

Videos

Wireless Control for Cyber-physical Systems:
https://www.youtube.com/watch?v=Fr2YNLPaUpY

Remote Stabilization, Mobility Experiment:
https://www.youtube.com/watch?v=19xPHjnobkY

Feedback Control Goes Wireless, Demonstration at CPS-IoT Week 2019:
https://www.youtube.com/watch?v=AtULmfGkVCE

Feedback Control goes Wireless, Presentation at ICCPS 2019:
https://www.youtube.com/watch?v=1i6oBsat_Ww