HARDWARE-SOFTWARE CO-DESIGN FOR RELIABLE MULTI-HOP NETWORK CONTROL

Arnab Mondal

Hardware-Software Co-design for Reliable Multi-hop Network Control

Thesis submitted to

Indian Institute of Technology Kharagpur

For the award of the degree

of

Master of Science

by

Arnab Mondal
(Roll No: 17AT72P01)

Under the guidance of

Prof. Soumyajit Dey

Dept. of Computer Science & Engineering

Prof. Alok Kanti Deb

Dept. of Electrical Engineering

Advanced Technology Development Centre
Indian Institute of Technology Kharagpur
September 2021

CERTIFICATE

This is to certify that the thesis entitled **HARDWARE-SOFTWARE CO-DESIGN FOR RELIABLE MULTI-HOP NETWORK CONTROL**, submitted by **Arnab Mondal** to Indian Institute of Technology Kharagpur, is a record of bona fide research work under our supervision and is worthy of consideration for the award of the degree of Master of Science of the Institute.

Prof. Soumyajit Dey
Supervisor

Prof. Alok Kanti Deb
Joint Supervisor

DECLARATION

I, Arnab Mondal, Roll No. 17AT72P01, registered as a student in the Advanced Technology Development Centre, Indian Institute of Technology, Kharagpur, India (herein after referred to as the 'Institute') do hereby submit my project report, titled: HARDWARE-SOFTWARE CO-DESIGN FOR RELIABLE MULTI-HOP NETWORK CONTROL (herein after referred to as 'my thesis') in a printed as well as in an electronic version for holding in the library of record of the Institute.

I hereby declare that:

1. The electronic version of my thesis submitted herewith on CDROM is in PDF Format.

2. My thesis is my original work of which the copyright vests in me and my thesis does not infringe or violate the rights of anyone else.

3. The contents of the electronic version of my thesis submitted herewith are the same as that submitted as final hard copy of my thesis after my viva voce and adjudication of my thesis in August/September 2021.

4. I agree to abide by the terms and conditions of the Institute Policy and Intellectual Property (herein after Policy) currently in effect, as approved by the competent authority of the Institute.

5. I agree to allow the Institute to make available the abstract of my thesis in both hard copy (printed) and electronic form.

6. For the Institute's own, non commercial, academic use I grant to the Institute the non-exclusive license to make limited copies of my thesis in whole or in part and to loan such copies at the Institute's discretion to academic persons and bodies approved of from time to time by the Institute for non-commercial academic use. All usage under this clause will be governed by the relevant fair use provisions in the Policy and by the Indian Copyright Act in force at the time of submission of the thesis.

7. Furthermore,

   (a) I agree to allow the Institute to place such copies of the electronic version of my thesis on the private Intranet maintained by the Institute for its own academic community.

(b) I agree to allow the Institute to publish such copies of the electronic version of my thesis on a public access website of the Internet should it so desire.

8. That in keeping with the said Policy of the Institute I agree to assign to the Institute (or its Designee/s) according to the following categories all rights in inventions, discoveries or rights of patent and/or similar property rights derived from my thesis where my thesis has been completed.

   (a) With use of Institute-supported resources as defined by the Policy and revisions thereof,

   (b) With support, in part or whole, from a sponsored project or program, vide clause 6(m) of the Policy. I further recognize that:

   (c) All rights in intellectual property described in my thesis where my work does not qualify under sub-clauses 8(a) and/or 8(b) remain with me.

9. The Institute will evaluate my thesis under clause 6(b1) of the Policy. If intellectual property described in my thesis qualifies under clause 6(b1) (ii) as Institute-owned intellectual property, the Institute will proceed for commercialization of the property under clause 6(b4) of the policy. I agree to maintain confidentiality as per clause 6(b4) of the Policy.

10. If the Institute does not wish to file a patent based on my thesis, and it is my opinion that my thesis describes patentable intellectual property to which I wish to restrict access, I agree to notify the Institute to that effect. In such a case no part of my thesis may be disclosed by the Institute to any person(s) without my written authorization for one year after the date of submission of the thesis or the period necessary for sealing the patent, whichever is earlier.

Arnab Mondal

Acknowledgements

Abstract

Please write the abstract that I have edited in the google doc.

The topic of Cyber-Physical Systems (CPS) is extremely interdisciplinary and covers a wide assortment of utilizations. Numerous CPS educational and research plans intend to show different aspects of how designing control algorithms and embedded systems impact each other. Building an experimental platform to outline this interdependency is however non-trivial. A purely theoretical exposition on this topic without any genuine experiments is also not effective. To address this, in the first work, a novel programmable testbed is proposed that models complex, networked CPS with minimal design effort. As the heart of this testbed, a software tool is proposed that provides a simple-to-utilize interface for indicating various CPS configurations and implementing them in hardware.

While wired networks provide a reliable platform for networked cyber-physical systems (CPS), there is an expanding interest for CPS built upon wireless networks. However, wireless connectivity also implies varying and unpredictable end-to-end delays due to packet loss, interference by concurrently transmitting nodes or the necessity to forward packets via one or many intermediate nodes. In the second work of the thesis, for the first time, a generic technique is presented to handle varying end-to-end delays in wireless CPS. While maintaining a stable operation, our technique preserves a high control performance. Our proposed strategy is independent of the controller design technique and the communication protocol used. We also present a real-world implementation of our proposed technique on a physical testbed. Experiments suggest that the proposed strategy improves the control performance of the system by up to 63 % compared to existing control schemes.

Since wireless communication is prone to failures, e.g., by colliding packets, existing real-time wireless CPS schemes prioritize network reliability attained through packet broadcast and re-transmissions at the cost of energy efficiency. This becomes more challenging due to the limited battery life of wireless infrastructures. The third work in this thesis uniquely addresses this issue and focuses on achieving high reliability adhering to improved control performance with minimal energy consumption. As a solution, an energy-aware hierarchical control-theoretic approach has been proposed, which adaptively switches controllers and network schedules at the high level, while adaptively regulating the number of retransmissions of packets and associated energy consumption profiles at the low level. An experimental evaluation on a real-world testbed shows an improvement in reliability and control performance up to 53 % and 24 % respectively, while reducing the energy consumption up by 43 % compared to state-of-the-art techniques.

In summary, the thesis reports a generic technique to handle varying end-to-end delays, a novel adaptive control methodology to maintain reliability, control performance, and energy efficiency in wireless CPS, and the development of a novel programmable open architecture testbed for validating such CPS scenarios.

Keywords: Cyber-Physical Systems, Wireless CPS, Education, Research, Testbed, MCN, Multi-hop Wireless Networks, Control Performance, Variable Delay, Feedback, Reliability, Energy.

Notations and Abbreviations

CPS	Cyber-Physical System(s)
LQR	Linear Quadratic Regulator
LTI	Linear and Time-Invariant
$\mathcal{G}$	Network Graph
$V_{\mathtt{P}}$	The Set of Plant Nodes
$V_{\mathtt{I}}$	The Set of Intermediate Nodes
$\mathcal{C}$	The Control Node i.e. Network Manager
$\|x\|$	Euclidean Norm of variable x if any other norm is not mentioned
$x[k]$ or $x(k)$	Variable x at k-th sampling period
K	Controller Gain Matrix
F	Feedforward Gain Matrix
x or X	System State Vector
$\widehat{x}$	Predicted State
u	Control Input Vector
δ	Delay in network
k'	actuation Instant
h	Sampling Period
WC	Worst-Case delay based control scheme
FSFD	Fixed-Sampling and Fixed-Delay based control scheme
FSVD	Fixed-Sampling and Variable-Delay based control scheme
ATM	daptively-Controlled Time-Triggered Multi-Hop Wireless CPS
ACM	Adaptive Channel Mapping
$\mathcal{A}_v$	Activity Information
Ω	Network Schedule
R_k	Reliability Value
HCA	Hierarchical Control Architecture
LCS	Low-level Control Scheme
HCS	High-level Control Scheme

Contents

List of Figures

List of Tables

Chapter 1
Introduction

The proliferation of Cyber-Physical Systems (CPS) has been a matter of increased convenience in modern safety-critical systems engineering. The underlying architecture of most CPS implementations is essentially a Networked Control System (NCS) with complex software implementations for signal processing, on-board prognostics, health-monitoring and other related performance, safety and intelligence features. The design and development of intelligent control algorithms for CPS implementations need to take into account platform effects like timing delays, sensor data uncertainties, packet drops over a network, etc. In that way, a primary function of the CPS practitioner becomes creating a correct by construction translation of mathematical control laws to actual implementations while accounting for the aforementioned uncertainties. Moreover, such implementations often require to implement platform-level resource sharing. For example, in the context of Industrial Control Systems (ICS), multiple control loops often share a communication infrastructure [21, 50].

Multi-hop network-based control of physical systems is becoming increasingly common in the modern world owing to the ability of low-cost sensing, fast transfer of data, and low-power implementation of control commands [40, 70]. Several modern applications like robotic swarm coordination and motion control which would earlier be limited in range and functionality due to hardware and networking limitations in terms of computation and communication are now possible in this new era of edge devices with computing and transmit capabilities. As discussed in [39], the ability to close control loops fast enough is about to bring in a revolution in several domains like smart manufacturing, transportation, tactical networks for long-range drone control etc.

However, such an ambition for fast monitoring and control in a cyber-physical system (CPS) requires support for wireless communication infrastructure with reliability and timing guarantee so that necessary real-time properties of the system are maintained. Such properties, like the minimum required rate of control actuation in terms of packet delivery sequence, are key to the safety, performance and

stability of such systems. However, another issue in this regard is that example systems in this domain often require multi-hop communication over wireless links due to power consumption, transmission range constraints, and fault-tolerance requirements. With these in consideration, we are faced with a multi-objective requirement where control loops need to be closed fast enough, i.e. a set of physical systems need to communicate their state information over a multi-hop network to a shared central controller where actuation commands are computed and dispatched back for the physical system actuation. The entire round trip communication is required to be performed inside time intervals in the range of 10-200 ms depending on the fast/slow dynamics of the physical systems [39, 44].

1.1. Background and Motivations

Here I explain the inadequacies of prior works in this field, as well as the motivations I derive from them. The rationale for the unique techniques speak for themselves in terms of the solutions we chose to answer the difficulties in this thesis.

1.1.1. Scarcity of Proper Experimental ...m for CPS Research tion

Before going to these follwoing subs

In Cyber-Physical Systems (CPS), the behaviour of the entire system is defined by the complex interactions between control algorithms, the hardware platform, and the communication network connecting the different devices. In particular, CPS in which controller and plant are connected through a multi-hop wireless network are deployed frequently, since they offer a higher degree of flexibility, lower maintenance and installation cost, and better adaptability than conventional CPS [4, 36]. While a lot of research has been done on wired as well as wireless CPS, there are no low-cost and configurable platforms available that facilitate education and research of distributed CPS systems in general and wireless CPS in particular.

Understanding and developing CPS requires interdisciplinary knowledge from Computer Science, Mathematics, Electrical Engineering, Mechanical Engineering and other domains. Different design choices affect different layers of abstraction, and the complex interactions among multiple layers are often difficult to understand. Providing a good understanding of such cross-layer effects is of fundamental importance in CPS education and training [65], especially in the light of safety-critical CPS.

However, establishing a deeper understanding of cross-layer interactions that occur in real-world CPS cannot be achieved without performing experiments on feature-rich testbeds [66], which expose a future practitioner to the non-ideal char-

acteristics of CPS implementation platforms and their effects on performance and safety. However, many of the available testbeds consist of simplistic setups, which typically comprise only a simple plant and a single microcontroller. Here, the behaviour of the testbed does not differ significantly from what is predicted by the theory, because non-idealities occurring in real-world setups are not present. As a result, they are only of limited use for conveying the necessary insights into practical CPS. CPS users

A networked CPS consists of multiple nodes with different roles (e.g., plant, controller, intermediate node, router, etc.), which all require their custom-developed firmware. Moreover, these nodes have to be interconnected through a wireless network, which requires further design effort. As a result, developing such a setup from scratch remains a very complex and time-consuming task. Despite suitable textbook material on embedded and cyber-physical systems [46], in many courses and institutions, training on practical CPS is eluded. This leads to a gap between the growing importance of CPS on the one hand and appropriate training on the other hand. While CPS design methods have evolved as structured, software-centric approaches to connect and control physical systems; the absence of real-life, affordable testbeds often become a serious impediment to the design, development, and validation of associated control and scheduling algorithms.

1.1.2. Variable Delay in Event-triggered Multi-hop Wireless Networks

Rename as:Issue of Variable Delay in M

In recent years, control over wireless networks is becoming increasingly common due to requirements for low-cost sensing, flexibility, and low-power implementations [5,40]. In this new era of edge devices with computing and transmit capabilities, applications in mission/safety-critical domains (e.g., robotic swarm coordination and motion control) can now have extended ranges and functionalities that were not possible earlier due to hardware and network limitations. However, such systems might also have higher performance requirements, e.g., faster stabilization or lower settling time. As discussed in [39], high-performance feedback control, if realized through wireless networks, has the potential to revolutionize several domains like e.g., smart manufacturing, transportation or tactical networks for long-range drone control.

In a wired network, packets typically arrive reliably with small and predictable delays. In contrast, in a wireless network, packets might collide with packets from other devices and therefore might get lost. They hence need to be re-transmitted in such cases, which leads to variable and unpredictable delays. In addition, in multi-hop networks, intermediate nodes need to forward data from one node to the other. Here, the delay also depends on the route. In addition, if packets

are lost in a multi-hop network, the delay varies to an even higher extent. For example, Wireless HART networks [47, 64], which are being used frequently in industrial process automation applications, subdivide time into different slots. Every transmission takes one slot length, i.e., 10 ms. If a packet needs to be relayed among multiple hops, the delay will be a multiple of this slot length, since a node can only forward one packet per slot. Hence, the delay also varies with the number of hops and therefore the route taken through the network.

To get an impression on the variability of delays, let us consider a simple CPS that consists of a plant, controller, and a transmission node for intermediate communication. The control loop is formed by a forward (i.e., sensor-to-controller) and return (i.e., controller-to-actuator) path. We assume that each transmission between two nodes incurs a delay of 10 ms, and that the 3 nodes are arranged in a daisy-chained fashion, such that there are 2 hops between controller and plant. Since an intermediate node receives data in one slot and then forwards them to the next node in the subsequent slot, the $2 + 2 = 4$ hops of one round-trip would incur an end-to-end delay of at least 40 ms. If now a certain fraction of transmission attempts fail, each re-transmission would cause an additional delay of 10 ms. A failure rate of 5 % would result into the distribution of delays depicted in Figure 1.1. Congestion is also the main source of delay in most other wireless networks, even in single-hop ones. For example, when the channel in an IEEE-802.11 network (WiFi) is busy, every station waits for a random, exponentially distributed amount of time before transmitting [16]. This will also lead to variable delays, especially when the network load is high.

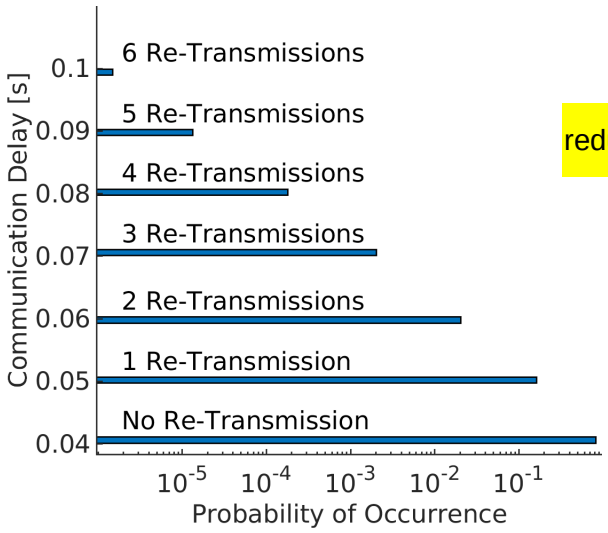

Figure 1.1: Delay distribution.

With the growing importance of wireless CPS, techniques to mitigate the effects of varying delays have been studied thoroughly in the literature. While it is known that a lower sampling period and a shorter sensing-to-actuation delay allows the design of controllers with a higher control performance [12, 54, 57], large and varying round-trip delays in the underlying wireless network negatively impact the control stability. Hence, previous works have mainly emphasized on designing a

stable controller in the presence of varying delays [23, 25, 44], without mitigating the corresponding performance degradation. Other works have used a Kalman filter to predict the control input when packets are delayed [62, 63]. On the other hand, certain works have tried to address the issue from the networking side by providing more resources, i.e., either adding a high-quality communication alternative [9, 52, 53] or redundantly transmitting each packet to the destination via multiple routes [44].

1.1.3. Reliability and Energy-aware Control of Time-triggered

Challenges in Developing Reliability and Energy-aware Multi-hop ...

Most existing works in this area consider *time-triggered* packet transmission using fixed, time-slotted static schedules. They can be classified into 1) *single-path approaches*, in which the network schedule defines a time-slotted point-to-point communication scheme across multiple nodes from a source to a sink [27, 64], and 2) *flooding-based approaches*. In the later case, whenever any node receives a packet, it repeats it a fixed number of times in the form of a broadcast to all other nodes in vicinity. All nodes are synchronized and transmit the repetitions concurrently [20, 73]. This results in a very low probability of a packet being lost, e.g., by collisions, and hence high reliability. However, such multiple parallel transmissions of a single packet results in an energy overhead making it inconvenient for most mobile, energy-constrained setups. On the other hand, the single-path approach is the most energy-efficient one, since the number of packet transmissions is minimized. This results in lower channel utilization, thus creating an opportunity for multiplexing communication of more nodes [27, 64]. However, the dependence on a single path makes this scheme more vulnerable to colliding packets [61], since every collision leads to a loss. Our work addresses this issue by adaptively adjusting control strategy and the number of retransmissions at runtime to trade-off reliability and energy consumption.

1.2. Our Objectives

In view of the motivations from the inadequacies of the cutting edge approaches here we sum up the essential objectives of our work beneath (and portray them in Fig. 1.2).

here no will ur

1. Development of a programmable testbed with open architecture for scheduling and analysis of multi-hop control networks which is useful for both research implementation and experimental demonstration in CPS education. The CPS testbed allows high-level specification of faults and associated communications schedules which can be used to generate code for this programmable testbed and run it as per the high-level specification.

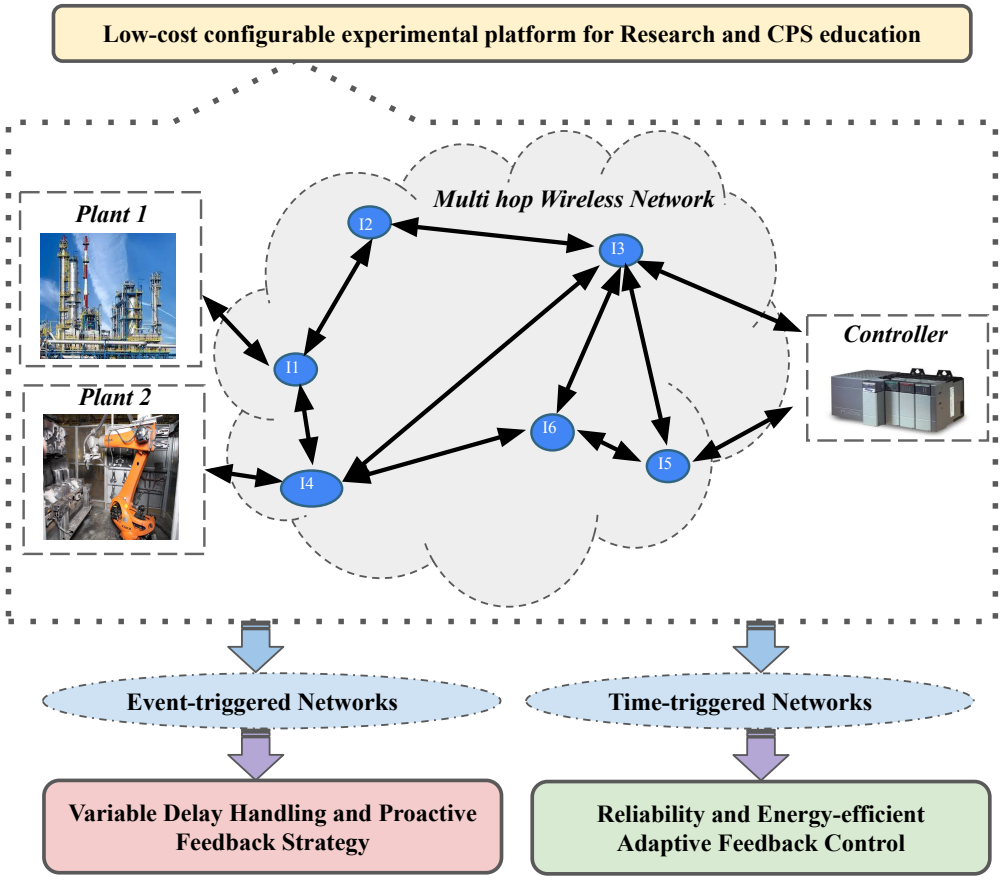

Figure 1.2: Overview of Objectives

2. Proposal of a generic technique that can be used in conjunction with any given wireless network and controller. In other words, we provide the "glue" to efficiently implement a given high-performance controller in a given network, in spite of variable delays. In particular, neither the controller needs to be designed using any knowledge about the underlying network and its delay distribution, nor does the network need to be adjusted based on the designed controller. Furthermore, the actuator does not need to execute any computationally expensive prediction algorithms, while at the same time any delay distribution and hence network protocol can be accounted for.
3. Development of a hierarchical control architecture for adaptively regulating the communication schedule of a multi-hop control network ensuring the stability and reliability of individual control loops with significant energy savings at the network level and validating the idea by implementing it on a coexisting setup of different networks.

1.3. Contributions of the Thesis

In this thesis, we plan to utilize hardware-software co-design techniques and generic proactive control methodologies in order to achieve above objectives, aim-

ing a robust and reliable design of wireless CPS. The exploration contributions of this thesis are summed up underneath.

1.3.1. A Low-Cost Programmable Open Architecture Testbed for CPS Research and Education

In this work we address this problem and present a novel programmable testbed for specifying, implementing, and analyzing various networked CPS configurations in a user-friendly way. The testbed provides a unique opportunity to design, run, and validate wireless control algorithms for different networked CPS configurations through hands-on experiments. The proposed testbed is complex enough to exhibit hardware effects like timing delays, packet drops, etc., which are also present in real-world setups. As a result, students and CPS practitioners can validate the robustness of their control algorithms against these non-idealities. Since our main goal is exhibiting this behavior for educational purposes, we have additionally included the option to insert programmable, artificial faults.

At the heart of our testbed, there is a software tool that supports a graphical user interface (GUI)-based easy-to-use front-end for specifying a given CPS configuration, and then implementing it on real hardware.Based on a CPS specification, which is given in terms of network parameters, control parameters and plant dynamics, our tool automatically generates and deploys the firmware of the individual nodes in the network. It thereby configures the nodes, as well as their network interfaces. Each such node is formed by a widely-used Arduino UNO microcontroller, which is equipped with a nRF24L01+ RF module [49]. Some of these nodes take over the role of a controller, whereas others are equipped with some physical setup to be controlled (the physical plant), or act as intermediate nodes that relay information. The firmware images are deployed over-the-air, which reduces the effort of testbed deployment. After these steps, a testbed that implements the specified CPS configuration and failure characteristics is ready to use. Further, during the experiments on the resulting CPS, every node records and stores data. After the experiment, this data is wirelessly transmitted to the PC, on which our proposed tool orchestrates data collection, aggregation, and visualization. The data can be *replayed* for analyzing events of interest (e.g., specification violations, unstable behavior, etc.). The salient features of our testbed are summarized as follows.

1. The testbed supports specifying and analyzing a CPS configuration in an easy-to-use manner using a novel GUI-driven software tool with minimal design effort.
2. Artificial faults can be injected by specifying the probability distributions for packet loss and other errors.

3. Experimental data and execution statistics of each node can be retrieved with zero effort. Based on them, the platform's and controller's behavior can be evaluated.

The testbed thus targets educational and teaching purposes. The extremely low design effort greatly reduces the development cost and time. The testbed allows studying various CPS concepts by carrying out different experiments through an easy-to-use interface. The required hardware consists solely of widely-available off-the-shelf components (e.g., Arduino), thereby making the testbed accessible and low-cost.

1.3.2. Proactive Feedback for Networked CPS

In this work, we propose a generic proactive feedback strategy to run a high-performance controller reliably in spite of large delay variations in wireless networks. Let the controller be designed with a certain sampling period h that is chosen by considering ideal network operations with minimal delay. Whenever this controller is triggered, it proactively computes different control inputs considering different possible round trip delays, i.e., $h, 2 \times h, \cdots, k \times h$. These control inputs are then transmitted to the actuator together in a single packet. Now, if a packet arrives on time at the actuator, the new control input, which has been computed for the delay of h, is applied. Otherwise, if a packet is delayed, the control input from the previously received packet that corresponds to the actual delay is applied. Thus, the proposed control strategy allows to proactively compensate for the delay that might occur due to the communication over the wireless network. This proactive delay compensation preserves the high control performance, e.g., low settling times. The strategy we propose here can be used with any control design techniques, including event- and self-triggered control.

Compared to the existing works, we make the following contributions:

- We, for the first time, propose a generic technique to increase the control performance in wireless CPS by proactively pre-computing and transmitting different control inputs for different possible delays.

- We propose a proactive feedback strategy that exploits the knowledge of delay variations of the network to pre-calculate control inputs for future actuation time instants and sends them to the actuator. These inputs can then be applied to the plant when the delay is large and the new inputs have not arrived in time.

- We implement our proposed strategy on a real-world CPS testbed. Using real-world experiments on this testbed, we show that a significantly higher control performance (i.e., up to 63 % w.r.t. existing approaches) can be

obtained for a wireless CPS using our method, when large and variable end-to-end delays are present.

1.3.3. AEW: Adaptively-Controlled Energy-Efficient Wireless CPS

In this work, we for the first time propose a generic framework called *AEW* (Adaptively-Controlled Energy-Efficient Wireless CPS) as a low power alternative to flooding-based techniques for reliable wireless control of fast feedback systems. It provides an additional layer of *reliability management* on the top of a single path communication protocol. Compared to flooding-based approaches, it provides significant energy savings at the network level. At the same time, a hierarchical control architecture adaptively switches through different control modes and regulates number of retransmissions for guaranteeing the desired reliability and performance of all the control loops. The high level control scheme (HCS) of AEW allows dynamically switching between multiple pre-synthesized control modes. Each such control mode is represented by a *control-schedule pair* consisting of n controllers and network schedules for n control loops. The network schedules are synthesized based on the hard deadlines of those loops. In addition, a low level control scheme (LCS) tunes the number of retransmission attempts in the network to balance reliability and energy consumption. Consecutive retransmissions are carried out on different wireless channels and hence, with a growing number of retransmissions, the collision probability declines. This retransmission management is done by employing a dedicated PID control routine on each node. Our methodology thereby converts the control performance requirement of individual control loops to node level transmission reliability targets, which serve as references for the underlying energy-aware PID control routines.

1.4. Thesis Organisation

The thesis is coordinated into five parts and following are their outlines:

1. **Chapter 1:** This is an introductory part, that gives a summed up outline of how the best in class work and practices in wireless CPS space inspire our work in this thesis, and what are the significant contributions of this thesis.
2. **Chapter 2:** This is a chapter that provides a detailed overview of the development of the programmable testbed with open architecture for CPS research and education.
3. **Chapter 3:** In this chapter, we describe the generic proactive feedback strategy, that we have proposed in order to handle end-to-end variable delays in wireless CPS.

4. **Chapter 4:** We describe the adaptive energy-efficient control architecture that we develop as a low-power alternative to flooding-based techniques in order to ensure provable guarantees of control performance of Cyber-Physical Systems (CPS) interacting over wireless networks.
5. **Chapter 5:** In this chapter, we sum up our research contributions and conclude the dissertation with some possible future explorations of this research.

Chapter 2
A Low-Cost Programmable Open Architecture Testbed for CPS Research and Education

Modern cyber-physical systems (CPS) are increasingly embracing low-power embedded devices and wireless multi-hop communication to facilitate monitoring and control of physical systems at unprecedented flexibility and low cost in safety critical systems engineering. While CPS design methods have evolved as structured, software-centric approaches to connect and control physical systems; the absence of real-life, affordable testbeds often become a serious impediment to design, development, and validation of associated control and scheduling algorithms. This work reports the development of such an affordable CPS testbed in CPS education which is useful for performance evaluation of such algorithms against well-known notions of CPS faults like packet drops, transmission errors, fault attacks, etc. We propose a programmable hardware-software co-designed system which may be used for characterizing the behavior of control algorithms and their shared Networked Control System (NCS) implementation in terms of uncertainties possible in the network infrastructure.

2.1. CPS Testbed for Education

Fig. 2.1 shows the overview of our proposed testbed. A set of n physical plants are controlled by n controllers running in parallel on a dedicated node, called the *control node*. Intermediate nodes connect plants and the control node by realizing a muti-hop wireless network. Each plant can be a real-world physical system or a Simulink model. Currently, our tool supports Double Integrator Circuits (DICs) as a physical plant. However, support for other physical plants can be added easily in the future. A DIC is an operational amplifier-based circuit that computes the integral over the integral of its input signal. We have chosen a DIC as our first supported plant, since it forms a standard control-theoretic benchmark that is

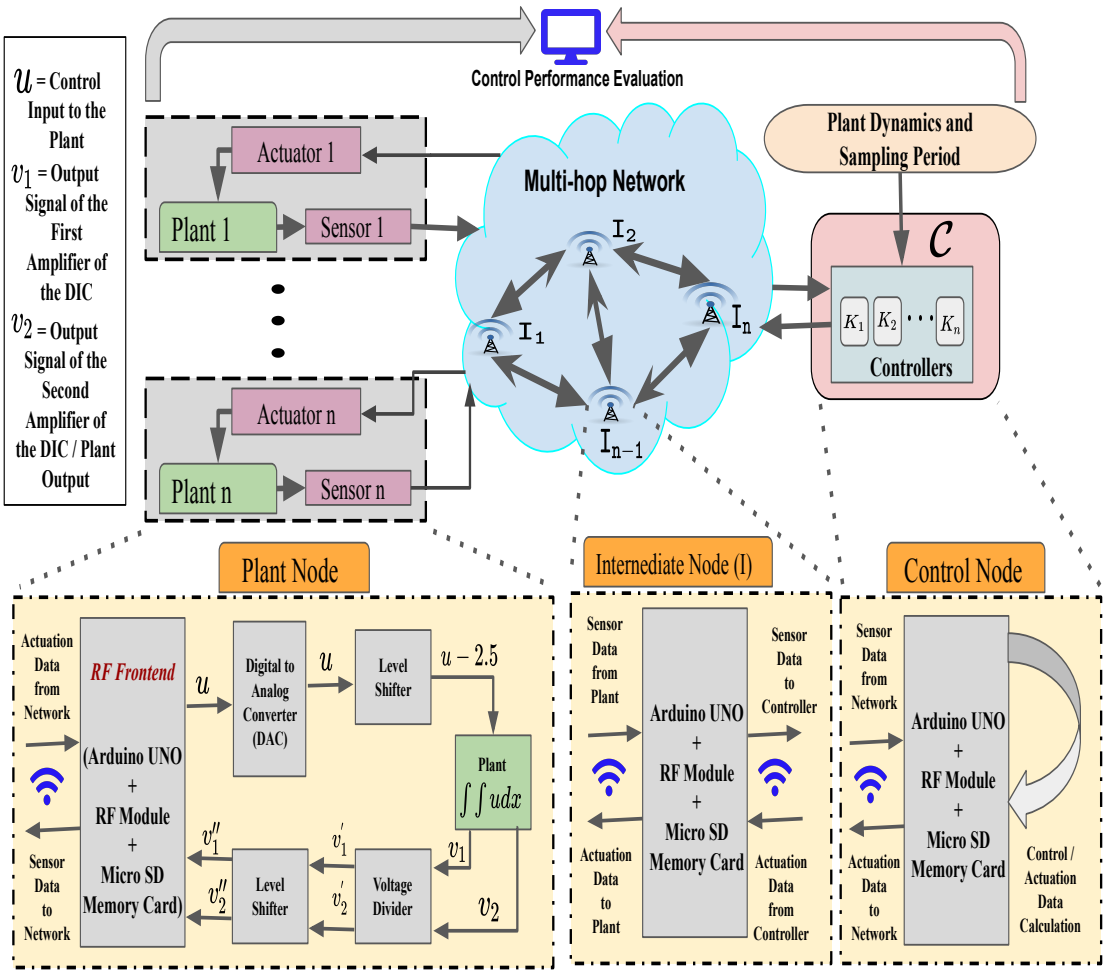

Figure 2.1: Overview of our proposed CPS-testbed.

widely used in control engineering.

As already mentioned, the proposed testbed targets educational purposes. We have therefore implemented the following five primary features that make it particularly suitable for CPS education and training.

Low-Cost Components: Many of the existing CPS testbeds target research applications [71]. Here, the cost of the components is not a key aspect, since often, a single setup is sufficient. In contrast, hands-on educational formats are more cost-sensitive, since a larger number of course participants require multiple setups in parallel. Therefore, our testbed is built upon the off-the-shelf Arduino UNO microcontroller equipped with a nRF24L01+ RF module [49]. Besides the wide global availability of these components, they lead to a low cost of ≈ 30 USD per intermediate- or control node, and a cost of ≈ 45 USD per plant node. Overall, a CPS setup having one plant node, five intermediate nodes and a control node incurs ≈ 225 USD, which easily suits the typical budgets of most educational institutions.

Versatility: A testbed used for education and training should be versatile enough

to support a large number of different experiments, which will convey knowledge on various aspects of CPS. The proposed testbed can easily realize different CPS configurations by simply adjusting the following parameters and configuration options.

1. **Plant Parameters:** Number of plants and their types, given by their mathematical descriptions.
2. **Control Parameters:** Sampling periods and specifications of the different discrete controllers.
3. **Network Parameters:** Number of nodes, network topology, routing paths of all plant-control loops, number of parallel channels, number of retransmissions of packets by each node, etc.

Realistic Behavior: The behavior of a CPS testbed should be as close as possible to that of a real-world setup. In particular, it should be suitable for studying the robustness of a CPS design in the presence of non-ideal behavior. Hence, it is inevitable that a testbed exhibits non-idealities such as packet drops, network delays, etc. Our proposed testbed is complex enough to exhibit such non-ideal behavior. In addition, it comes with a programmable fault insertion methodology, using which users can insert artificial faults that correspond to the errors described above. Hence, the robustness of e.g., a given control algorithm, can be tested easily.

Easy-To-Use GUI: A feature that makes our testbed unique compared to existing ones is its easy-to-use GUI-based interface. Based on this GUI, the following tasks can be controlled. 1) Automatized CPS configuration, 2) generation of firmware for each node, 3) wireless deployment of the firmware onto each node, 4) real-time execution and orchestration of the CPS, and 5) data aggregation and visualization. In other words, a user can specify its configuration in a simple fashion, and our tool will generate a CPS testbed realizing these specifications on-the-fly. This allows for studying CPS testbed configurations that are actually different CPS, while using one physical setup. With our approach, manually integrating plants, controllers and the communication network is no longer required.

Easy Development and Scalability: The proposed testbed can be developed with minimalistic development effort and is highly scalable. Extra hardware can be added or existing devices can be upgraded easily, without requiring a time-consuming software redesign. For example, a small network can be easily transformed into a larger network by adding additional nodes. The only required step is the GUI-driven configuration, specification and automated regeneration of the software. Furthermore, different network topologies, e.g., single- and multi-hop, can be generated easily.

The proposed system has already demonstrated its practicality for teaching at universities. In particular, it has been used in the courses entitled 1. *Compu-*

tational Foundation of Cyber-Physical Systems, 2. *Introduction to Programming Intelligent Physical Systems* at the Indian Institute of Technology (IIT) Kharagpur in 2019. More than 50 students from several disciplines, including CS, EE, and others attended both the courses. Many of them did not have any prior background in control theory and/or networking. Based on their feedback, we can claim that the participants found the setup helpful for understanding different aspects of CPS, e.g., communication, distributed controllers, different design options and performance issues under non-ideal conditions.

2.2. Proposed CPS Testbed

In this section, we describe our proposed software tool and the corresponding hardware details of the testbed.

2.2.1. Tool for Automated CPS Generation and Operation

The progression of execution inside individual node is administered by a C++ script running on them, which involves different open source libraries created for the hardware onboard. The resulting program is therefore an assortment of standard C++ and hardware library specific functions which have adequate learning curve associated with them. In addition, to oblige our necessities, a few of such library functions were modified and added which are not supported by original developers. Such nitty-gritties of implementation severely make the utility of the test-bed to a more broad utilization by anyone apart from the developers. Moreover, for making the usefulness of the testbed more generalize, there is a need to foster system specific tools that would supplement the testbed by giving visual data overlay and appreciation to the client about the action in the network system, in this way abstracting execution explicit subtleties.

Hence, as the core of our proposed testbed, a software tool is developed, which runs on a PC that is wirelessly connected to the CPS setup. The tool is developed in Qt® framework to avail its cross platform compatibility under the GNU Lesser General Public License v.3 (LGPL). Binaries are successfully generated and tested on the Win64 (Windows 7 and 10) and Linux (Ubuntu 16.04) platforms with minimal modification to the source. Our software tool mainly fulfills the following two purposes.

Configuration and Firmware Generation

The GUI-based front-end is shown in Fig. 2.2. It is used for specifying the CPS that is to be realized. Among others, aspects that can be configured are the network topology, routing paths of the control loops, reference values of the plant outputs, and properties related to fault injection.

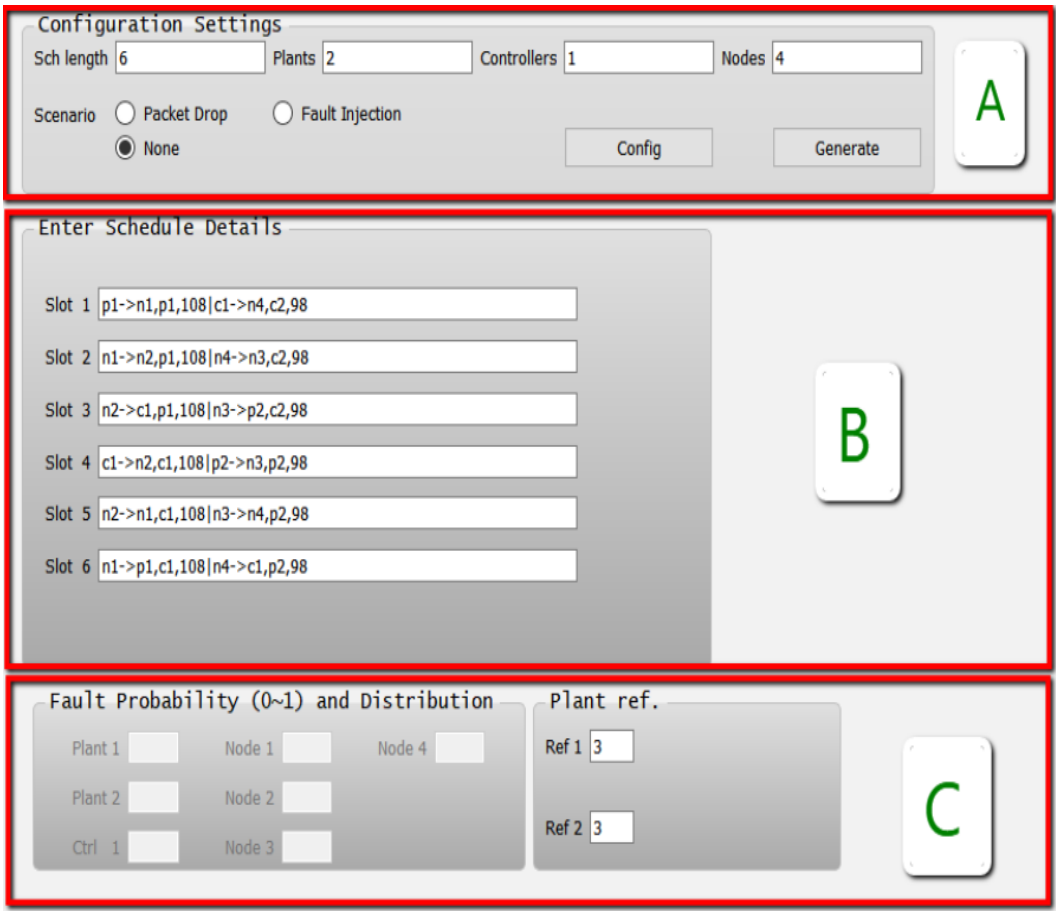

Figure 2.2: Configuration front-end of the GUI-driven software tool.

The upper part marked '*A*' of Fig. 2.2 shows the main configuration window, which can be used for specifying the length of routing paths, the number of nodes, and the injection of artificial faults. In the middle Part *B*, inputs for specifying the routes of the control loops are provided. More specifically, the wireless channel is subdivided into multiple time-slots, and each slot can be used for transmitting data between a certain pair of sender and receiver on a certain wireless channel. The user can specify the set of nodes that form a route and the corresponding set of slots used.

The lower Part *C* of the figure is used for specifying probabilities of simulated packet loss and for specifying the target values of plant outputs (i.e., reference values). Based on these inputs and more elaborate specifications from configuration files, the software-tool generates and wirelessly deploys a firmware image for each node.

Data Analysis

After an experiment of the CPS has been carried out, the data is transmitted wirelessly to the PC and can then be analyzed using our software tool. Our tool orchestrates data collection and aggregation and provides functionality to visualize

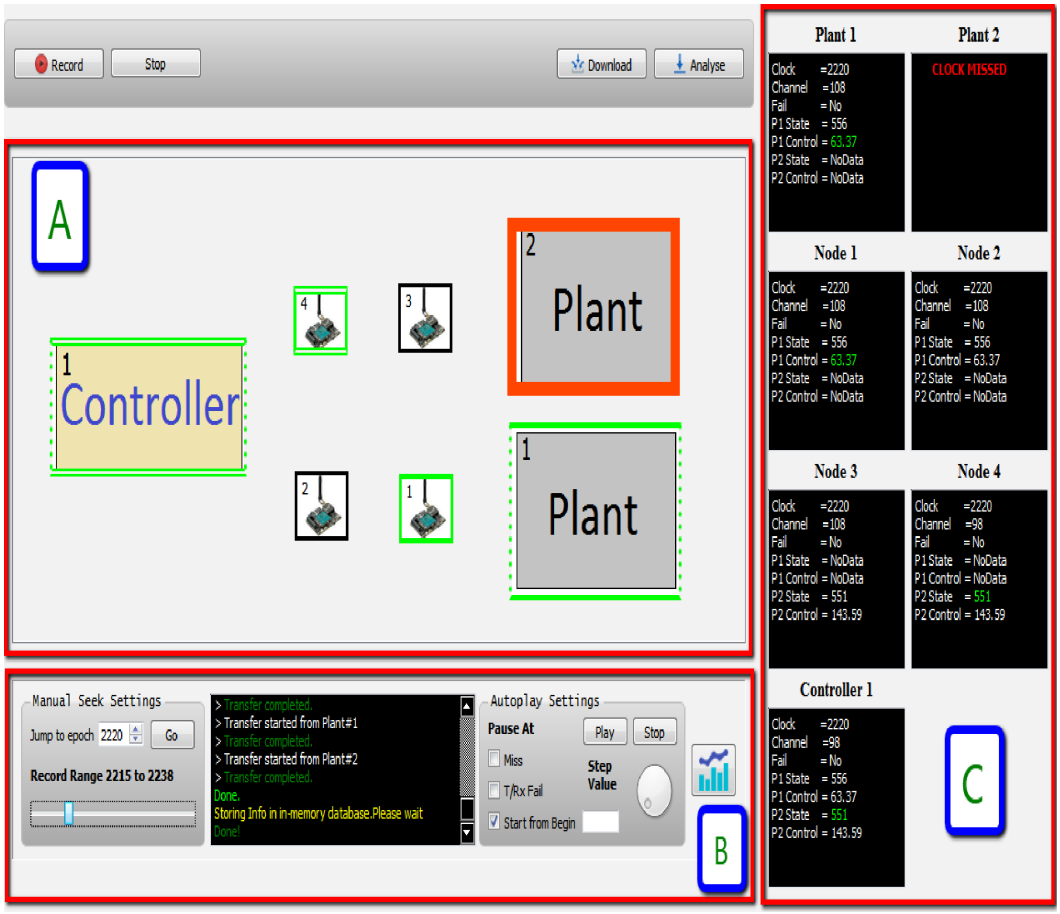

Figure 2.3: Analysis Front-end of the GUI-driven software tool.

and analyze the CPS behavior. The front-end for data analysis is depicted in Fig. 2.3. It offers the following functionality.

Record and Store Data: During an experiment, each node stores information on every network transaction on a SD memory card. After the experiment, this data is transmitted to the PC for analysis.

Visualize and Analyze: The upper Part *A* of Fig. 2.3 visualizes the CPS. Suitable icons represent plants, intermediate nodes and controllers. On request, the tool animates the actual CPS operation, based on the recorded data. As shown in Fig. 2.3, Plant 1 and Node 1 are highlighted using green color, indicating a successful transmission between them. In the depicted example, Plant 1 is in reception mode, which is annotated by the "dotted" line. Similarly, Node 4 successfully transmits to the controller, which is indicated by the green double-frame. In contrast, Plant 2 faces a transmission failure as indicated by the red color.

On the left of the lower Part *B* of Fig. 2.3, the user can select the point in time of the experiment to be examined. In the middle of Part *B*, a console depicts various messages. On the right of Part *B*, the *Autoplay Settings* allow for

automatically advancing the point in time under consideration. This enables a record and playback - style of operation. Since an experiment typically generates a large set of data, the *Autoplay Settings* also provide a discontinuous and fast mode of replay. Here, the playback is interrupted whenever certain events, such as packet loss or communication errors, occur. The right Part C of the figure shows the internal state of each node at the point in time under consideration. For each node, the current timestamp, the corresponding channel, the most recently received data from each plant, the controller output and the failure/success status of the last executed transmission are depicted.

2.2.2. Hardware Implementation

We next describe key aspects of the testbed hardware. The different hardware components forming the testbed are depicted in Fig. 3.4.

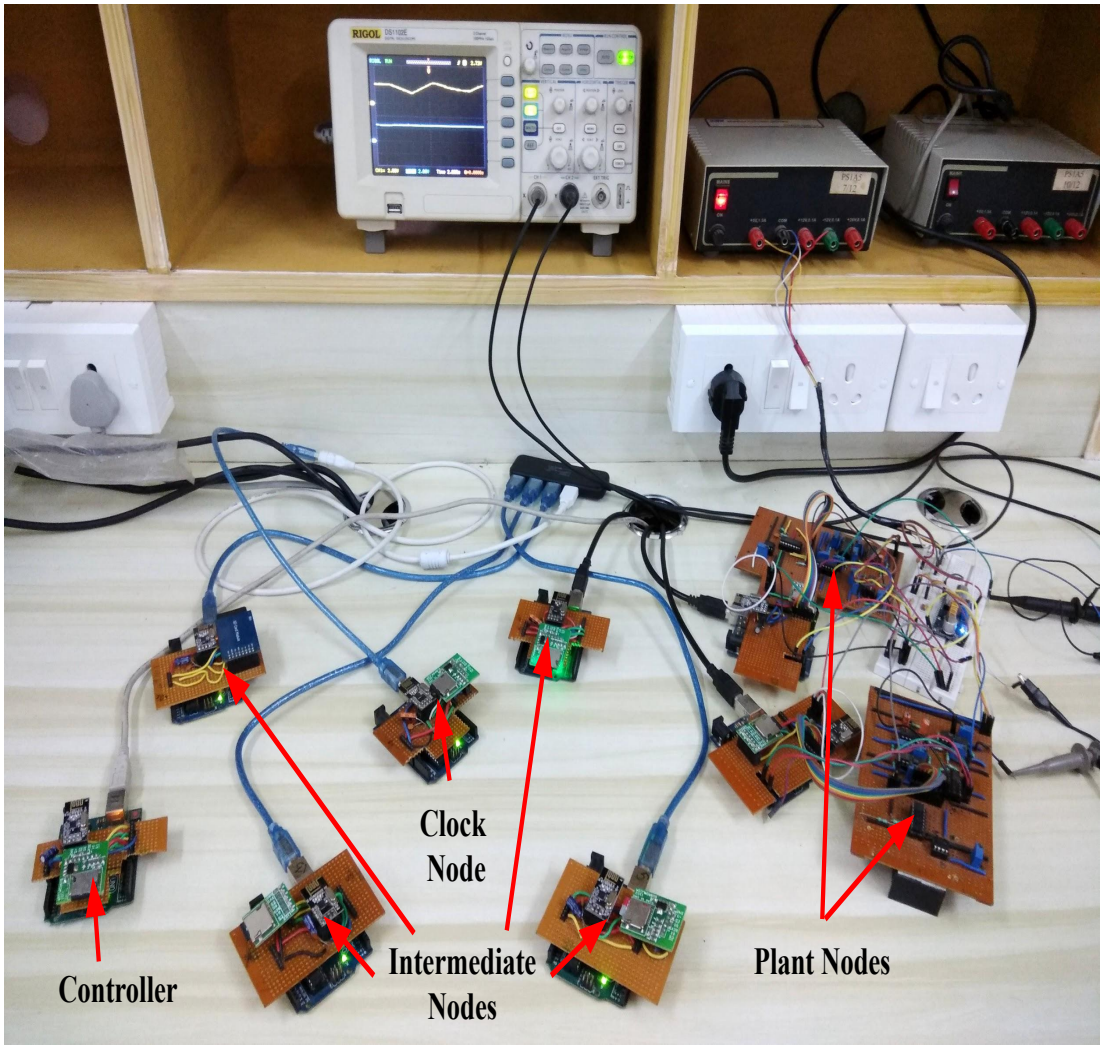

Figure 2.4: The CPS testbed consisting of a control node, four intermediate nodes and two physical plants.

Plant Node

As mentioned earlier, we consider a DIC as the first physical plant supported by our testbed. It consists of two operational amplifiers (UA741) connected in series, thereby computing the integral over the integral of the input signal of the first amplifier. The control objective is maintaining a given target voltage (i.e., the *reference value*) at the output of the second amplifier by adjusting the input voltage of the first amplifier. An overview of a plant node is depicted in the lower-left part of Fig. 2.1.

Control and Intermediate Nodes

Controller- and intermediate nodes consist of a microcontroller, RF interface and SD memory card. The microcontroller is entirely programmed using our software tool. On a control node, the microcontroller receives plant output, computes the control input using standard control design techniques such as pole placement [7], and then wirelessly transmits control input.

Global Clock Node

The clock node consists of the same hardware and provides a clock signal to which all nodes are synchronized. With a period equivalent to the slot length, a clock signal is broadcasted to all nodes. In addition, the clock node is also responsible for orchestrating the collection of experimental data over the network.

Communication Network

Given the GUI-based CPS specification, firmware for each node is generated and deployed by the software tool. This firmware is also responsible for controlling the connectivity. The wireless communication is built on top of the *Enhanced ShockBurst* protocol, which operates in the ISM band (2.400 - 2.4835 GHz). We use an over-the-air data rate of 250 kbit/s. As a congestion control mechanism, the NRF24L01P RF module implements up to 6 logical addresses (called *pipes*) for listening in parallel. A module can only transmit to one pipe at a time. In our setup, three pipes are maintained by each RF module for handling the global clock, plant/control and log data. Channel arbitration is done based on time-multiplexed media access. Time is subdivided into 10 ms-slots and the clock node transmits a synchronization pulse to all nodes every 10 ms. Upon receiving the clock pulse, each node first checks if there is any data to be processed. A node may initiate a transmission, whenever the current slot is assigned to this node for transmission. Similarly, a node will listen for incoming transmissions from other nodes during the appropriate slots. The clock node can also initiate data logging on a node, or request the transmission of the previously logged data.

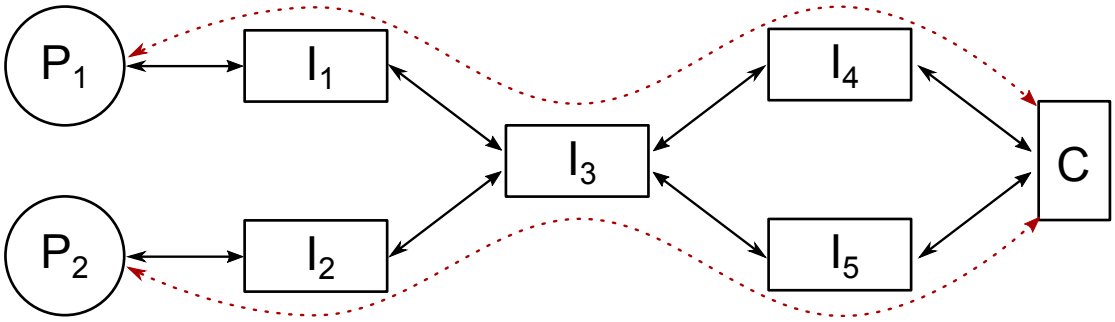

Figure 2.5: Network graphs for Experiments 2, 3, 4 and 5.

2.3. Demonstration of Teaching Experiments

In this section, we demonstrate how important curricular activities like CPS- and controller design, execution, performance evaluation, and refinement can be easily performed using our testbed. Five different teaching experiments, which can be used to study different aspects of CPS, are exemplified below. In each experiment, the testbed is executed for 15 s. We initially set the output voltage of both plants to 0 V, whereas the reference value is 3 V. Upon starting such an experiment, the controller will attempt to bring the output voltage to the reference value. The standard measure of quality of control in CPS is minimizing the *settling time*, which is the time until the 2% envelope around the reference value is reached.

Experiment 1 - Simple CPS

We demonstrate a simplistic CPS platform, in which a single-hop wireless network connects the two plants and the shared control platform. Here, each plant is sampled with a period of 20 ms and pole placement-based controllers are employed for both plants. Fig. 2.6 depicts the output signal of Plant 1, which is measured using an oscilloscope. As can be seen in Fig. 2.6, the system settles quickly within 3.303 s for Plant 1.

Experiment 2 - Multi-hop CPS

Here, we demonstrate a more involved CPS model, in which plants and controllers exchange messages through a multi-hop wireless network. Due to the limited receiver sensitivity, platform level data loss occurs (e.g., ∼1% in our example). We realize the topology of the wireless network depicted in Fig. 2.5, with two plants P_1 and P_2, and two controllers running on the shared control node C. The routing paths for the pair of control loops are depicted by the dotted lines between P_1 and C and between P_2 and C.

As mentioned in Sec. 2.2.1, in each slot, one transmission between one pair of nodes per channel is possible. Recall that transmissions occur with a slot length of 10 ms. For the control loop between P_1 and C, a round-trip delay of 80 ms

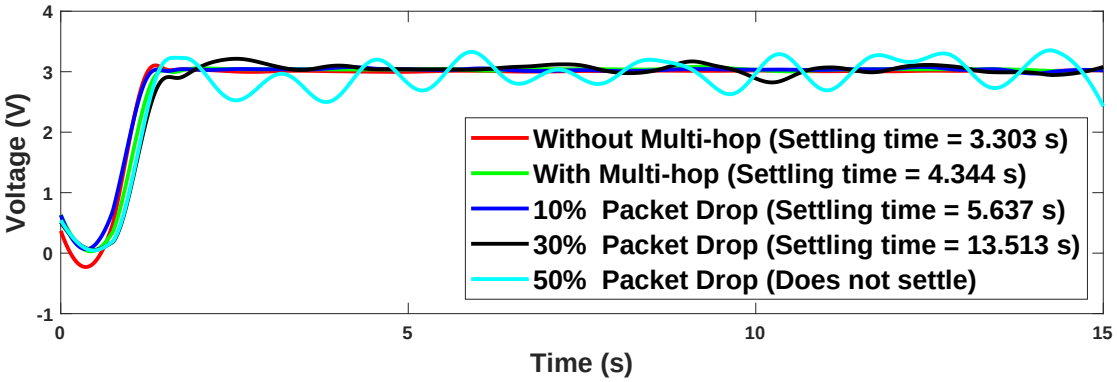

Figure 2.6: Output of Plant 1 for all Teaching Experiments.

is induced by the 2×4 hops. For the loop between P_2 and C, Node I_3 will be busy with relaying data for P_1 for two slot lengths, and hence, the round-trip delay for this loop becomes 100 ms. We therefore choose sampling periods of 80 ms and 100 ms for controlling P_1 and P_2, respectively, and apply pole-placement based controllers. As can be seen from Fig. 2.6, the settling time for Plant 1 has increased to 4.344 s as a result of the increased sampling period of 80 ms. The settling time of Plant 2, as reported in Table 2.1, exhibits a similar behavior.

Experiment 3 - Impact of Different Controllers

Experiment 2 makes clear that the multi-hop network degrades the control performance. In this experiment, we study how this degradation can be mitigated by applying different controllers. Whereas we have used a pole placement-based controller in the previous experiments, we switch to a controller based on the linear quadratic regulator [7] optimal control design technique for Plant 2. Table 2.1 clearly shows the improvement in settling time on applying this new controller.

Experiment 4 - Effect of Non-Ideal Multi-hop CPS

In order to explore more practical situations, we now consider packet drops during transmission. We therefore specify packet drop rates for Node I_3 (cf. Fig. 2.5), which are realized by regenerating the firmware of Node I_3 using our tool. Fig. 2.6 shows the output response of Plant 1 for different packet drop rates. As can be seen, such transmission loss significantly reduces the control performance. For Plant 2, similar effects in terms of increasing settling time and eventual unstable behavior are reported in Table 2.1.

Experiment 5 - Impact of Re-Transmission of Packets

The effect of packet drops can be reduced by tuning the number of re-transmissions. In particular, our tool can configure the number of re-transmission attempts to be automatically made by each node whenever its transmission has failed. Settling time values considering this refinement are given in Table 2.1. As

Table 2.1: Settling Time (in s) of Plant 2

No Loss		**Packet Loss**		
Single-Hop	Multi-Hop	10%	30%	50%
3.285	4.979	6.248	14.144	unstable
Performance Improvement by Changing the Controller				
2.896	4.162	5.229	13.059	unstable
Performance Tuning by Enabling Re-Transmission				
3.138	4.517	5.183	9.361	unstable

can be seen in Table 2.1, a larger number of re-transmissions improve the control performance.

2.4. Concluding Remarks

Each of the experiments provides opportunities for imbibing students with the salient features of CPS design. The first two experiments will help students to learn the effects of network parameters (e.g., routing path, delay) and control parameters (e.g., sampling period) on the control performance. Experiment 3 will teach how the degradation in system performance caused by network-delay can be mitigated by designing the controller appropriately. Experiment 4 will help students to understand the impact of environmental non-idealities and faults on the system behaviour. In Experiment 5, students will learn how the effect of such non-idealities (e.g., packet drops) can be mitigated by re-configuring some network parameters, e.g., tuning the number of re-transmissions. This experiment also helps students to understand how a vulnerable node in the network can cause performance degradation. Such directions can potentially motivate a student to undertake more involved work in the domain of fault-tolerant, distributed CPS design, fault-diagnosis, mitigation, etc.

Chapter 3
Proactive Feedback for Networked CPS

In networked cyber-physical systems (CPSs), a physical plant is controlled using software running on a processing unit that typically receives sensor information and sends control signals over a communication network. In such systems, the network timing plays a crucial role in determining the physical behavior of the plant. Thus, a controller that is designed obliviously to the network behavior might violate the control requirements [56]. It is particularly challenging to implement a stable controller while at the same time preserving the control performance from the design stage when the network resources are constrained, i.e., in the presence of noise, data loss, and large and variable delay.

3.1. Related Works and Contributions

Designing feedback control strategies in the presence of closed-loop delay is studied in the literature in several contexts [3, 6, 8, 14, 23, 25, 26, 32–34, 37, 43, 44, 56–58, 60, 62, 63, 67]. Here, we broadly categorize the related works into four orthogonal directions *D1*, *D2*, *D3* and *D4*.

D1: This direction of research focuses on the design and analysis of networked control systems by predicting states and/or delays based on different communication models [3,23–25,62,63]. For example, the works in [23,25] focus on theoretical guarantees on the robustness of the system under immeasurable variable delay. A predictor-based controller is presented and its robustness is analyzed for different uncertainties. In [62, 63], a Kalman filter is used to compute upper and lower bounds on the estimation error based on the delay probability. In [3], a pre-defined time frame is used to mitigate the delay variability where the control inputs are applied instantly after that specific time. Finally, a predictor-based analysis is performed for constant delay compensation.

D2: This direction concerns designing a robust controller that can withstand

large delay variations, packet drops and/or network faults without jeopardising stability [5, 6, 17, 34, 69]. For example, formal models for analyzing the robustness and stability of a multi-hop wireless control network (MCN) are given in [6, 69]. In [17], a fault-tolerant stabilization technique is provided, for which the necessary and sufficient conditions on the plant dynamics and the communication protocol are proposed. In a similar context, [34] gives an analytical bound on the fraction of deadline misses that can be sustained without violating the control requirements. Note that the analytical worst-case delay bounds are mostly pessimistic and a controller designed based on such a bound will result in a lower average performance.

Most of the works in *D1* and *D2* focus only on the robustness of the system. Although, in theory, a performance-aware predictor might be used to predict the control input when the delay is large, such an implementation is not feasible on mobile devices, since the actuator node does not have the bandwidth to run a computationally expensive predictor, e.g., a Kalman filter. Moreover, unlike *D1*, we do not require any knowledge of the actual delay distribution of the network, but the upper and lower bound of the delay. Furthermore, unlike *D2*, we do not design the controller considering only the worst-case delay and, thus, avoid pessimistic control design and lower average performance.

D3: This direction of works handles the large and variable delay by providing more network resources [9, 33, 37, 44, 52, 67]. In [37, 44], each data packet is simultaneously broadcasted to all possible nodes in range to increase the reliability and to reduce the number of re-transmissions, which essentially reduces the end-to-end delay. On the other hand, in [9, 33, 52, 67], high-quality network resources are provisioned in parallel to the low-quality resources to obtain a better performance. Furthermore, [67] considers using an adaptive controller to improve the performance even when using a low-quality network. Note that unlike our proposed approach, these approaches are expensive in terms of cost and/or computation and might not always be feasible.

D4: This direction of research investigates the co-design of control and network parameters [8, 26, 27, 43, 44, 57]. In [43], a holistic controller is proposed that generates actuation signals to physical plants and re-configures the wireless network (i.e., tunes the re-transmission count) to maintain the desired control performance, while saving wireless resources. The problem of selecting sampling rates and synthesizing network schedules for multiple controllers sharing a wireless network is addressed in [8, 57]. It is based on a worst-case end-to-end delay analysis [58] over the network. The work presented in [26, 27] synthesizes stable control and network schedules for a shared wireless control network in an integrated fashion. In the

same vein, [32, 60] design controllers with certain assumptions on the closed-loop delay, and then, a strict constraint on the delay is assumed while implementing the controller. Here, if the constraint is violated during implementation, then the controller has to be redesigned.

The aforementioned works do not consider the performance degradation owing to large and variable end-to-end-delays, which is the main focus of this work. Barring two exceptions in [43, 44], most of the related works either ignore the validation of the design in a real-world setup or validate the proposed design only through simulations. In contrast, we evaluate our proposed technique through experiments on a custom-built, real-world testbed.

3.2. System Model

In this work, we study networked CPSs that are commonly found in MCNs. An MCN can be represented as a tuple $\mathcal{N} = (\mathcal{P}, \mathcal{K}, \mathcal{G})$, where $\mathcal{P} = \{P_1, \cdots, P_n\}$ is the set of n physical plants, $\mathcal{K} = \{K_1, \cdots, K_n\}$ is the set of n controllers, and $\mathcal{G}$ is a graph representing the communication network between the plants and the controllers. The feedback controller K_i controls the plant P_i, where the controller receives the sensor data and sends the control inputs over the network $\mathcal{G}$. Such a system architecture, where the controller is located remotely, is common in process control [44].

3.2.1. Network Model

The network graph is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of vertices $\mathcal{V}$ are the nodes of the network, and the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ models the connectivity. An edge $(v_i, v_j) \in \mathcal{E}$, if and only if the node v_j can receive data from node v_i. We have $\mathcal{V} = V_P \cup V_I \cup \mathcal{C}$. Here, $V_{\mathtt{P}} = \{P_1, \cdots, P_n\}$ denotes the set of plant nodes. Each plant node comprises the sensor and actuator units connected locally to the physical plant (see Figure 4.2). Sensors read the states of the plant and transmit the data to the controller, while the actuator receives the control input from the controller and applies it to the plant. $\mathcal{C}$ denotes the control node responsible for computing the control input based on the states of the plant. The communication between the plant nodes and the controller is realized by the set of nodes $V_{\mathtt{I}}$ denoting the intermediate nodes that follow a receive and forward policy to route data over the network. A node $v_i \in \mathcal{V}$ can transmit data to a set of nodes, $N(v_i) = \{v_j | (v_i, v_j) \in \mathcal{E}\}$ within its transmission range.

We consider a setting where the individual plants $\{P_1, \cdots, P_n\}$ are remotely controlled by software running on a shared computing platform residing in the MCN manager. The MCN manager is a centralized node $\mathcal{C}$ in the network graph $\mathcal{G}$. It collects connectivity information from all the nodes in $\mathcal{V}$, computes routing paths

for the control loops, and disseminates routing information among the nodes. A routing path is a sequence of communicating pairs of nodes through some channels, i.e., frequencies. According to Figure 4.2, a possible routing path for the control loop (P_1, K_1) is: $\langle P_1 \rightarrow I_1, f_1\rangle, \langle I_1 \rightarrow I_2, f_1\rangle, \langle I_2 \rightarrow I_4, f_1\rangle, \langle I_4 \rightarrow C, f_1\rangle, \langle C \rightarrow I_4, f_1\rangle, \langle I_4 \rightarrow I_3, f_1\rangle, \langle I_3 \rightarrow I_1, f_1\rangle, \langle I_1 \rightarrow P_1, f_1\rangle$, where, e.g., in the first hop, the plant node P_1 sends state measurement data to the intermediate node I_1 using the frequency f_1.

3.2.2. Feedback Control Systems

In this work, we study linear and time-invariant (LTI) systems for which the discrete time mathematical model can be written as:

$$x[k+1] = Ax[k] + Bu[k], \quad y[k] = Cx[k]. \tag{3.1}$$

For an n-th order system with m outputs and p inputs, the vectors $x[k] \in \mathbb{R}^n$, $y[k] \in \mathbb{R}^m$, and $u[k] \in \mathbb{R}^p$ denote the plant state, the output, and the control input respectively at time $t = kh$, where $k = 0, 1, 2, ...$, h is the sampling period, and t is the time for the k-th sampling instant. The matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, and $C \in \mathbb{R}^{m\times n}$ represent the discrete-time state transition matrix, the input matrix, and the output matrix respectively. We further consider a state-feedback controller for which the control law is:

$$u[k] = Kx[k] + Fr, \tag{3.2}$$

where K is the feedback gain, F is the feedforward gain, and r is the reference output. By combining Eqs. (3.1) and (3.2), we get the closed-loop system model as follows:

$$x[k+1] = (A + BK)x[k] + Fr = A_{cl}x[k] + Fr. \tag{3.3}$$

The eigenvalues of A_{cl} must lie inside the unit circle for the closed-loop system to be asymptotically stable. Besides stability, control requirements are often specified, among others, in terms of settling time, rise time, overshoot and quadratic cost. We assume that the designed controller $\{K, F\}$ must meet the requirements. Standard design techniques, e.g., pole-placement and linear quadratic regulator (LQR) [7], can be employed to determine K, while the final value theorem [15] can be applied to compute F.

Note that, in this work, we do not introduce a new controller design technique. We instead show how the high performance that is obtained in the controller design phase can be preserved when the control loop is being implemented over a wireless network.

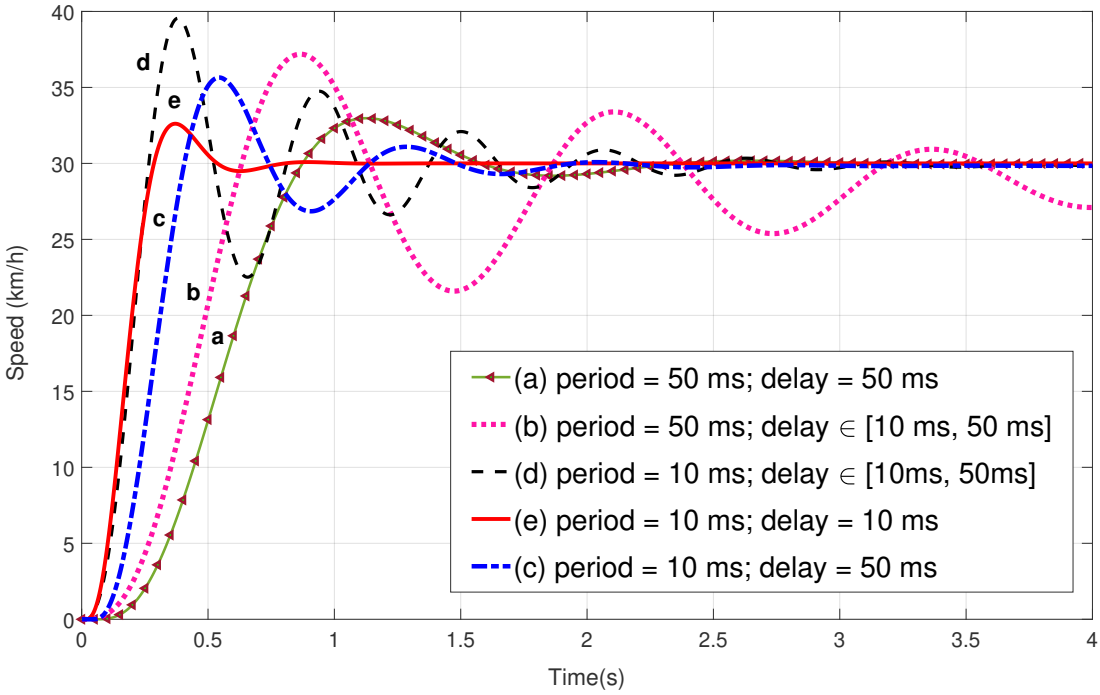

Figure 3.1: Effect of variable delay in system response.

3.3. A Motivational Example

For loops requiring fast sampling, the period choice which accounts for worst-case delay may not ensure close-loop stability. Conversely, for achieving fast feedback, an aggressive controller could be designed based on the best-case or average delay when computing the sampling rate. Though resulting in a faster response, the system can still become unstable as we will see next. This especially holds true when large delay variations are possible. Given that in-vehicle wiring lengths and costs are now considered to be very high, in the future there is a distinct possibility of using wireless networks in the automotive electrical and electronic (E/E) architecture. Hence, in this section, we consider the model of a cruise control system from [55] as a motivational example to demonstrate the challenges in the implementation of a high-performance controller. In a cruise control system, the controller regulates the vehicle speed at a reference level by adjusting the engine throttle angle. Let the reference speed be $r = 30\,\text{km/h}$ and the settling time requirement be 1 s. Settling time is the time taken by the system output to reach and stay within a certain threshold (e.g., 1 %) of the reference value. We assume that the controller is implemented in an MCN setting, where the sensing-to-actuation delay can vary between 10 ms and 50 ms. We consider different sampling periods h for the simulation.

h=50 ms: We now consider a controller that is designed based on a sampling period $h = 50\,\text{ms}$ given by $K = \begin{bmatrix} 23.851 & 10.29 & 2.76 \end{bmatrix}$ and $F = 875.64$. Here, we use the LQR controller design technique.

a) We first consider an implementation where the control input is always applied after a time interval of 50 ms from the sensing. That is, even if the control input has reached the actuator with a lower delay, it will not be applied until the worst-case delay has elapsed. This is also a standard technique for implementing a controller in an MCN setup for a more predictable control performance. In Figure 3.1, the green colored solid curve with maroon arrow markers gives the control response for such an implementation, when the plant is sampled at 50 ms. Here, the slow sampling rate leads to a low control performance, i.e., the settling time is greater than 1 s.

b) Next, we consider an implementation where the control input is applied as soon as it reaches the actuator. The pink dotted curve in Figure 3.1 gives the plant response when the delay varies randomly between 10 ms and 50 ms. In this case, we can observe that the system might take longer to settle down than the first implementation, despite the average delay being less than 50 ms. This is because the system here experiences a jitter, i.e., the time between two successive actuation instants varies. However, the controller is tuned for a fixed period of 50 ms, and correspondingly, a control input is expected to be held for 50 ms. Now, if a control input is held longer, it might result in an overshoot. Similarly, if it is changed to early, the response might become slower.

h=10 ms: Next, we consider a controller that is designed for a sampling period $h = 10$ ms using the LQR design technique. Note that the sampling period h is equal to the best-case end-to-end delay, i.e., $h = 10$ ms. The obtained controller is given by $K = \begin{bmatrix} 873.206 & 131.223 & 10.045 \end{bmatrix}$ and $F = 826.2$.

c) We first consider an implementation where the input is always applied with a delay of $d = 50$ ms. The blue dash-dotted line shows the response for such an implementation. It may be noted that the settling time requirement is not met.

d) We further consider an implementation where the delay varies between 10 and 50 ms and the control input is applied as soon as it reaches the actuator. The black dashed line shows the response for this case. Again, the requirement is violated and the settling time is longer compared to the first implementation.

Note that the settling time in Case d) is even longer than that obtained for the implementation in Case a). In Case a), the output rises quickly towards the reference, however, it experiences a large overshoot because of which it takes a longer time to settle down. The main reason for this overshoot is the large jitter experienced by the system. When a controller is designed for a sampling period of 10 ms and a constant delay equal to the sampling period, it is expected that the control input will be adjusted every 10 ms. However, with a delay variation from 10 ms to 50 ms, the two successive actuations can be separated by 50 ms in the worst-case. Now, when the output is closer to the reference, the control

input might not get adjusted accordingly because the new value has not reached the actuator. In such a case, the output can go beyond the reference and might experience a large overshoot. Note that such events are nondeterministic when there is a delay variation, and in certain cases, the output might continuously oscillate around the reference.

e) We also consider a hypothetical case where the delay is constant at $d = 10\,\text{ms}$. The red solid line in Figure 3.1 shows the response for this case. Here, we get a significantly faster response with a settling time less than 1 s, which meets the requirement. Note that with $h = 10\,\text{ms}$, in each of the implementations, the system response is faster, i.e., it reaches the reference within 0.5 s. However, for large and variable delays, the plant oscillates around the reference for a significant amount of time before settling down.

From the above experiment, we make the following observations: (i) A higher control performance might be obtained by designing a controller with a smaller sampling period. (ii) When a controller experiences a large delay variation, the control performance deteriorates. (iii) When the delay is fixed, the control performance improves with a smaller delay. Thus, our goal is to show how to maintain the higher control performance obtained during controller-design time with a smaller sampling period over the network having large delay variations.

3.4. Proactive Feedback Strategy

The overview of our proposed feedback strategy is as follows. Let the delays in the forward path (i.e., sensor-to-controller) and the return path (i.e., controller-to-actuator) of a control loop vary from $\check{\delta}_f$ to $\hat{\delta}_f$ and from $\check{\delta}_r$ to $\hat{\delta}_r$ respectively. Our proposed scheme ensures that sensing and actuation are performed periodically at discrete time instants according to a given period h, where h can be smaller than $\hat{\delta}_f + \hat{\delta}_r$. Now, let us assume that the sensor data corresponding to the k-th sampling instant reaches the controller after a delay $\delta_f \in [\check{\delta}_f, \hat{\delta}_f]$. The delay is calculated based on timestamps. Using the minimum delay (i.e., $\check{\delta}_r$) in the return path, we can determine the earliest actuation time instant (i.e., $\check{k}_a = \check{F}(k, \delta_f + \check{\delta}_r)$) before which the control data might reach the actuator. Furthermore, considering the maximum delay in the control loop (i.e., $\hat{\delta}_f + \hat{\delta}_r$), we can also determine the latest actuation instant (i.e., $\hat{k}_a = \hat{F}(k+1, \hat{\delta}_f + \hat{\delta}_r)$) before which the control inputs based on the next state of the plant will reach the actuator. Here, $\check{F}(\cdot)$ and $\hat{F}(\cdot)$ are functions. We compute appropriate control inputs, $U_{\check{k}_a, \hat{k}_a}$, for all possible actuation instants from $\check{k}_a$ to $\hat{k}_a$, based on the latest knowledge of the state of the plant. Towards computing a control input $u[k']$ for the actuation instant $k' \in [\check{k}_a, \hat{k}_a]$, we apply the feedback control law on the predicted state $\hat{x}[k']$. Here,

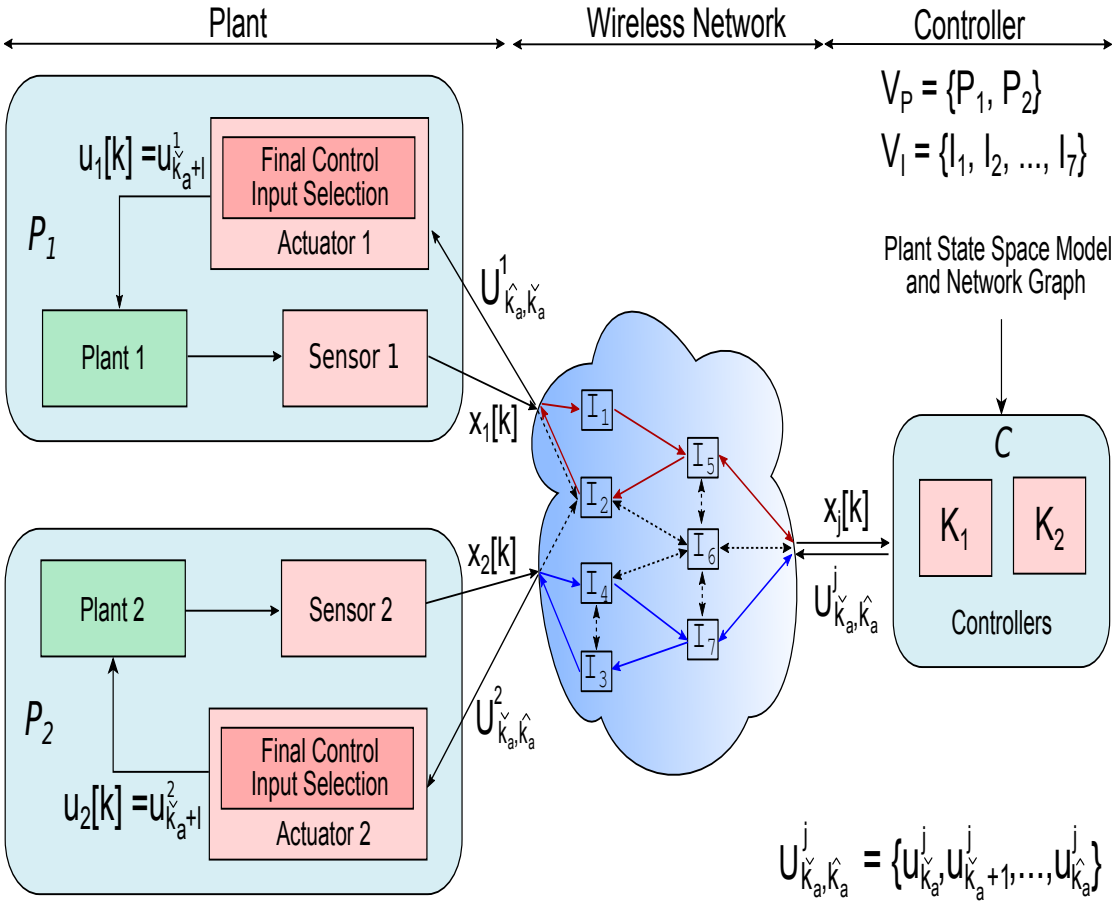

Figure 3.2: Proposed proactive feedback scheme in a multi-hop wireless network.

$\hat{x}[k']$ is calculated based on the closed-loop system model comprising the plant and the controller. The computed set of control inputs for all possible delays, i.e., $U_{\check{k}_a,\hat{k}_a}$, is sent to the actuator in a single data-packet. For a particular actuation instant, the actuator finds the appropriate input to apply from the latest set of control data that it has received.

This scheme is depicted in Figure 4.2 for a wireless multi-hop control network (MCN) with two control loops. Here, $U^1_{\check{k}_a,\hat{k}_a}$ and $U^2_{\check{k}_a,\hat{k}_a}$ contain the control inputs, i.e., $U^j_{\check{k}_a,\hat{k}_a} = \{u^j_{\check{k}_a}, u^j_{\check{k}_a+1}, \cdots, u^j_{\hat{k}_a}\}$,
$\forall j \in \{1, 2\}$, for loop 1 and loop 2 respectively. For each actuation instant, the actuator applies the appropriate input from the latest set received by it. Thus, we set $u_j[k] = u^j_{\check{k}_a+l} \in U^j_{\check{k}_a,\hat{k}_a}$, $j = 1, 2$.

Now we discuss each step in details. Let us assume a minimum delay $\hat{\delta}$ and a maximum delay $\check{\delta}$ in the considered network.

3.4.1. Predictor Operation

Our proposed strategy relies on a predictor, which works as follows. The predictor takes as inputs (i) the states of the plant ($x[k]$) and (ii) the delay experienced by

the packet carrying the state information from the plant to the controller (δ_f). Based on δ_f, the maximum closed-loop delay, and the minimum delay in the return path (i.e., controller to actuator), it first predicts the actuation instants for which control input might be required to be applied based on the received state information. Now, corresponding to these predicted actuation instants, it predicts the state of the plant considering the evolution of the system based on the proposed strategy. Note that the predictor operates in an event-triggered fashion, i.e., its operation is triggered by the arrival of the plant data as sent by the sensor unit in the plant node of the control loop.

Predicting the actuation instants: Let $\check{\delta}_r$ be the minimum delay and $\hat{\delta}_r$ the maximum delay in the return path. Given δ_f as the delay experienced in the forward path (i.e., sensor to controller), we can calculate the earliest actuation instant $\check{k}_a$ in which the control input calculated based on the current state information $x[k]$ can be applied as:

$$\check{k}_a = k + \lceil (\delta_f + \tau_c + \check{\delta}_r)/h \rceil \tag{3.4}$$

where τ_c is the computation time for the control input when receiving the sensor data. Now, we must consider the maximum end-to-end delay to compute the latest actuation instant $\hat{k}_a$ where the control input calculated based on the current state information $x[k]$ might be applied. This is the instant after which we can guarantee that the control input calculated based on the next state information will reach the actuator. Thus, the latest actuation instant $\hat{k}_a$ can be calculated as follows:

$$\hat{k}_a = k + \lceil (\delta_f + \tau_c + \hat{\delta}_r)/h \rceil \tag{3.5}$$

The controller needs to compute appropriate control inputs for all instants from $\check{k}_a$ to $\hat{k}_a$.

State Prediction: In this work, we derive a predictor to estimate the plant state at the k'-th actuation instant ($\check{k}_a \leqslant k' \leqslant \hat{k}_a$) based on the following information.

i) The plant dynamics given in Eq. (3.1).

ii) The delay (in terms of number of samples) from sensing to actuation, i.e., $\Delta = k' - k$.

iii) The last measured state $x[k]$ that has reached the predictor.

iv) All previous control inputs, i.e., $u[k + j]$, where $0 \leqslant j \leqslant \Delta - 1$.

Formally, we define the dynamics of predictor as follows [25].

$$\hat{x}[k'] = A^{\Delta}x[k] + \sum_{j=0}^{\Delta-1} A^{\Delta-1-j}Bu[k+j] \tag{3.6}$$

Following Eq. (3.6), for each such predicted actuation instant $k' \in \{\check{k}_a, \check{k}_a + 1, \cdots, \widehat{k}_a\}$, the predictor estimates the plant-state $\hat{x}[k']$ based on the last measured state valuation $x[k]$ and all the intermediate control inputs $\{u_{\check{k}_a}, u_{\check{k}_a+1}, \cdots, u_{k'-1}\}$.

3.4.2. Controller Operation

On receiving the estimated state of the plant, $\hat{x}[k'] = \hat{x}[k+\Delta]$, for a sampling instant k', the controller computes the control input, $u[k']$ following Eq. (3.2). Note that the control input $u[k']$ calculated for the actuation instant k' can be used to predict the state $x[k'+1]$. Thus, an efficient way of implementing the controller and the predictor is to predict a state followed by computation of the control input for an actuation instant. Then, the next state can be predicted based on the predicted state and the control input of the last instant as:

$$\widehat{x}[k'+1] = A\widehat{x}[k'] + Bu[k']. \tag{3.7}$$

Let us consider an example when $x[2]$ reaches the control node after a delay of one sample. The minimum delay in the return path is one sample and the maximum end-to-end delay is 4 samples. Here, $\check{k}_2 = 4$ and $\widehat{k}_2 = 6$. Thus, the controller needs to compute $u[4]$, $u[5]$ and $u[6]$ based on $x[2]$. We denote $u[k']$ calculated based on $x[k]$ as $u^k_{k'}$. Thus, the control input vector sent by the control node is $U^k = \{u^k_{\check{k}_a}, u^k_{\check{k}_a+1}, \cdots, u^k_{\widehat{k}_a}\}$.

The controller relies on the assumption that the actual state at the k-th instant will be equal to the predicted state, i.e., $\hat{x}[k] = x[k]$. A controller designed with an assumption of *zero delay* when implemented using our proposed strategy does not violate the assumption. Hence, the stability and the control performance of the closed-loop system are preserved from the controller design stage to the implementation.

3.4.3. Actuator Operation

At each actuation instant, the actuator selects the most appropriate control input from the control input vector currently available to it and applies the input to the plant. Thus, it ensures *zero delay in actuation* of the control input and exhibits the behaviour of a zero-delay sampled system with the best choice of the sampling period. More details are given next.

Control Input Selection: Let the end-to-end delay encountered by the control

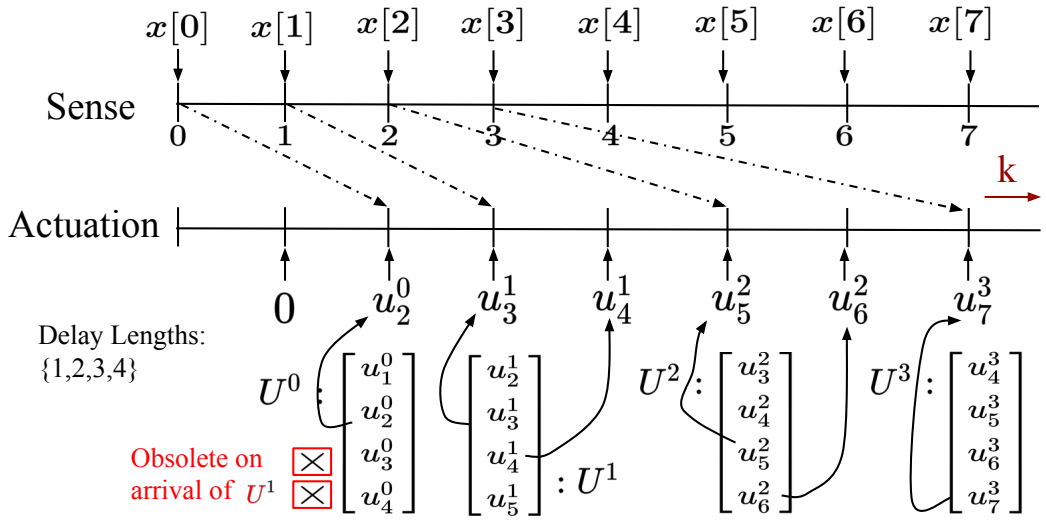

Figure 3.3: Implementing zero delay in actuation.

input U^k for reaching the actuator be Δ samples, i.e., U^k arrives at the k'-th actuation instant, where $k' = k + \Delta$. Then, the actuator selects the control input $u^k_{k+\Delta} \in U^k$ and subsequently applies the control input to the plant, i.e., $u[k'] = u^k_{k'} = u^k_{k+\Delta}$. E.g., for $U^2 = [u^2_4, u^2_5, u^2_6]$, if the actual delay-length is $\Delta(k) = 3$, the actuator applies $u[5] = u^2_5$. Note that $u^k_{k'}$ is calculated based on the predicted state valuation $\widehat{x}[k']$. Here, $\widehat{x}[k']$ is calculated using Eq. (3.6) based on the plant state $x[k]$, which is sensed at the k-th sampling instant.

An illustrative example is given in Figure 3.3. Here, the actuator sets $u[1] = 0$, since no control input has reached by that time. On reaching the control input vector U^0 at the 2^{nd} time step, the actuator sets $u[2] = u^0_2$. Similarly, it sets $u[3] = u^1_3$ on receiving U^1 at the 3^{rd} time step. Since it does not receive any input from the controller at the 4^{th} time step, it sets $u[4] = u^1_4$ from the previously received input vector U^1 to maintain zero delay in actuation. Similarly, it sets $u[6] = u^2_6$ at the 6^{th} time step, while it updates $u[5]$ and $u[7]$ with u^2_5 and u^3_7 on receiving U^2 and U^3 at the 5^{th} and 7^{th} time steps respectively. Note that the control input is updated at each and every sample.

3.5. Experimental Evaluation

To evaluate the behavior of our proposed technique in real-world, we present experimental results obtained on a CPS testbed. We next describe our experimental setup, i.e., the CPS testbed, and then give experimental results.

3.5.1. Experimental Setup

We used the hardware/software co-designed testbed as depicted in Figure 3.4 in our experiments, which is built using off-the-shelf radio hardware. It provides a

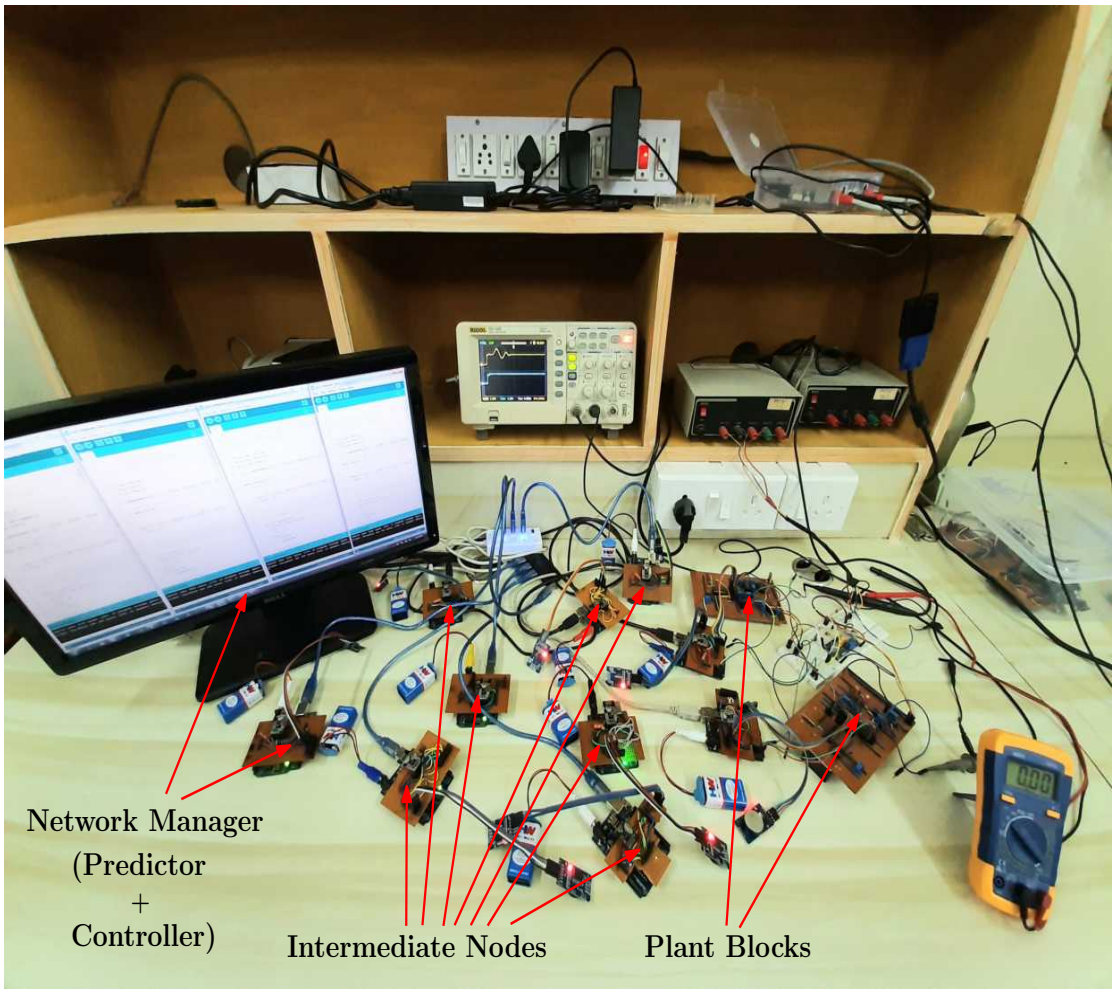

Figure 3.4: The MCN testbed.

unique platform to implement, run, and validate the proposed strategy for any given MCN specification.

Though our testbed is well equipped to configure any given MCN configuration, in this work, the MCN specification for the testbed that we have considered for evaluating the proposed strategy comprises two physical plants, seven (battery powered) intermediate nodes, and a network manager (i.e., the control node) containing the controller and the predictor. The network topology is depicted in Figure 4.2. The forward and return paths are colored red for loop 1 and blue for loop 2. Figure 3.4 depicts all devices of the testbed within a single image. However, during the experiments, these nodes were distributed in a 6 m × 6 m room ensuring at least 2 m separation between any two nodes. Significantly larger distances can be realized easily, since our network supports multi-hop communication with a single hop transmission range of up to 30 m. It is very much compatible to carry over the experiments and the results to any hardware setup over a larger wireless network having any standard protocol. Each one of the nodes are open to outer obstruction from other electronic equipment and human activities.

Based on a given MCN specification, the software tool automatically generates and deploys the firmware to all the Arduino based nodes in the network. The details of our customly developed software tool, which supports specifying and configuring a wireless network through a GUI-driven user interface, can be found in [29]. This firmware configures the nodes as well as their network interfaces for realizing the given MCN configuration such that, when the full system runs, each node performs its designated set of transmissions and receptions at user-specified frequencies, bit rates and power levels, thus realizing the routing solution for each control loop as indicated in the MCN specification. For generating timestamps, all nodes are equipped with a DS3231 precision real-time clock (RTC), which get synchronized before carrying out any experiment. Once the hardware components and the wireless network are configured, the system execution starts. Plant data is sensed and sent over the forward path to the network manager as scheduled. The network manager works on the plant data and computes the control inputs accordingly which will be sent back to the plant over the return path for actuation. All the nodes in the MCN run in event triggered mode i.e. only on receiving data, each node wakes up and execute its corresponding transmission. Event triggering allows us to tolerate delays and minimize packet drops while adjusting the actuation.

Plant Node: In our testbed, Double Integrator Circuits (DICs) form the physical plants, however, the hardware implementation of other physical plants can be easily integrated in the future. We consider DICs that have the following discrete time dynamics [28].

Move all the details to chapter2. since there u have introcuded the plant node. always remenber it is

$$y[t] = \begin{bmatrix} 0 & 1 \end{bmatrix} x_p[t] \tag{3.9}$$

In a DIC, the control goal is to maintain a given reference voltage at the second amplifier's output by controlling the first amplifier's input voltage. In our hardware implementation of DIC, we use two operational amplifiers (UA741) connected in series, thereby computing the integral over the integral of the input signal of the first amplifier. The plant node consists of a DIC, one Digital to Analog Converter (DAC), two voltage level shifters, one voltage divider, and an ATmega328P-based Arduino UNO microcontroller running at 16 MHz. The UNO board is interfaced with a nRF24L01+ radio-frequency (RF) transceiver device [49], a Real Time Clock (RTC) module [41], and a SD memory card. The outline of the plant node is delineated in Fig. 3.5. During the sensing stage, plant state measurements are sent to the microcontroller through the voltage divider and lower level shifter.

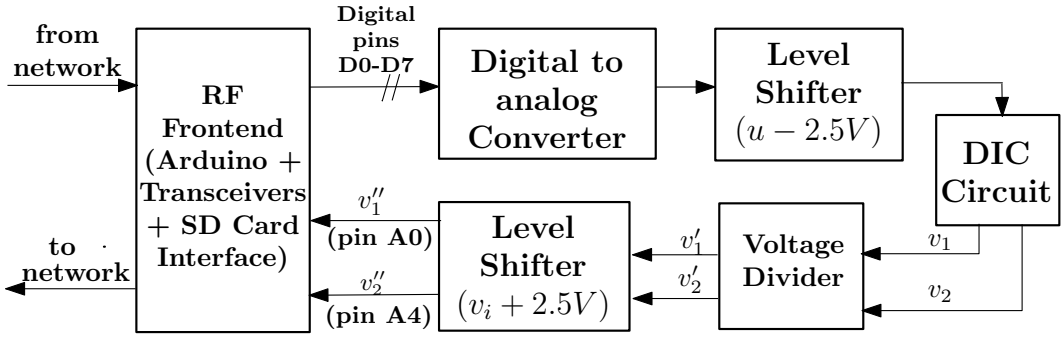

Figure 3.5: Hardware details of the physical plant node.

In case of actuation, on receiving the data from the network, the microcontroller sends it to the physical plant through the DAC and the upper level shifter.

Control and Intermediate Node: The control node (i.e., the network manager) is responsible for performing the operation of the controller and predictor. This node consists of the same hardware setup as used for the plant node. Additionally, it is connected to a host PC having high computation capabilities. We use a MATLAB-Simulink based interface for connecting the Arduino microcontroller to the host PC. For controlling the plant, the microcontroller receives the plant output, sends it to the host PC that computes the control inputs using a standard LQR control design technique [7], gets back the control inputs from the PC, and then transmits them.

The predictor model in the network manager works based on the delays as reported by intermediate nodes in the forward path of the corresponding control loop. On receiving the plant state measurements and forward path network delays

move all details to chap2. Here just mention what chnages u have made over the testbed introduced in chap 1 to

delay in the return path and plant states accordingly. Thus, two sets of control input matrices are selected for both plants based on estimated states and delays.

Each intermediate node also consists of a microcontroller + RF + RTC interface and a SD memory card. Based on the given routing paths of the control loops, each intermediate node is configured using a customly generated firmware. The firmware runs periodically and performs all transmission-reception cycles in which the node is a participant.

Wireless Network: On top of the *Enhanced ShockBurst* protocol [48], our barebone wireless network is built operating in the 2.4 GHz ISM band. The modulation scheme is Gaussian frequency shift keying (GSFK) at a data rate of 250 kbit/s. Each node is tuned to use a transmit power of −12 dBm, which is sufficient to cover the distances needed in our experiments. For avoiding collisions, all communication in the forward and return path is carried out on distinct wireless channels assigned to each of the two control loops. The network topology depicted in Figure 4.2 is realized using static routing. The basic network operation is as follows.

1. When being idle, every node listens for incoming packets on its respective channel during all times.

2. Whenever there is new data to transmit, e.g., at plant P_1, the data is transmitted immediately to the next node, e.g., I_1.

3. After such a transmission, the sending device switches to the receive mode to wait for an acknowledge packet. If the acknowledge has arrived, the described procedure repeats and the device listens for further incoming packets again. Otherwise, if no acknowledge packet was received within a time-window of 4 ms, the first re-transmission is initiated. Next, the radio listens for an acknowledge for another 500 µs, after which the next re-transmission attempt is started. Up to 8 transmission attempts are possible, which are spaced by 500 µs, each.

4. Upon a successful reception, the receiving device will immediately transmit the data to the next intermediate hop in the same manner.

When an intermediate node is part of two control loops, it relays data from a certain node to another one following its routing scheme. The radio alternates every 8 ms between both channels for listening. Upon a reception, it will first serve the corresponding control loop before listening to the channel assigned to the other control loop. For example, in Figure 4.2, the shared node I_2 may implement the following routing sequence. $\langle I_5 \rightarrow I_2, f_1\rangle, \langle I_2 \rightarrow P_1, f_1\rangle, \langle P_2 \rightarrow I_2, f_2\rangle, \langle I_2 \rightarrow I_6, f_2\rangle$. This means that I_2 is supposed to receive a packet from node I_5 and transmit to P_1 using frequency f_1 in one control loop and only after that it can perform the next transmissions corresponding to another control loop using frequency f_2. This may lead to an additional delay, e.g., when considering that the message from I_5 is delayed while the one from P_2 is ready to be transmitted. If a node is not ready for reception on a certain channel, the re-transmission mechanism that has already been described is used for realizing a later successful reception.

When a receiving node listens on a different channel w.r.t. a potential sender, the sender will start re-transmitting its packet after 4 ms, and make its additional re-transmission attempts within the next 4 ms. Therefore, once the receiver changes its channel within this time-window, it will successfully receive such a re-transmission.

Timestamping: For estimating delays, timestamps have to be generated consistently on all nodes. We realize this by using the DS3231 Precision Real Time Clocks (RTC) in each node. Before we start an experiment, we synchronize all of these RTCs to a common reference clock (in the host PC). In each node, the local clocks (in Arduino) are continuously synchronized to the ticks of the attached

RTCs. Throughout our experiments, which took approximately 15 s of wall-clock time, the skew among the different RTCs was extremely negligible, allowing us to generate precise time-stamps. Of course, when running the setup continuously, well-known time-sychronization algorithms from the literature e.g, [22], can be applied in this setup.

Clock Time Synchronization: Although our setup runs in event-triggered style, we require consistent data time-stamps for sampling and delay estimation. For providing a consistent notion of time in the distributed setup, we use battery powered DS3231 Precision Real Time Clock (RTC) modules and interface them with the microcontrollers using I2C serial communication protocol. The timing setting of all RTCs are synchronized with a common PC for global synchronization and they generate a consistent tick every second independent of the whole setup being powered on/off. The chip uses an internal temperature-compensated crystal oscillator (TCXO) that neutralizes temperature dependent clock-drifts using frequency compensation through an internal load capacitance adjustment circuit [1]. There also exists techniques for achieving finer synchronization in such modules [2] which can be used. When the setup initializes, each of the Arduino nodes start respective transmission attempts synchronizing on a common RTC tick. This can be repeated at periodic intervals (defined in multiple of hyperperiod of all loops) for synchronized operation. Once started, in each transmission attempt, the Arduino nodes can measure offset to RTC ticks at precision of microseconds using their internal clocks. In that way precise timestamps are generated for each transmission. For achieving periodic synchronization of the nodes and preventing the smallest of clock drifts, one human-in-loop possibility can be occasionally synchronizing the RTCs with a common PC clock (like a periodic maintenance). An alternative can also be implementation of well known level-sensitive node synchronization algorithms for MCNs adopting from works like [22]. Given the focus of our current work, we skip these aspects from the current implementation.

3.5.2. Experimental Results

To illustrate the advantages of our proposed proactive feedback strategy in preserving the high control performance from the design to the implementation, we conduct the following experiments in our MCN testbed.

1. We compare our proposed strategy with standard control schemes.
2. We evaluate the performance of our feedback strategy with variations in the uncertainties of the plant model.
3. We evaluate the performance of our feedback strategy with variations in the packet-drop rate of the network.

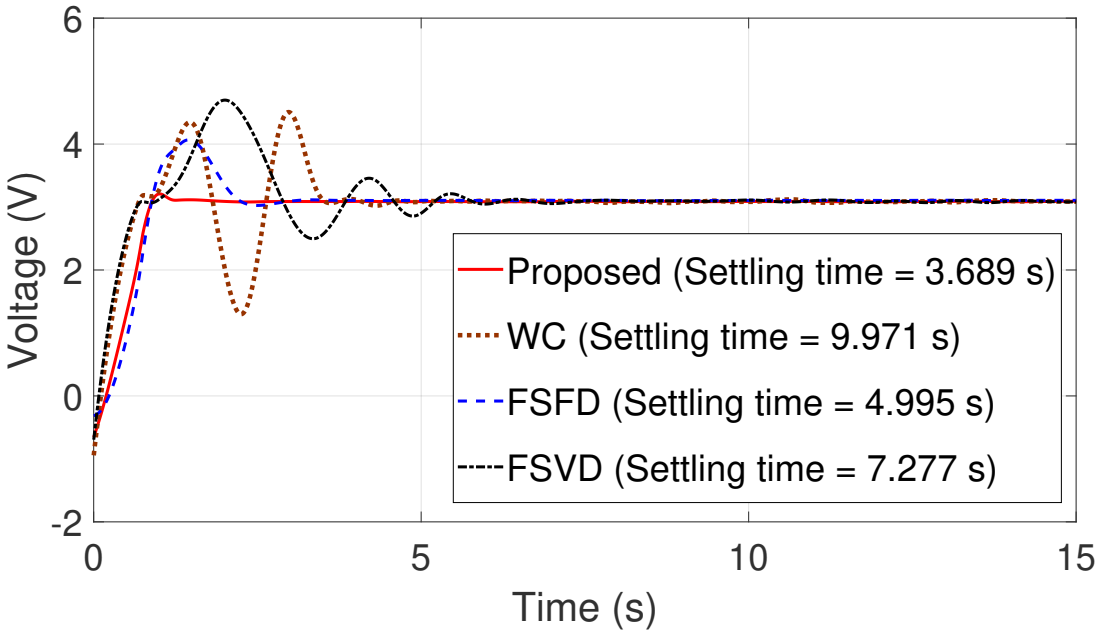

Figure 3.6: Response of plant 1 under different schemes.

We consider settling time as the performance metric for the experiments, though any other performance metric can also be used (e.g., overshoot). Both DICs (see Sec. 4.5.1) have a reference value of $3V$ (including offset). We analyze our MCN testbed to calculate the delay variation in a single transmission between a pair of nodes. We find that the maximum delay in transmitting and receiving a packet over the network and the maximum delay in processing a packet on a node is 3457 µs and 8726 µs, respectively, which leads to an end-to-end delay range of [9 ms, 50 ms] for both the control loops. The routing paths of both control loops are highlighted in Figure 4.2 (in red and blue respectively). We set the the sampling period for the DICs as $h_1 = h_2 = 10$ ms. For these plants, we design LQR controllers for a sampling period of 10 ms. We conduct each experiment for a time duration of 15 s, in each of which we first set the voltage output to 0 V for both the DICs. The control objective is to move the output voltage back to the 3 V reference value. Plant output waveforms and CSV files are obtained from a RIGOL DS1102E digital oscilloscope. We use the MATLAB(x64) version R2019a for control theoretic and other calculations.

Comparison against standard schemes:

The efficacy of the proposed proactive feedback scheme can be evaluated when we compare our proposed approach with three state-of-the-art control design schemes, which are as follows.

WC: In the "worst-case delay based control scheme [10]", the sampling periods of both DICs are chosen as 50 ms. Due to a higher sampling period, this approach

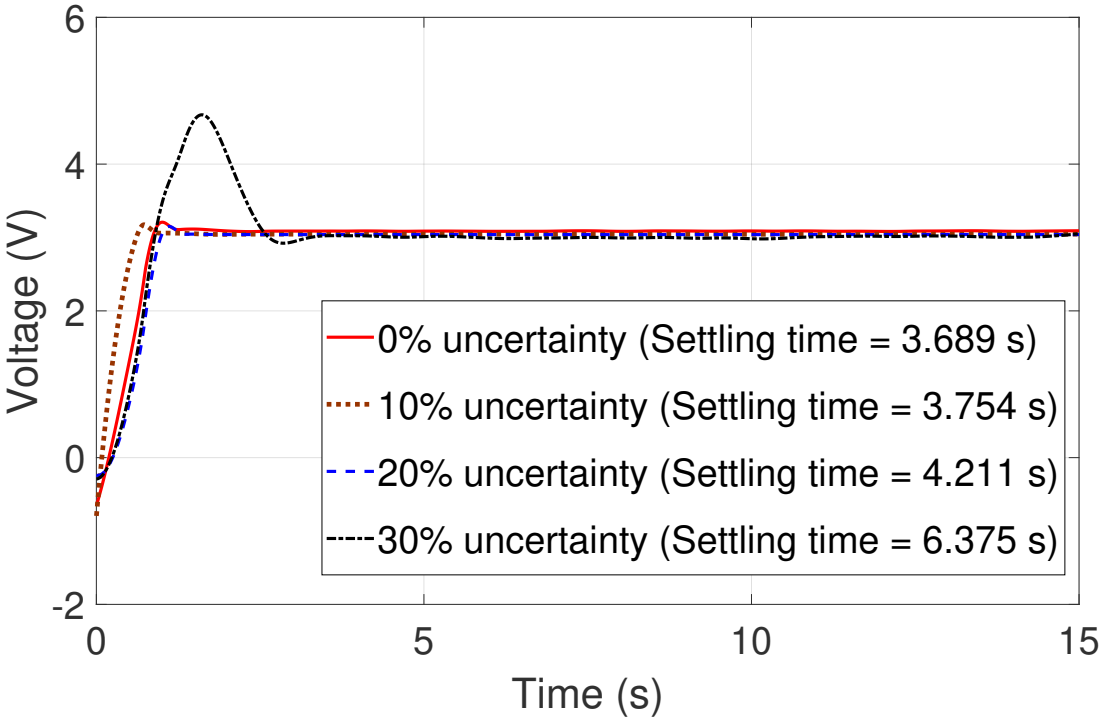

Figure 3.7: Impact of model uncertainties on plant 1.

suffers from a lower performance.

FSFD: In the "fixed-sampling and fixed-delay based control scheme [68]", the sampling periods of both DICs are chosen as 10 ms. However, we fix the sensing-to-actuation delay to 50 ms. That is, when the input reaches the actuator with a lower delay, it waits until the delay is equal to 50 ms for the actuation.

FSVD: In the "fixed-sampling and variable-delay based control scheme [8, 45]", the sampling periods of both DICs are chosen as 10 ms. But the sensing-to-actuation delay varies with time, leading to aperiodic actuation sequences.

Note that in our proposed proactive feedback strategy, the effect of this variable delay in actuation is mitigated by allowing the actuator to apply the most appropriate control input from its currently available control input vector. Figure 3.6 compares the output responses of the plant 1 using our proposed strategy and these aforementioned control techniques. As it is evident in Figure 3.6, the system settles quickly (i.e., within 3.689 s) when our proposed approach is used, as compared to other state-of-the-art control strategies. In particular, our proposed approach achieves approximate improvements of 63 %, 26 %, and 49 % compared to WC, FSFD, and FSVD respectively.

Evaluation under model uncertainties:

In order to demonstrate that the proposed strategy preserves the higher control performance reasonably well even under model uncertainties, we perform the following experiment. We can capture model uncertainties by adding an error margin

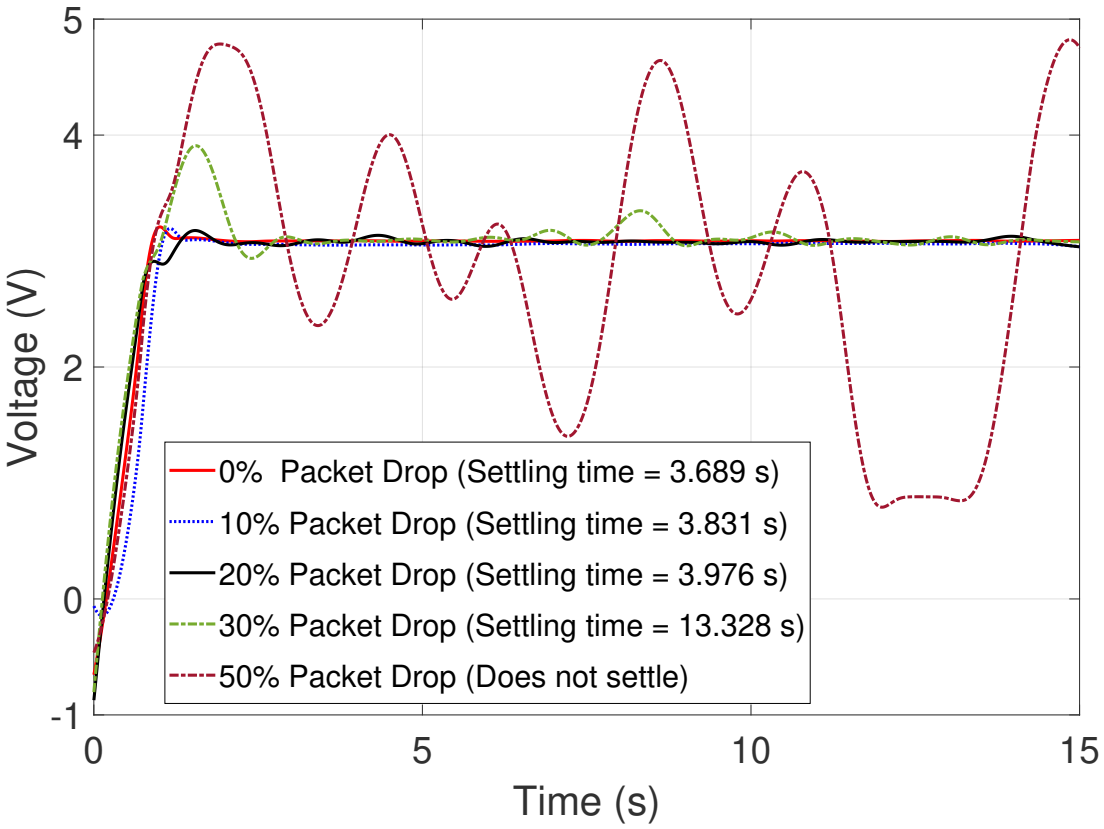

Figure 3.8: Impact of packet drops on plant 1.

to the system matrices A and B (cf. Eq. (3.1)), i.e., we obtain the modified matrices, $\tilde{A} = A + \gamma_1 A$ and $\tilde{B} = B + \gamma_2 B$, for some scalars $\gamma_1, \gamma_2 > 0$. Figure 3.7 depicts the response of plant 1 and also reports the settling time when we consider γ_1 to be 0%, 10%, 20%, and 30% respectively. For the cases shown in the figure, no uncertainty is considered in the input matrix B of plant 1, i.e., $\gamma_2 = 0$. In Figure 3.7, we see that even with 30% uncertainty, our proposed strategy provides a settling time of 6.375 s. This is better than even the settling times obtained under no model uncertainties using WC (9.971 s, cf. Figure 3.6) and FSVD (7.277 s, cf. Figure 3.6) respectively. Note that in the presence of uncertainties, the performance of each of the three state-of-the-art techniques, WC, FSVD, and FSFD, will also degrade and, hence, will be even lower compared to our proposed technique.

Evaluation under packet drops:

To establish the effectiveness of the proposed approach further, we consider a scenario of external non-idealities by modeling packet drops. We inject drops during the transmissions between P_1 and I_1 (cf. Figure 4.2), which are in the routing path of plant 1. Drops are injected at the rate of 0%, 10%, 30%, and 50%. The drop injection is implemented by adding suitable monitors in the Arduino code that probabilistically create transmission failures. All other transmission/reception events at other nodes are assumed to be ideal. Figure 3.7 shows the responses

Table 3.1: Settling Time (in s) of Plant 2

Performance of our proposed feedback scheme								
3.7								
Performance against other techniques			**Performance against model-uncertainties**			**Performance against packet drops**		
WC	**FSFD**	**FSVD**	**10%**	**20%**	**30%**	**10%**	**20%**	**30%**
10.4	**4.8**	**7.5**	**3.9**	**4.4**	**7.1**	**3.8**	**4.2**	**12.7**

of plant 1 with different drop rates. Note that with the gradual increment of the packet drop rate from 10% to 30%, the settling time for plant 1 increases from 3.831 s to 13.328 s, exhibiting a gradual performance degradation, whereas the system becomes unstable for a 50% packet drop rate.

Results for Plant 2:

We perform the same three experiments for plant 2, and observe a similar performance as we have shown earlier for plant 1. In this case, drops are injected during the transmissions between I_3 and I_7 (see Figure 4.2) in the routing path of plant 2. The comparisons of settling times for the aforementioned experiments are reported in Table 3.1. We get a settling time of 3.7 s with our proposed proactive scheme, which outperforms the settling times obtained for the other three state-of-the art control techniques. As reported earlier for plant 1, we also get satisfactory results with our scheme in case of plant 2 in the presence of model uncertainties and packet drops. In the case of 50% packet drops, plant 2 becomes unstable.

Please add conclusion and remarks as given in chap 2

Chapter 4
AEW:Adaptively-Controlled Energy-Efficient Wireless CPS

Multi-hop wireless cyber-physical systems (CPSs) are becoming popular, owing to features like low-cost sensing, deployment flexibility, low-power implementation, and fast data transfer [30, 38, 44]. Most wireless CPS applications have stringent requirements in terms of both *energy efficiency* and *control performance*. The later essentially translates to closing the loops fast enough implying *short end-to-end timing* guarantees, typically in the order of tens of milliseconds. Applications include e.g., smart manufacturing, intelligent transportation, and drone swarms. In the near future, multi-hop wireless networks will also be employed, e.g., in automotive applications, both intra- and inter-vehicular. In the later case, where multiple vehicles and roadside units must interact, certain nodes (e.g., in the roadside units) are either battery-powered, or are run by energy harvesting. In such applications, both the packet transmission reliability and the energy consumption play a key role for proper functioning of such high performance control applications. However, in practice, this becomes a challenging problem due to the limited battery life, small memory footprint, and failure prone nature of wireless infrastructures. In this work, we address this issue by proposing a framework **AEW** that 1) copes with the reliability issue by adaptively regulating the number of retransmissions, 2) guarantees desired control performance by adaptively changing the control mode of operation, and 3) ensures minimal energy consumption by regulating the power consumption profiles of wireless devices.

4.1. Overview of AEW

The main concept behind AEW is as follows. For stabilizing a control loop, a set of sampling periods may be admissible. While smaller periods typically ensure better performance, it may be less guaranteed due to varying network quality. However, with larger periods, though one may just meet a baseline performance, the guarantee of that performance will be higher, since the increased period sup-

ports a higher number of packet retransmissions. Based on this idea, we propose a novel hierarchical control architecture at the heart of AEW, that does the following operation.
1) At the high level, it computes possible network schedules for different combinations of controllers, which can handle performance degradation by allowing dynamic switching between some pre-synthesized control modes. For each mode, the network schedules are synthesized satisfying the end-to-end timing guarantee of each control loop.
2) At the lower level, it maintains the reliability requirement of each control loop. It computes node-level target reliability, which if satisfied, would by construction guarantee the desired control performance of a loop. This is provided as a set-point to node-level local PID controllers, which then try to achieve the requisite reliability requirement through *just enough* packet retransmissions in the presence of network non-idealities.

AEW minimizes the communication energy cost by 1) pre-synthesizing network schedules and distributing them at the deployment time instead at runtime, 2) driving each node to a low-power state at runtime when not in active operation.

4.2. Related and Existing Works

Existing works on reliable protocols for low-power wireless CPSs can be categorized as follows.

1. Flooding-based Approaches: Glossy [20] and RedFixHop [18] leverage a network-wide flooding technique for reliable communication. TTW [38] extends this using a schedule of communication rounds. The main drawback of these techniques is their energy demand due to the higher number of transmissions. In contrast, CXFS [13] reduces the set of nodes participating in the flooding for saving energy. However, no external interference is considered. Reliability enhancement through retransmissions is achieved in [51]. However, none of these works adjust the number of retransmissions and the control strategy adaptively at runtime to trade-off reliability and energy consumption, as we propose in this work.

2. Channel Hopping: Another possibility to improve the transmission reliability is frequency diversity, i.e., transmitting different packets on different wireless channels [31, 35]. The work in [31] utilizes IEEE802.15.4e Timeslotted Channel Hopping (TSCH) together with flooding to combat external interference and multi-path fading. Similarly, [35] adjusts the channel hopping scheme adaptively to the link quality. Adaptive Channel Mapping (ACM) is used in [19] to increase reliability by avoiding channels that are influenced by co-located networks. However, none of these methods are germane to control-oriented CPS applications, as they hardly support end-to-end timing guarantees of feedback control loops.

3. Control-Theoretic Solutions: A control-scheduling co-design for low-power wireless CPS based on the TTW protocol is presented in [44]. While it considers the schedule to be immutable, we in this paper adjust the schedule and the number of re-transmissions adaptively. We thereby save energy by avoiding unnecessary flooding. A recent work [30] proposes a feedback strategy to pre-calculate the future control inputs and adaptively apply them on demand to improve the overall reliability. However, this work doesn't consider the energy efficiency of the network.

4.3. System Description

A multi-hop wireless CPS is a tuple $\mathcal{N} = (\mathcal{P}, \mathcal{K}, \mathcal{G})$ consisting of a set of n physical plants $\mathcal{P} = \{P_1, \cdots, P_n\}$, a set of n feedback controllers $\mathcal{K} = \{K_1, \cdots, K_n\}$, and a network $\mathcal{G}$ where the i-th controller K_i controls the plant P_i remotely over the shared network $\mathcal{G}$.

4.3.1. Plant-Controller Model

We consider discrete-time linear time-invariant (LTI) plant models given by: $x[k+1] = Ax[k] + Bu[k]$, $y[k] = Cx[k]$. For an n-th order system having m outputs and p inputs, the vectors $x[k] \in \mathbb{R}^n$, $y[k] \in \mathbb{R}^m$, and $u[k] \in \mathbb{R}^p$ represent the plant state, the output, and the control input at time $t = kh$, where h is the sampling period, $k = 0, 1, \cdots$ are the sampling instants, and t is the time at k-th sampling instant. The matrices A, B, and C describe the transition matrix, the input map and the output map for the plant. The dynamics of the state-feedback controller is: $u[k] = Kx[k]$, where K is the feedback gain. Hence, for a control loop (P, K), the closed loop dynamics is:

$$x[k+1] = (A + BK)x[k] = \mathcal{A}_{cl}\ x[k]. \tag{4.1}$$

A shorter value of sampling period h provides faster response from the controller and thus better control performance. For asymptotic stability all the eigenvalues of $\mathcal{A}_{cl}$ must lie within a unit circle. The performance, i.e. the quality of control (QoC) is generally quantified with respect to the user requirements, e.g., nature/speed of the system response. Hence, we use *settling time* as the performance metric.

4.3.2. Time-slotted Wireless Communication

The time-slotted architecture has the basic unit called as the *time-slot* (T_{Slot}) which can be defined as:

$$T_{Slot} = T_{Start-up} + T_{Comm} + T_{Pros}. \tag{4.2}$$

The entire time-slot structure is shown in Fig. 4.1. $T_{Start-up}$ is the time the radio

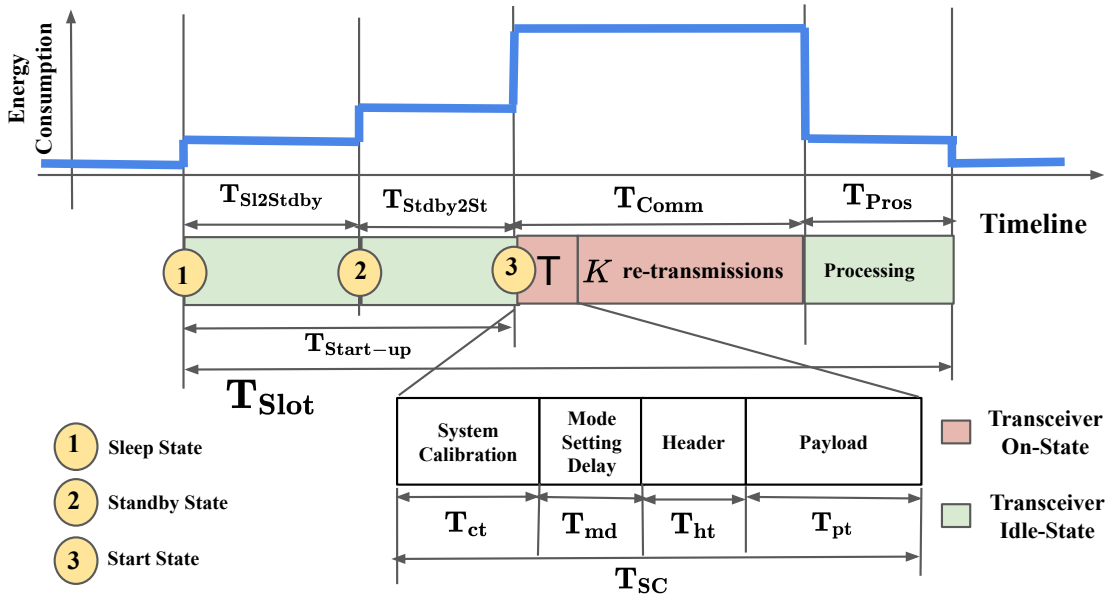

Figure 4.1: Details of a time-slot with energy consumption profile.

needs for switching from the sleep mode to the active mode. It can be further decomposed into: $T_{Start-up} = T_{Sl2Stdby} + T_{Stdby2St}$, where $T_{Sl2Stdby}$ is the time to *wake up from sleep* and go to the *standby state* and $T_{Stdby2St}$ is the transition time from the standby state to communication-ready, i.e., the *start state*. Next, T_{Comm} is the transmitter-receiver communication time (which is explained below), whereas T_{Pros} is used to process data and to go to a low-power (sleep/standby) state. Let T_{SC} be the time required for the transmission/reception of a single packet. It is $T_{SC} = T_{ct} + T_{md} + T_{ht} + T_{pt}$, where T_{ct} is the radio system calibration time, T_{md} is the delay time needed to set the transmitting/receiving mode, T_{ht} and T_{pt} are the time required to transmit the data packet header composed of M_{hd} bytes and the message payload composed of M_{pld} bytes respectively. If a radio transceiver transmits at a bit rate of N_{bit}, then $T_{ht} = (8 \times M_{hd})/N_{bit}$, and $T_{pt} = (8 \times M_{pld})/N_{bit}$. If a successful transmission of an M byte packet is obtained after λ retransmissions, then we have $T_{Comm}(M, \lambda)$ as,

$$T_{Comm}(M, \lambda) = (\lambda + 1) \cdot (T_{ct} + T_{md} + T_{ht} + T_{pt}) \tag{4.3}$$

As delineated in Fig. 4.1, during $T_{Start-up}$ and T_{Pros}, the transceiver remains idle without receiving/transmitting, whereas it remains on during T_{Comm}. This on-state time duration can vary since it depends on number of retransmissions. Hence, following Eq. (4.2), $T_{Slot}(M, \lambda) = T_{Start-up} + T_{Comm}(M, \lambda) + T_{Pros}$.

In order to avoid the wake-up delay, nodes that aren't part of any transmission go to the *sleep state*. The other radios, which actively participate in the network switch between *standby state* and *start state* in each active slot, as given by its

communication activity information (cf. Fig. 4.1 and more on this later). In case a node is configured for some λ retransmissions, they go to the *standby state* after this requisite number of retransmissions. Further, they remain at *standby state* in each inactive slot. Hence, Eq. (4.2) applies when a node that has previously not been part of any communication, but now starts participating. This can happen either in the initial set up phase, or after a dynamic change in the network schedules (explained next). Hence, for steady state operations with λ retransmissions, Eq. (4.2) becomes,

$$T^{e}_{Slot}(M, \lambda) = T_{Stdby2St} + T_{Comm}(M, \lambda) + T_{Pros} \tag{4.4}$$

4.3.3. Network Model

The network $\mathcal{G}$ consists of multiple nodes identified by the set $\mathcal{V}$. They communicate by forwarding packets to each other within their transmission range. For each plant P_i, a node is associated having its sensor and actuator units connected locally to it. The network is managed centrally by the node $\mathcal{C}$, called as the *control node*. All feedback controllers are also running on a shared computing platform residing at $\mathcal{C}$. For saving energy, all nodes are in a sleep state when idle. For every node, the network schedules define a) the time instants the node should wake up, and b) destination nodes to whom it should forward a packet. The control node $\mathcal{C}$ collects the connectivity information from all nodes, computes *network schedule* for all control loops, and then based on that disseminates the *activity information* among the nodes. The *activity information* $\mathcal{A}_v$, of a node v is a sorted list of *records* of the form: [`LT, ON/OFF, NID, CID, T/R`], where `LT` is the local timestamp at v, `ON/OFF` indicates whether the radio is on or off at `LT`, `NID` is the node id to communicate with, `CID` is the array of channel ids that v uses during its communication via channel hopping, `T/R` indicates whether v transmits or receives. $\mathcal{A}_v$ is sorted based on `LT`.

Formally, a *network schedule* Ω, for a control loop (P, K) is a sequence of transmissions between a pairs of nodes through some channels realizing the end-to-end communication between P and K. Therefore, Ω starts and ends with a transmission between P and its neighbor, and in between it must have transmissions via $\mathcal{C}$ over the network $\mathcal{G}$, since K gets executed on $\mathcal{C}$. E.g., according to Fig. 4.2 (shown in red color), the network schedule of (P_1, K_1) is: $[\langle P_1 \rightarrow v_5, c_1\rangle, \langle v_5 \rightarrow v_2, c_2\rangle, \langle v_2 \rightarrow C, c_1\rangle, \langle C \rightarrow v_1, c_2\rangle, \langle v_1 \rightarrow v_6, c_1\rangle, \langle v_6 \rightarrow P_1, c_2\rangle]$, where c_x is the default channel that is used between a pair of nodes when no collision occurs. If a node participates in the network schedules of two control loops at multiple distinct time steps, then its activity information will contain the record for each of these corresponding communication activities. Note that given

a network schedule Ω for a loop (P, K) having $l = |\Omega|$ consecutive transmissions, the *end-to-end timing* will be *guaranteed*, if $l \times T^{e}_{Slot}(M, \lambda) \leqslant h$, where h is the sampling period of P.

4.3.4. End-to-End Communication Reliability of a Loop

Wireless transmissions might fail if packets collide due to interference from neighboring networks. Hence, given a network schedule Ω of a control loop, we assign a reliability value $R_k, 1 \leqslant k \leqslant |\Omega|$ to each packet transmission between the pair of nodes in Ω. Assume that the k-th transmission is happening between a node pair (v_i, v_j) via some channel c. We define $TX_{(i,c)}$ as the probability of a successful transmission by v_i, and $RX_{(i,c)}$ as the probability of a successful reception and sending back acknowledgement by v_j via the channel c. Therefore, R_k can be defined as: $R_k = R_{(i,j,c)} = TX_{(i,c)} \times RX_{(j,c)}$, with the triplet (i, j, c) characterizing a *communication event*.

With this definition of R_k, the *end-to-end communication reliability* of the loop can be defined as the product of the individual reliabilities of each communication event between the node pairs participating in Ω, i.e., $R(\Omega) = \prod_{k=1}^{|\Omega|} R_k$. Now, if we have a requirement to maintain the communication reliability of that loop as R_{cl}, we have: $\prod_{k=1}^{|\Omega|} R_k \geqslant R_{cl}$. Let us consider that at some time instant, the second transmission in the schedule starts experiencing a high packet loss rate due to external interference, leading to a violation of the requirement. With p retransmission attempts, we obtain the communication reliability as $R_1 \cdot (1-(1-R_2)^p) \cdot \prod_{k=3}^{|\Omega|} R_k$. Since $(1-(1-R_2)^p) \geqslant R_2$, the resulting increment may be sufficient to satisfy R_{cl}. This motivates us to customize the number of retransmissions to increase the reliability.

4.4. Proposed Framework

This section describes the foundational details of the proposed AEW framework and it is summarized in Fig. 4.2.

4.4.1. Hierarchical Control Architecture.

At the heart of AEW, a two-level hierarchical control architecture is developed that does the following two operations during the execution of control loops over a multi-hop wireless network.

Level-1: At Network Layer. It performs a control theoretic analysis to calculate the number of retransmissions a node (participating in the network schedule of that control loop) has to do on facing transmission failures. We refer to this as the *Low-level Control Scheme* (LCS).

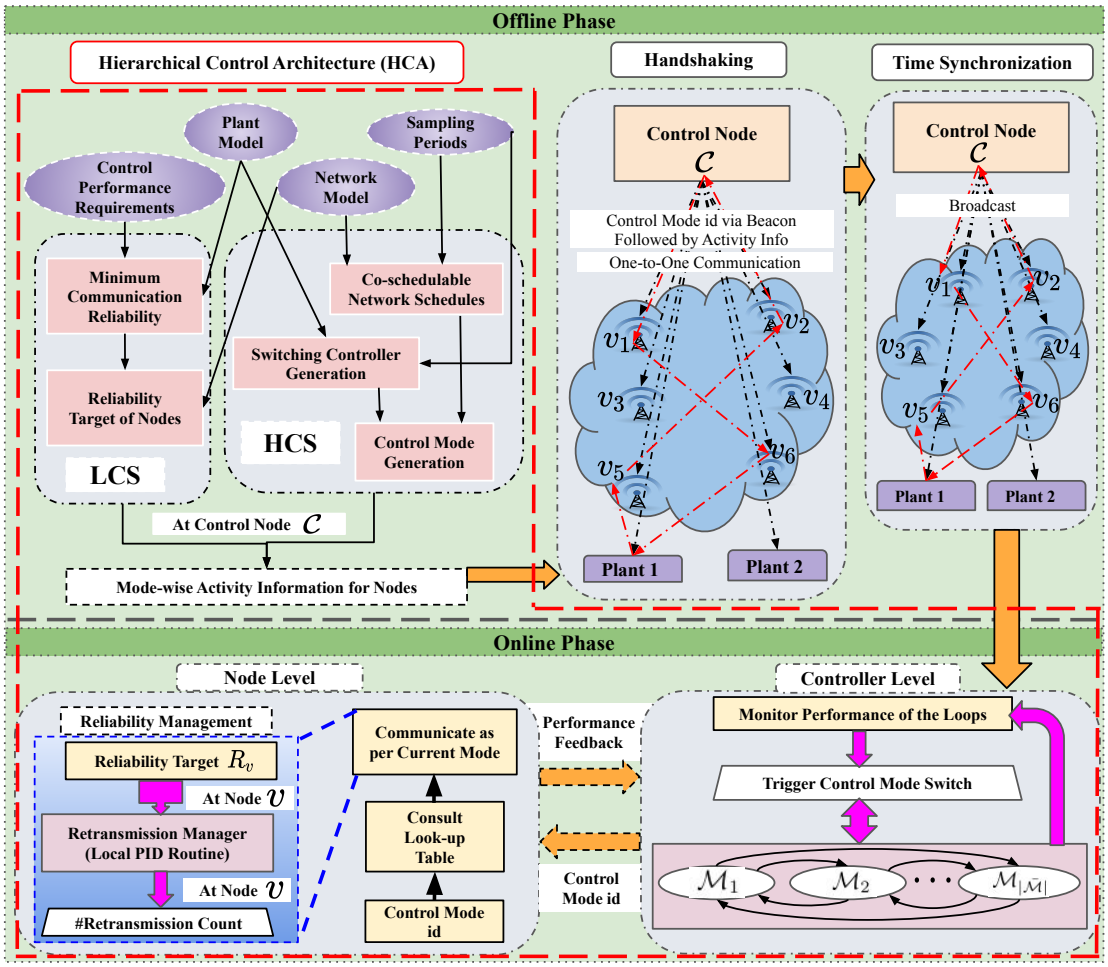

Figure 4.2: The proposed AEW framework.

Level-2: At Control Layer. It performs another control theoretic analysis to switch to a different controller for that loop if LCS at each of the participating nodes in the network schedule of that loop fails to handle the performance degradation. We refer to this as the *High-level Control Scheme* (HCS).

Details of Low-level Control Scheme

LCS has three main steps as outlined below, and details are given next to it.

1. For all the control loops $\{(P_i, K_i), i = 1, \cdots, n\}$, calculate a *minimum bound*, R_{cl_i}, on the end-to-end communication reliability of (P_i, K_i) from its control performance requirement τ_i (i.e., the settling time).
2. Compute a *reliability target*, R_v, to each node v participating in the network schedule of *any* control loop.
3. Generate a PID control routine to calculate the number of re-transmissions, U_v, required by a node v to maintain R_{cl_i} based on the deviation from R_v.

1. Calculating Minimum Bound: We follow the control theoretic approach presented in [27] to calculate the minimum execution rate, r_{min_i}, of the control

loop from τ_i, the sampling period h_i, and the dynamics matrices of P_i. r_{min_i} essentially says that, to meet τ_i the loop has to be executed minimum q times in l consecutive attempts, where $q = \lceil l \times r_{min_i} \rceil$. Now, the successful execution of a loop only can be achieved by successful packet transmission at each node participating in the respective network schedule. Hence, we can set the minimum bound $R_{cl_i} = r_{min_i}, i = 1, \cdots, n$.

2. Computing Reliability Target: The next step is to compute the *reliability target*, R_v, of each node v participating in the network schedule of a control loop. As defined in Sec. 4.3.4, the end-to-end communication reliability of the loop (P_i, K_i) is: $R(\Omega_i) = \Pi_{k=1}^{|\Omega_i|} R_k$. In order to satisfy the minimum end-to-end communication reliability, R_{cl_i}, of each control loop, following conditions have to be satisfied.

$$R(\Omega_i) = \Pi_{k=1}^{|\Omega_i|} R_k \geqslant R_{cl_i} \ \forall i = 1, \cdots, n. \tag{4.5}$$

Note that the network schedules for two or more control loops may share the node pair (v_i, v_j), using the same channel, c, at different time points, i.e., the same communication event (i, j, c) is active at the k-th transmission of Ω_1 and also at the k'-th transmission of Ω_2, where $k \neq k'$. In that scenario, we assign $R_{(i,j,c)} = max(R_k, R_{k'})$, where R_k and $R_{k'}$ can be computed using Eq. (4.5). Since $R_{(i,j,c)} = TX_{(i,c)} \times RX_{(j,c)}$ (ref. Section 4.3.4), this process helps in setting a *fixed* reliability target, R_{v_i} for each *sender* v_i participating in multiple communication events. As a result, we have one retransmission manager (i.e., the PID control routine described next) in each node with a reliability target. Implementing multiple such managers with different reliability targets for each node leads to an increased memory buffer requirement at that node, which is not possible with the low onboard memory support.

The list of transmissions in which a node v_i participates is given by, $C_{v_i} = \{(q, k) \,|\, \exists j, c \text{ s.t. } k\text{-th transmission of } \Omega_q \text{ has the communication event } (i, j, c)\}$. We solve the following optimization problem to assign R_{v_i} to each node $v_i \in \mathcal{V}$ such that successful transmission through these nodes guarantee the minimum communication reliability of all the control loops.

$$\text{Min.} \sum_{v_i \in \mathcal{V}} R_{v_i} \ \text{ s.t. Eq. (4.5) \& } R_{v_i} = max\{R_k \,|\, (q, k) \in C_{v_i}\}$$

Clearly, this optimization problem is nonlinear in nature and for the solution we use the state-of-the-art analytical technique of *sequential quadratic programming*, which is an iterative method for constrained nonlinear optimization.

3. Managing Retransmission Count: Next, we generate the PID control routine in each node which acts as a *retransmission manager* (cf. Fig. 4.2) at that node, as follows. From an implementation perspective, we consider a buffer B

storing the transmission status of the last N transmission instances of v. Let H be the amount of time taken to fill up N buffer entries and $S_N(B)$ be the number of successful transmissions. We compute the current reliability of v as $R = \frac{S_N(B)}{|B|}$, whereas R_v is the reliability target of v as computed in the previous step. Using these, we compute the error $\mathcal{E}(\delta) = (R_v - R)$, where $\mathcal{E}(\delta)$ denotes deviation error for the δ-th time with the interval of such updates being $H(\delta)$. The quantity $H(\delta)$ is time varying (hence, a function of δ) as it depends on amount of time taken for N transmissions by the participating loops. We consider the PID control law:

$$U(\delta+1) = k_p \times \mathcal{E}(\delta) + k_i \times \sum_{\Delta} \mathcal{E}(\delta) \times H(\delta) + k_d(\mathcal{E}(\delta) - \mathcal{E}(\delta-1))/H(\delta),$$

with k_p, k_i, k_d as usual PID parameters which are auto-tuned online following the Ziegler-Nichols tuning rule [72] and Δ being the window size for PID integral term. The control output U provides the number of retransmission attempts in each of the N subsequent transmissions by v. Note that consecutive retransmissions are carried out on different wireless channels using channel hopping, hence, with a growing number of retransmissions, the collision probability declines.

Details of High-level Control Scheme

As mentioned in Sec. 4.3.2, given a control loop (P, K) having h as its sampling period and Ω as its network schedule, to ensure its end-to-end timing guarantee, $|\Omega| \times T^e_{Slot}(M, \lambda) \leqslant h$. Therefore, the number of possible retransmission attempts in a node is limited by the slot-length T^e_{Slot} which is further limited by the sampling period for each control loop. Therefore, if the retransmission manager at each participating node in Ω fails to maintain the reliability requirement due to the above limitation, one solution would be switching to different controllers using higher sampling periods, thus allowing for bigger slot-length initiating more retransmissions for each node. The steps of HCS (cf. Fig. 4.2) are described next.

1. Designing Switching Controllers: For a plant P_i, we choose a set of sampling periods, $\mathcal{H}_i = \{h_{i,1} < h_{i,2} < \cdots h_{i,m_i}\}$, in a sorted order and the corresponding set of feedback controllers, $\mathcal{K}_i = \{K_{i,1}, K_{i,2}, \cdots, K_{i,m_i}\}$. Each $K_{i,j} \in \mathcal{K}_i$ for the period $h_{i,j} \in \mathcal{H}_i$ is designed guaranteeing the closed loop stability. Since in general, with the longer sampling period the control performance deteriorates, we keep the set $\mathcal{H}_i$ (as well as $\mathcal{K}_i$) very small having a few sampling periods for which the performance does not deteriorate significantly. The set $\mathcal{K}_i$ is sorted according to the performance of the controllers. In order to avoid stability issues while switching among controllers, we synthesize $\mathcal{K}_i$ ensuring the existence of common lyapunov functions [7] as discussed in [59].

2. Co-Schedulability of the Network Schedules: Note that with the increased slot-length obtained due to increment of sampling period of a loop, the co-schedulability of the network schedules of multiple loops may get hampered. E.g., let the node v participates in the network schedules Ω_1 and Ω_2 of the loops (P_1, K_1) and (P_2, K_2) respectively. It has two records, [`1, ON,` v'`, {`c_1, c_3`}, T`] and [`2, ON,` v''`, {`c_2, c_4`}, R`] in its activity information (ref. Sec. 4.3.3) corresponding to its participation in Ω_1 and Ω_2 respectively. Now, if T^e_{Slot} increases by a $2\times$ factor due to (P_1, K_1), then at the local time `2`, v can not receive from v'' since it still busy in sending to v'. This may violate the end-to-end timing guarantee of (P_2, K_2). Hence, to maintain the co-schedulability, we need to redesign the network schedules of all the control loops when we allow changes in the sampling period of any control loop. We leverage the technique provided in [27] to synthesize the co-schedulable network schedules of the control loops for a given choice of their sampling periods and the network model.

3. Defining Control Mode of the Network: Given a multi-hop wireless CPS, we define its p-th *control mode*, $\mathcal{M}_p$, as a tuple, $\langle (K^p_{1,j_1}, K^p_{2,j_2}, \cdots, K^p_{n,j_n}), \bar{\Omega}^p \rangle$, where $K^p_{i,j_i} = K_{i,j} \in \mathcal{K}_i$ is the current active controller of i-th loop and $\bar{\Omega}^p = \langle \Omega^p_1, \Omega^p_2, \cdots, \Omega^p_n \rangle$ contains the network schedules of n loops at that mode which are co-schedulable. There are total $\bar{\mathcal{M}}$ such control modes where $\bar{\mathcal{M}} \subseteq \mathcal{K}_1 \times \mathcal{K}_2 \cdots \times \mathcal{K}_n$. If for some choices of sampling periods of the loops, a co-schedulable combination of network schedules may not exist, so we ignore those choices. $\bar{\mathcal{M}}$ is ordered lexicographically with increasing order of sampling periods of the loops. Since we choose $\mathcal{K}_i, \forall i = 1, ..., n$ be very small (ref. Sec. 4.4.1-1), we restrict $\bar{\mathcal{M}}$ not to be very large.

4. Switching of Control Modes: The initial/primary control mode is $\mathcal{M}_1 = \langle (K^1_{1,j_1}, K^1_{2,j_2}, \cdots, K^1_{n,j_n}), \bar{\Omega}^1 \rangle$ where $K^1_{i,j_i} = K_{i,1} \in \mathcal{K}_i, \forall i = 1, \cdots, n$, i.e., controller designed with the smallest sampling period $h_{i,1}, \forall i = 1, ..., n$. Let for the k-th loop, the retransmission managers at participating nodes in $\Omega^1_k \in \bar{\Omega}^1$ fail to maintain the reliability requirement R_{cl_k} at runtime. Then we allow adaptive switching to the control mode $\mathcal{M}_p = \langle (K^p_{1,j_1}, K^p_{2,j_2}, \cdots, K^p_{n,j_n}), \bar{\Omega}^p \rangle$ with $K^p_{i,j_i} = K_{i,1}$ for $i \neq k$ and $K^p_{i,j_i} = K_{i,2}$ for $i = k$, i.e, for k-th loop, switch to the controller designed with the next higher period, i.e, $h_{k,2}$. Once the performance degradation is under control with $\mathcal{M}_p$, control mode switches back to $\mathcal{M}_1$.

Offline and Online Phases

We partition the operation of the HCA into *offline* and *online* phases. The first two steps of LCS are done offline beforehand since for these steps we only need the settling time requirements of the loops and which are known a priory. Thereby, we avoid solving the optimization problem (Step.2 of LCS) on-the-fly. Step.3 of

LCS is an online operation invoked when the reliability requirement of a loop is not getting met due to packet drops. The first three steps of HCS, i.e., up to generating all possible control modes, are done offline and distributed among all the nodes in the network at deployment time (ref. Sec. 4.4.2). Each node then aware of its activity under each such control mode *a priory*. At runtime, based on the reliability/performance degradation of loops, the current control mode is selected adaptively by the control node $\mathcal{C}$ (following Step.4 of HCS), and the entire system runs accordingly. This approach thus ensures the *runtime adaptability* requirement of CPS. It also ensures very little protocol communication overhead at runtime, which is highly importance in any low-power implementation.

4.4.2. Handshaking and Time Synchronization

Following HCA, once the target reliability of each node (Step. 1-2 of LCS) and all the control modes are computed (Step. 1-3 of HCS) offline, it is required to distribute these among the nodes in the network at the deployment time. This is done via *handshaking* which is an offline process enabling a minimal communication overhead at runtime. All the offline computations are done in the control node $\mathcal{C}$ which is also responsible for the handshaking. But before handshaking, $\mathcal{C}$ generates the set of activity information, $\{\mathcal{A}_v^i\}_{i=1}^{|\bar{\mathcal{M}}|}$, for each node v based on its participation in the network schedules of each the $|\bar{\mathcal{M}}|$ control modes. During handshaking, a *one-to-one communication* is performed between each node and the node $\mathcal{C}$ where $\mathcal{C}$ first sends the *id* of current control mode through a *beacon* to the node v, and then it sends the *payload* having $\{\mathcal{A}_v^i\}_{i=1}^{|\bar{\mathcal{M}}|}$. Based on this information, v then creates a look-up table in its local memory and communicates accordingly on-the-fly based on the current control mode. At runtime, if $\mathcal{C}$ decides to change the control mode to handle the performance degradation of some loops, it just sends a beacon to all nodes indicating the id of the current control mode.

To eliminate unnecessary delays and achieve high reliability and efficiency, network-wide *time synchronisation* is required. Hence, after handshaking, through time synchronisation a consistent notion of time is provided over the entire network. It is achieved by broadcasting the time at the control node $\mathcal{C}$ to all the other nodes in the network before switching over to the online phase.

4.5. Experimental Setup and Evaluation

Realistic testbeds are essential for evaluation and validation of real-time networking solutions in cyber-physical systems [11, 42].

4.5.1. Testbed Details

plase mention how the testbed of chapter 2 ha

To show the effectiveness of the proposed framework, we adopting the hardware setup demonstrated in [30]. The testbed is deployed across two adjacent rooms (6 m × 12 m) as depicted in Fig. 4.3. It consists of total 12 nodes including two real Double Integrator Circuits (DICs) as physical plants (with attached transceiver), nine (battery powered) intermediate nodes, and a control node. Control and intermediate wireless nodes have functions as defined earlier. They are built using Arduino UNO boards with attached transceiver chips (nRF24L01).

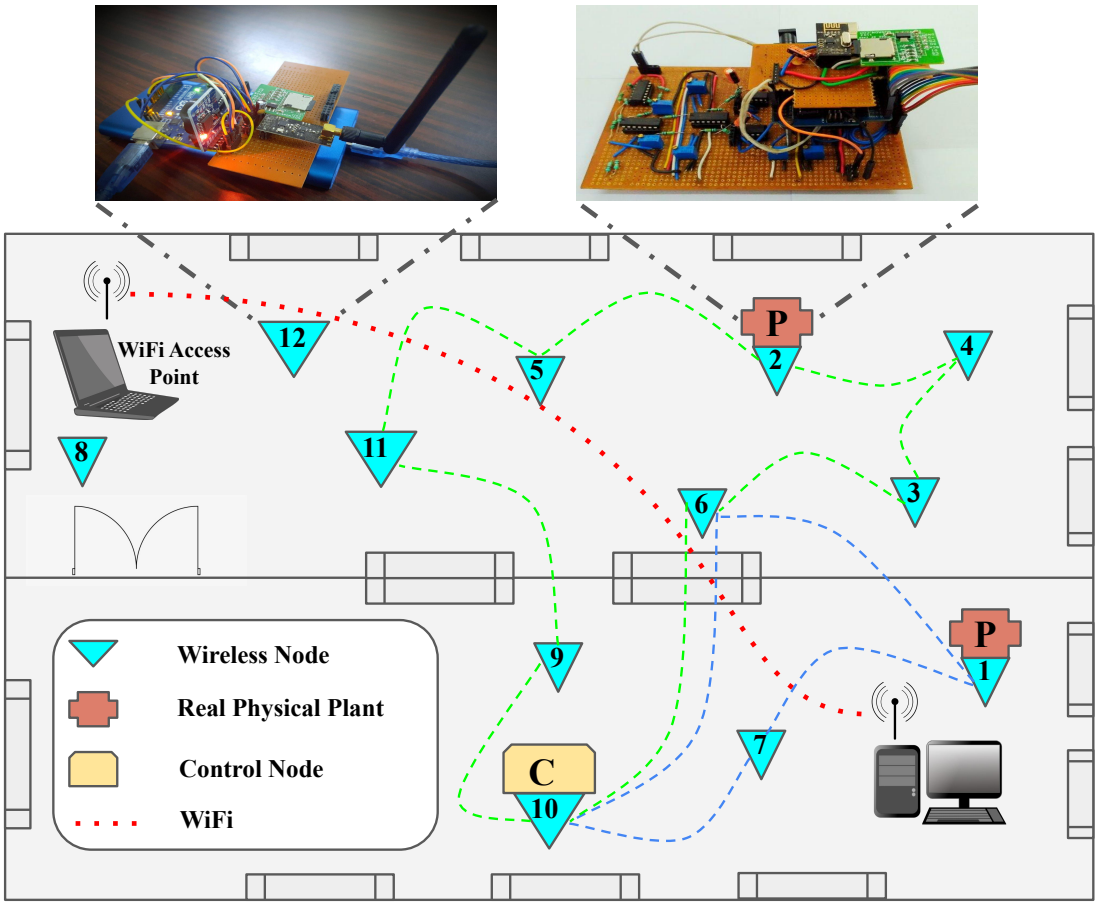

Figure 4.3: Coexistence testbed setup deployed across two rooms (6 m × 12 m): Two real systems (P) are stabilized over a multi-hop wireless network of twelve embedded devices under real-world WiFi noise.

Also mention all the changes and upgrad

Further, we consider a WiFi (*IEEE* 802.11) network (ref. Fig. 4.3) coexisting with the testbed. This setup is used to evaluate the advantage of AEW under real-world wireless uncertainties, such as, transmission failure due to interference produced by the co-located network.

4.5.2. Details of the Experiment

We conduct three experiments in our CPS testbed to report the gain (i.e., %-improvement) obtained by AEW in terms of 1) reducing transmission failure ratio (hence, improving reliability), 2) control performance improvement, and 3) energy savings w.r.t standard state-of-the-art techniques.

We conduct each of the experiments for 15 s, and in each experiment, we first set the initial output voltage of both the DIC plants to 0 V. For both the DICs the reference value is considered as 3 V (including offset). The control objective is to bring back output voltage under the 2% envelope around 3 V within 4 s (the *settling time* requirement). During experiments, we fix the use of channels for both the WiFi and our network so that channels can overlap and interference can happen. Finally, we use IPERF3 application to generate sudden traffic on the WiFi to generate real-world external interference.

For these plants, we consider the set of sampling periods as $\mathcal{H}_1 = \{h_{1,1} = 40\,\text{ms}, h_{1,2} = 80\,\text{ms}\}$ and $\mathcal{H}_2 = \{h_{2,1} = 80\,\text{ms}, h_{2,2} = 100\,\text{ms}\}$ respectively. Then, we design set of stabilizing controllers, $\mathcal{K}_1 = \{K_{1,1}, K_{1,2}\}$ and $\mathcal{K}_2 = \{K_{2,1}, K_{2,2}\}$, respectively, using pole-placement technique with the choice of sampling periods in $\mathcal{H}_1$ and $\mathcal{H}_2$ (ref. Sec. 4.4.1-1). Next, we compute the minimum end-to-end communication reliability of these two loops as $R_{cl_1} = \{0.6, 0.65\}$ and $R_{cl_2} = \{0.65, 0.7\}$ for the sampling periods in $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively (ref. Sec. 4.4.1-1). With $\mathcal{H}_1$ and $\mathcal{H}_2$, we get 4 possible control modes where the initial control mode is $\mathcal{M}_1 = \langle (K_{1,1}, K_{2,1}), \bar{\Omega}^1) \rangle$. Note that $K_{1,1}$ and $K_{2,1}$ are designed with $h_{1,1} = 40\,\text{ms}$ and $h_{2,1} = 80\,\text{ms}$ respectively. The network schedules for the loops in $\bar{\Omega}^1$ are marked by blue and green color in Fig. 4.3 respectively. As mentioned in Sec. 4.4.1, all these 4 control modes statically pre-computed at $\mathcal{C}$ which then deploys the activity information $\{\mathcal{A}_v^i\}_{i=1}^4$ to each node v based on its participation in $\bar{\Omega}^i, i = 1, 2, 3, 4$.

4.5.3. Experimental Results

During the experiment session of 15 s, the WiFi interference causes highest transmission failure within the time window of 1 s-3 s as shown in Fig. 4.4.

Elsewhere, it affects moderately, hence, the node-level PID routines control their relative transmission failures by regulating the number of retransmissions while staying at the initial control mode $\mathcal{M}_1$. The bar chart in Fig. 4.4 shows the average retransmissions calculated by node-level PID routines. However, in the window of the high noisy scenario, $\mathcal{C}$ finds the performance degradation in loop 1 deviating from the 2 % band of 3 V. Hence, it sends the signal to change the mode to $\mathcal{M}_2$. Note that, in $\mathcal{M}_2 = \langle (K_{1,2}, K_{2,1}), \bar{\Omega}^2) \rangle$, the active controller for loop 1 is $K_{1,2}$ which is designed with $h_{1,2} = 80\,\text{ms}$, thereby, doubling the slot

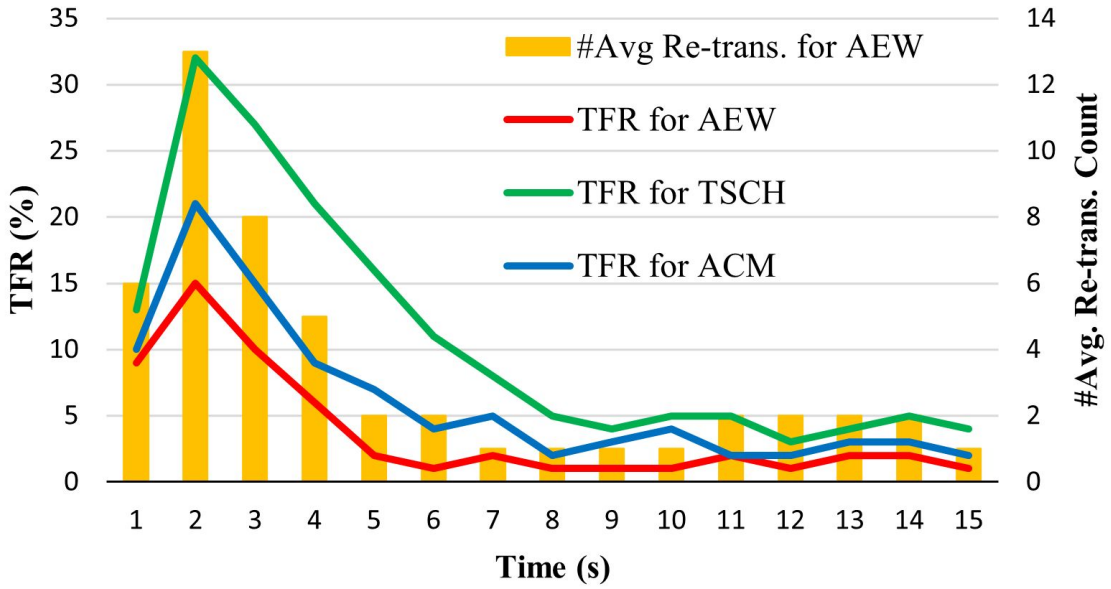

Figure 4.4: Improvement in reducing transmission failure ratio (TFR).

length, and hence, allowing more retransmissions to cope with the performance degradation. With AEW, we get much lesser transmission failure ratio (the red curve in Fig. 4.4), hence an improved reliability of 53 % and 24 % as compared with the existing techniques TSCH [31] (the green curve), and ACM [19] (the blue curve) respectively under the identical external interference caused by the WiFi.

The efficacy of AEW can further be appreciated when we compare our results with PFS [30] in terms of improvement in control performance. The red curve in Fig. 4.5 depicts the output response of plant 1 obtained following AEW, while the blue curve is obtained by applying PFS [30]. We observe nearly 24 % improvement in settling time by applying AEW.

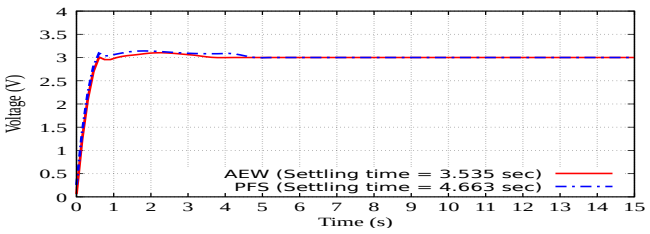

Figure 4.5: Improvement in control performance.

Next, we compare AEW with the flooding-based Glossy protocol which has also been implemented in our setup for a comparative study. Fig. 4.6 highlights the relative energy savings achieved by AEW under the WiFi interference scenario as reported in Fig. 4.4. This is because, AEW allows the nodes to switch to the relatively low-power consuming state, i.e., *standby state* in two situations; 1) when they are not operating in any time-slot as per their activity information (ref. Sec. 4.3.2) and 2) after a certain number of retransmissions, λ_A, as calculated by respective PID control routine. In contrast, in Glossy, due to network-wide

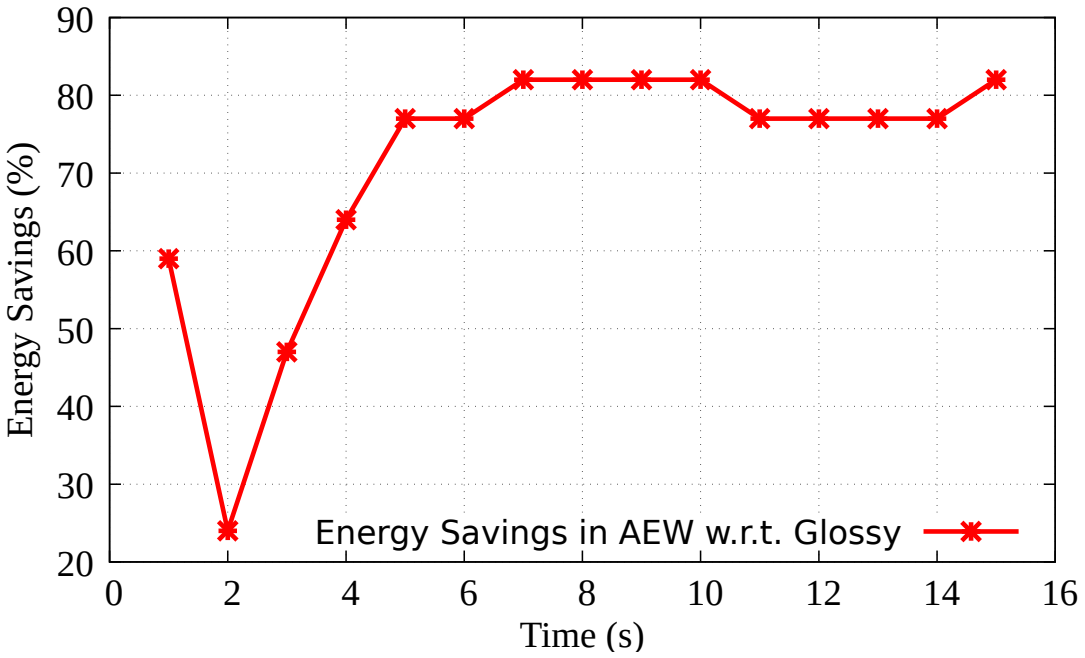

Figure 4.6: Relative energy saving scenario.

flooding through parallel channels, most of nodes remain active in all the time-slots, and hence, stay at the *start state* with relatively higher power consumption. Moreover, Glossy allows a fixed number of retransmissions λ_G all the time, while with AEW we customize the number of retransmissions based on requirement. This results in slot-length with shorter on-state communication time in our case leading to a lower energy consumption. Hence, following Eq. (4.4), we calculate the relative energy saving as,

$$E^{sv}_{Slot} = [T_{Slot}(M, \lambda_G) - T^{e}_{Slot}(M, \lambda_A)]/T_{Slot}(M, \lambda_G).$$

Now following E^{sv}_{Slot} and the energy profile given in Fig. 4.1, we get relative energy savings as reported in Fig. 4.6. Note that in Fig. 4.6 within the time interval of 1 s-3 s, we get lowest energy savings. This is because, during this time window, AEW triggers switching to control mode $\mathcal{M}_2$ allowing more retransmissions as compared to control mode $\mathcal{M}_1$, thereby, leading to relatively more power consumption by the participating nodes. Even comparing with this worst-case scenario, we get 43 % energy savings on average w.r.t Glossy. AEW offers maximum energy savings if it always operates staying at the initial control mode $\mathcal{M}_1$.

also mention the details of the exp. setup that are under comment mode in our paper. due to lack of spa

i5-2400 machine with 8 GB of memory. Plant output waveforms and CSV files have been taken through RIGOL DS1102E Digital Oscilloscope. We have used the MATLAB(x64) version R2020b for control theoretic and other calculations.

write conclusion and remarks

Chapter 5
Conclusion

Start with the following text:This chapter primarily summar

While the scalability of formal techniques is unlikely to improve very soon, there is a possibility that robustness guarantees of control software implementations can be validated against platform level faults using a well designed, programmable fault insertion methodology. From the education perspective, each of the experiments demonstrated with our developed testbed provides opportunities for imbibing students with the salient features of CPS design. Further, in the research context, our testbed is well equipped to handle physical plants as well as real-time simulation of plants using Hardware-in-loop simulators. We are working towards creating a more generalized version of our testbed interfaced with such real-time simulators so that our MCN test and validation capabilities can be extended to a Wide Area CPS domain. In this thesis, a framework is presented using which high-performance controllers can be implemented over wireless multi-hop networks, mitigating the effect of large and variable delays. According to the proposed strategy, the control algorithm proactively computes future control inputs based on the current state measurement for different possible delay values. These inputs are packed in a data packet and sent to the actuator. The actuator at the plant side sees which delay was experienced and uses the most appropriate control input from the vector to actuate. A novel performance- and energy-aware adaptive control architecture is also provided as a low power alternative to flooding-based techniques for reliable wireless control of fast feedback systems. Synergizing control-theoretic analysis with real-time network packet scheduling provides us with control-theoretic stability guarantees about the implementation, which is also validated by our testbed setup for the work. In future work, the optimal trade-off between the number of control inputs per packet for maximizing the control performance and minimizing the collision probability needs to be studied. Another interesting future work could be the realization of real industrial systems in the direction of the proposed approach, based on the transmission capabilities of emerging wireless protocols, e.g., 5G.

For each work do the following: First write the conclusionThen mention the assumption/limitat

Bibliography

[1] `https://www5.epsondevice.com/en/information/technical_info/pdf/wp_e20130918_rtc.pdf`. [*Cited on page 38.*]

[2] `http://www.kerrywong.com/2014/07/19/ds3232-clock-frequency-calibration/`. [*Cited on page 38.*]

[3] G. Alldredge, M. S. Branicky, and V. Liberatore. Play-back buffers in networked control systems: Evaluation and design. In *American Control Conference (ACC)*, 2008. [*Cited on page 23.*]

[4] R. Alur, A. D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss. Compositional modeling and analysis of multi-hop control networks. *IEEE Transactions on Automatic Control*, 56(10):2345–2357, Oct 2011. [*Cited on page 2.*]

[5] R. Alur et al. Modeling and analysis of multi-hop control networks. In *Proc. RTAS*, pages 223–232, 2009. [*Cited on pages 3 and 24.*]

[6] R. Alur et al. Compositional modeling and analysis of multi-hop control networks. *IEEE TAC*, 56(10):2345–2357, Oct 2011. [*Cited on pages 23 and 24.*]

[7] K. J. Åström and B. Wittenmark. *Computer-controlled systems*. Prentice-Hall, Inc., 1997. [*Cited on pages 18, 20, 26, 36, and 51.*]

[8] J. Bai, E. P. Eyisi, F. Qiu, Y. Xue, and X. D. Koutsoukos. Optimal cross-layer design of sampling rate adaptation and network scheduling for wireless networked control systems. In *Proc. ICCPS*, pages 107–116, 2012. [*Cited on pages 23, 24, and 40.*]

[9] M. Balszun, D. Roy, L. Zhang, W. Chang, and S. Chakraborty. Effectively utilizing elastic resources in networked control systems. In *Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017. [*Cited on pages 5 and 24.*]

[10] D. Baumann, F. Mager, R. Jacob, L. Thiele, M. Zimmerling, and S. Trimpe. Fast feedback control over multi-hop wireless networks with mode changes

and stability guarantees. *ACM Transaction on Cyber-Physical Systems)*, 4(2), 2019. [*Cited on page 39.*]

[11] D. Baumann, F. Mager, H. Singh, M. Zimmerling, and S. Trimpe. Evaluating low-power wireless cyber-physical systems. *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, Apr 2018. [*Cited on page 53.*]

[12] E. Bini and A. Cervin. Delay-aware period assignment in control systems. In *Real-Time Systems Symposium (RTSS)*, 2008. [*Cited on page 4.*]

[13] D. Carlson, M. Chang, A. Terzis, Y. Chen, and O. Gnawali. Forwarder selection in multi-transmitter networks. In *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 1–10, 2013. [*Cited on page 44.*]

[14] W. Chang, L. Zhang, D. Roy, and S. Chakraborty. *Control/architecture codesign for cyber-physical systems.* Handbook of Hardware/Software Codesign. Springer Netherlands, 2017. [*Cited on page 23.*]

[15] C. Chen. *System and signal analysis.* The Oxford Series in Electrical and Computer Engineering,Oxford University Press, UK, 1994. [*Cited on page 26.*]

[16] L. S. Committee et al. Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE-SA Standards Board*, 2003. [*Cited on page 4.*]

[17] A. D'Innocenzo, M. D. Di Benedetto, and othres. Fault tolerant control of multi-hop control networks. *IEEE Trans. on Automatic Control*, 58(6):1377–1389, 2013. [*Cited on page 24.*]

[18] A. Escobar, C. Gonzalez, F. J. Cruz, J. Garcia-Jimenez, J. Klaue, and A. Corona. Redfixhop: Efficient ultra-low-latency network flooding. In *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–2, 2016. [*Cited on page 44.*]

[19] M. Feldman, G. Cainelli, G. Kunzel, I. Muller, and C. E. Pereira. Adaptive channel map for time slotted channel hopping industrial wireless networks. *IFAC-PapersOnLine*, 53(2):8237–8242, 2020. [*Cited on pages 44 and 56.*]

[20] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 73–84, 2011. [*Cited on pages 5 and 44.*]

[21] G. Fiore, V. Ercoli, A. J. Isaksson, K. Landernäs, and M. D. D. Benedetto. Multihop multi-channel scheduling for wireless control in wirelesshart networks. In *Emerging Technologies And Factory Automation*, pages 1–8, Sept 2009. [*Cited on page 1.*]

[22] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003. [*Cited on page 38.*]

[23] H. Gao and T. Chen. New results on stability of discrete-time systems with time-varying state delay. *IEEE Transactions on Automatic Control*, 52(2):328–334, 2007. [*Cited on pages 5 and 23.*]

[24] P. Garcia, P. Castillo, R. Lozano, and P. Albertos. Robustness with respect to delay uncertainties of a predictor-observer based discrete-time controller. In *Conference on Decision and Control (CDC)*, 2006. [*Cited on page 23.*]

[25] P. Garcia, A. Gonzalez, P. Castillo, R. Lozano, and P. Albertos. Robustness of a discrete-time predictor-based controller for time-varying measurement delay. *Control Engineering Practice*, 20(2):102 – 110, 2012. [*Cited on pages 5, 23, and 32.*]

[26] S. Ghosh, S. Dey, and P. Dasgupta. Co-synthesis of loop execution patterns for multihop control networks. *IEEE Embedded System Letters*, 10(4):111–114, Dec 2018. [*Cited on pages 23 and 24.*]

[27] S. Ghosh, S. Dey, and P. Dasgupta. Pattern guided integrated scheduling and routing in multi-hop control networks. *ACM Transaction on Embedded Computing Systems*, 19(2), 2020. [*Cited on pages 5, 24, 49, and 52.*]

[28] S. Ghosh et al. A structured methodology for pattern based adaptive scheduling in embedded control. *ACM TECS*, 16(5s):189:1–189:22, Sept. 2017. [*Cited on page 35.*]

[29] S. Ghosh, A. Mondal, P. H. Kindt, P. Sharma, Y. Agarwal, S. Dey, A. K. Deb, and S. Chakraborty. A programmable open architecture testbed for cps education. *IEEE Design & Test*, 37(6):31–38, 2020. [*Cited on page 35.*]

[30] S. Ghosh, A. Mondal, D. Roy, P. H. Kindt, S. Dey, and S. Chakraborty. Proactive feedback for networked cps. In *Proc. SAC*, 2021. [*Cited on pages 43, 45, 54, and 56.*]

[31] P. H. Gomes, T. Watteyne, P. Gosh, and B. Krishnamachari. Competition: Reliability through timeslotted channel hopping and flooding-based routing.

In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, EWSN '16, page 297–298, USA, 2016. Junction Publishing. [*Cited on pages 44 and 56.*]

[32] D. Goswami, R. Schneider, and S. Chakraborty. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2011. [*Cited on pages 23 and 25.*]

[33] D. Goswami, R. Schneider, and S. Chakraborty. Re-engineering cyber-physical control applications for hybrid communication protocols. In *Design, Automation & Test in Europe (DATE)*, 2011. [*Cited on pages 23 and 24.*]

[34] D. Goswami, R. Schneider, and S. Chakraborty. Relaxing signal delay constraints in distributed embedded controllers. *IEEE Transactions on Control Systems Technology*, 22(6):2337–2345, 2014. [*Cited on pages 23 and 24.*]

[35] J. Gu, Y. Xu, Y. Zhu, F. Mei, L. Kang, and Y. Liu. An improved fast transform algorithm based on adaptive hybrid channel allocation in tsch networks. *EURASIP Journal on Wireless Communications and Networking*, 2020:1–16, 2020. [*Cited on page 44.*]

[36] M. Hashemi, W. Si, M. Laifenfeld, D. Starobinski, and A. Trachtenberg. Intra-car multihop wireless sensor networking: a case study. *IEEE Communications Magazine*, 52(12):183–191, 2014. [*Cited on page 2.*]

[37] R. Jacob, L. Zhang, M. Zimmerling, J. Beutel, S. Chakraborty, and L. Thiele. The time-triggered wireless architecture. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2020. [*Cited on pages 23 and 24.*]

[38] R. Jacob, L. Zhang, M. Zimmerling, J. Beutel, S. Chakraborty, and L. Thiele. The time-triggered wireless architecture. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2020. [*Cited on pages 43 and 44.*]

[39] R. Jacob, M. Zimmerling, P. Huang, J. Beutel, and L. Thiele. End-to-end real-time guarantees in wireless cyber-physical systems. In *Proc. RTSS*, pages 167–178, Nov 2016. [*Cited on pages 1, 2, and 3.*]

[40] X. Jin, A. Saifullah, C. Lu, and P. Zeng. Real-time scheduling for event-triggered and time-triggered flows in industrial wireless sensor-actuator networks. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1684–1692, April 2019. [*Cited on pages 1 and 3.*]

[41] lastinuteengineers.com. Interface DS3231: Precision RTC module with Arduino. `https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial`, 2020. [*Cited on page 35.*]

[42] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proceedings of the IEEE*, 104(5):1013–1024, May 2016. [*Cited on page 53.*]

[43] Y. Ma, D. Gunatilaka, B. Li, H. Gonzalez, and C. Lu. Holistic cyber-physical management for dependable wireless control systems. *ACM TCPS*, 3(1):3:1–3:25, Sept. 2018. [*Cited on pages 23, 24, and 25.*]

[44] F. Mager, D. Baumann, R. Jacob, L. Thiele, S. Trimpe, and M. Zimmerling. Feedback control goes wireless: Guaranteed stability over low-power multi-hop networks. pages 97–108, 2019. [*Cited on pages 2, 5, 23, 24, 25, 43, and 45.*]

[45] R. Mangharam and M. Pajic. Distributed control for cyber-physical systems. *Journal of the Indian Institute of Science*, 93(3):353–387, 2013. [*Cited on page 40.*]

[46] P. Marwedel. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Springer, 2011. [*Cited on page 3.*]

[47] A. N. Kim, F. Hekland, S. Petersen, and P. Doyle. When hart goes wireless: Understanding and implementing the wirelesshart standard. pages 899 – 907, 10 2008. [*Cited on page 4.*]

[48] Nordic Semiconductor. nRF Connect SDK. `https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.0.0/nrf/ug_esb.html`, 2019. [Online; accessed 11-Nov-2019]. [*Cited on page 36.*]

[49] Nordic Semiconductor. nRF24 series. `https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF24-series`, 2019. [Online; accessed 11-Nov-2019]. [*Cited on pages 7, 12, and 35.*]

[50] G. J. Pappas. Wireless control networks: Modeling, synthesis, robustness,security. In *Hybrid Systems: Computation and Control*, pages 1–2, 2011. [*Cited on page 1.*]

[51] J. W. Ro, P. Roop, and A. Malik. Schedule synthesis for time-triggered multi-hop wireless networks with retransmissions. In *Proc. ISORC*, pages 94–101, April 2015. [*Cited on page 44.*]

[52] D. Roy, W. Chang, S. K. Mitter, and S. Chakraborty. Tighter dimensioning of heterogeneous multi-resource autonomous cps with control performance guarantees. In *Design Automation Conference (DAC)*, 2019. [*Cited on pages 5 and 24.*]

[53] D. Roy, S. Ghosh, Q. Zhu, M. Caccamo, and S. Chakraborty. Goodspread: Criticality-aware static scheduling of cps with multi-qos resources. In *Real-Time Systems Symposium (RTSS)*, 2020. [*Cited on page 5.*]

[54] D. Roy, C. Hobbs, J. Anderson, M. Caccamo, and S. Chakraborty. Timing debugging for cyber-physical systems. In *Design, Automation and Test in Europe (DATE)*, 2021. [*Cited on page 4.*]

[55] D. Roy, L. Zhang, W. Chang, D. Goswami, and S. Chakraborty. Multi-objective co-optimization of flexray-based distributed control systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016. [*Cited on page 27.*]

[56] D. Roy, L. Zhang, W. Chang, S. K. Mitter, and S. Chakraborty. Semantics-preserving cosynthesis of cyber-physical systems. *Proceedings of the IEEE*, 106(1):171–200, 2018. [*Cited on page 23.*]

[57] A. Saifullah, C. Wu, P. B. Tiwari, Y. Xu, Y. Fu, C. Lu, and Y. Chen. Near optimal rate selection for wireless control systems. *ACM Transactions on Embedded Computing Systems*, 13(4s):128:1–128:25, 2014. [*Cited on pages 4, 23, and 24.*]

[58] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. End-to-end communication delay analysis in wirelesshart networks. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2011. [*Cited on pages 23 and 24.*]

[59] M. Schinkel, Wen-Hua Chen, and A. Rantzer. Optimal control for systems with varying sampling rate. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, volume 4, pages 2979–2984 vol.4, May 2002. [*Cited on page 51.*]

[60] R. Schneider, D. Goswami, S. Zafar, S. Chakraborty, and M. Lukasiewycz. Constraint-driven synthesis and tool-support for flexray-based automotive control systems. In *International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS)*, 2011. [*Cited on pages 23 and 25.*]

[61] M. Schuß, C. A. Boano, M. Weber, and K. Römer. A competition to push the dependability of low-power wireless protocols to the edge. In *In Proc.*

International Conference on Embedded Wireless Systems and Networks, page 54–65, 2017. [*Cited on page 5.*]

[62] L. Shi, L. Xie, and R. M. Murray. Kalman filtering over a packet-delaying network: A probabilistic approach. *Automatica*, 45(9):2134 – 2140, 2009. [*Cited on pages 5 and 23.*]

[63] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control (TAC)*, 49(9):1453–1464, 2004. [*Cited on pages 5 and 23.*]

[64] J. Song et al. Wirelesshart: Applying wireless technology in real-time industrial process control. In *Proc. RTAS*, pages 377–386, 2008. [*Cited on pages 4 and 5.*]

[65] J. Stankovic, J. Sturges, and J. Eisenberg. A 21st century cyber-physical systems education. *ACM Computer Magazine*, 50(12), 2017. [*Cited on page 2.*]

[66] M. Törngren et al. Education and training challenges in the era of cyber-physical systems: beyond traditional engineering. *Workshop on Embedded and Cyber-Physical Systems Education*, 2015. [*Cited on page 2.*]

[67] H. Voit, A. Annaswamy, R. Schneider, D. Goswami, and S. Chakraborty. Adaptive switching controllers for systems with hybrid communication protocols. In *American Control Conference (ACC)*, 2012. [*Cited on pages 23 and 24.*]

[68] W. Wang, D. Mosse, D. Cole, and J. G. Pickel. Dynamic wireless network reconfiguration for control system applied to a nuclear reactor case study. In *International Conference on Real-Time Networks and Systems (RTNS)*, 2018. [*Cited on page 40.*]

[69] G. Weiss, A. D'Innocenzo, R. Alur, K. H. Johansson, and G. J. Pappas. Robust stability of multi-hop control networks. In *Conference on Decision and Control held jointly with Chinese Control Conference*, pages 2210–2215, Dec 2009. [*Cited on page 24.*]

[70] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, and D. Raychaudhuri. Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1270–1278, April 2019. [*Cited on page 1.*]

[71] X. Zhou, X. Gou, T. Huang, and S. Yang. Review on testing of cyber physical systems: Methods and testbeds. *IEEE Access*, 6:52179–52194, 2018. [*Cited on page 12.*]

[72] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Journal of Dynamic Systems, Measurement, and Control*, 115(2B):220–222, jun 1993. [*Cited on page 51.*]

[73] M. Zimmerling, L. Mottola, and S. Santini. Synchronous transmissions in low-power wireless: A survey of communication protocols and network services. 53(6), Dec. 2020. [*Cited on page 5.*]

Publications From This Thesis

Conference Papers

1. Sumana Ghosh, Arnab Mondal, Debayan Roy, Philipp H. Kindt, Soumyajit Dey, and Samarjit Chakraborty. 2021. Proactive feedback for networked CPS. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing (SAC '21)*. Association for Computing Machinery, New York, NY, USA, 164–173. DOI:https://doi.org/10.1145/3412841.3441897
2. Arnab Mondal, Sumana Ghosh, Philipp H. Kindt, Soumyajit Dey, Alok Kanti Deb, and Samarjit Chakraborty. AEW: Adaptively-Controlled Energy-Efficient Wireless CPS. *(Submitted)*

Journal Papers

Sumana Ghosh, Arnab Mondal, Philipp H. Kindt, Prateek Sharma, Yash Agarwal, Soumyajit Dey, Alok Kanti Deb, and Samarjit Chakraborty, *"A Programmable Open Architecture Testbed for CPS Education"*, IEEE Design & Test, vol. 37, no. 6, pp. 31-38, Dec. 2020, doi: 10.1109/MDAT.2020.3006798.