To appear in 36th ACM/SIGAPP Symposium On Applied Computing (SAC), CPS Track, 2021

# Proactive Feedback for Networked CPS

Sumana Ghosh
sumana.ghosh@tum.de
TU Munich, Germany

Arnab Mondal
amondal23@iitkgp.ac.in
IIT Kharagpur, India

Debayan Roy
debayan.roy@tum.de
TU Munich, Germany

Philipp H. Kindt
philipp.kindt@tum.de
TU Munich, Germany

Soumyajit Dey
soumya@cse.iitkgp.ac.in
IIT Kharagpur, India

Samarjit Chakraborty
samarjit@cs.unc.edu
UNC Chapel Hill, USA

## ABSTRACT

While wired networks provide a reliable platform for networked cyber-physical systems (CPS), there is an increasing demand 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. This is typically accounted for by designing controllers for the worst-case end-to-end delay. This guarantees stability also when the largest possible delay occurs. However, the delays observed during normal operation are significantly below the worst-case. As a result of the overly pessimistic controller design, the control performance becomes unnecessarily low. In this work, for the first time, we present a generic technique to handle varying end-to-end delays in wireless CPS.

While maintaining a stable operation, our technique preserves a high control performance. In essence, we propose a proactive feedback strategy that computes future control inputs for different possible delays a priori and sends them to the actuator in a single packet. When new control inputs are delayed, pre-computed ones accounting for higher delays are applied at appropriate actuation instants. In this way, a controller responds fast when control input arrives with low latencies, while adaptively acting more conservatively when packets are delayed. 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.

**ACM Reference Format:**
Sumana Ghosh, Arnab Mondal, Debayan Roy, Philipp H. Kindt, Soumyajit Dey, and Samarjit Chakraborty. 2021. Proactive Feedback for Networked CPS. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3412841.3441897


*SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea*

ACM ISBN 978-1-4503-8104-8/21/03…$15.00
https://doi.org/10.1145/3412841.3441897

## 1 INTRODUCTION

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 [37]. 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.

In recent years, control over wireless networks is becoming increasingly common due to requirements for low-cost sensing, flexibility, and low-power implementations [2, 25]. In this new era of edge devices with compute 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 networking limitations.

However, such systems might also have higher performance requirements, e.g., faster stabilization or lower settling time. As discussed in [24], 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.

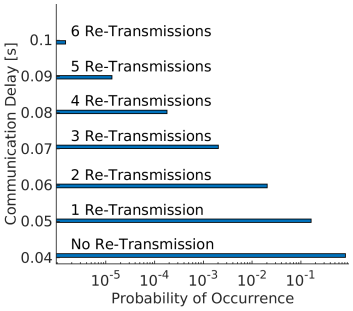


**Figure 1: Delay distribution.**

**Variable delay in wireless networks:** 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 [26, 43], 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. 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 [11]. This will also lead to variable delays, especially when the network load is high.

**Previously known techniques:** 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 [8, 35, 38], 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 [13, 15, 29], without mitigating the corresponding performance degradation. Other works have used a Kalman filter to predict the control input when packets are delayed [41, 42]. 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 [6, 33, 34] or redundantly transmitting each packet to the destination via multiple routes [29].

**Novelty of this work:** In contrast to these previously known techniques, as explained earlier, we in this paper present 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.

**Overview of the proposed proactive feedback strategy:** In this paper, 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.

**Contributions:** 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 precomputing 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.

**Organization:** The paper is organized as follows. Sec. 2 outlines related works. Sec. 3 formally describes the systems we consider. Sec. 4 discusses a motivational example to emphasize the challenges involved in implementing a high-performance controller when there is a large variation in the sensing-to-actuation delay. Sec. 5 presents our proposed technique in detail. Sec. 6 describes our experimental setup and presents experimental results to evaluate the performance of our proposed feedback strategy. Finally, Sec. 7 provides concluding remarks.

## 2 RELATED WORKS

Designing feedback control strategies in the presence of closed-loop delay is studied in the literature in several contexts [1, 3, 5, 9, 13, 15, 16, 20–23, 28, 29, 37–42, 44]. 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 [1, 13–15, 41, 42]. For example, the works in [13, 15] 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 [41, 42], a Kalman filter is used to compute upper and lower bounds on the estimation error based on the delay probability. In [1], 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 [2, 3, 12, 22, 46]. For example, formal models for analyzing the robustness and stability of a multi-hop wireless control network (MCN) are given in [3, 46]. In [12], 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, [22] 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 [6, 21, 23, 29, 33, 44]. In [23, 29], each data packet is simultaneously broadcasted to all possible nodes in range to increase the reliability and to reduce the number of retransmissions, which essentially reduces the end-to-end delay. On the other hand, in [6, 21, 33, 44], high-quality network resources are provisioned in parallel to the low-quality resources to obtain a better performance. Furthermore, [44] 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 [5, 16, 17, 28, 29, 38]. In [28], a holistic controller is proposed that generates actuation signals to physical plants and re-configures the wireless network (i.e., tunes the retransmission 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 [5, 38]. It is based on a worst-case end-to-end delay analysis [39] over the network. The work presented in [16, 17] synthesizes stable control and network schedules for a shared wireless control network in an integrated fashion. In the same vein, [20, 40] 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 paper. Barring two exceptions in [28, 29], 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 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 [29].

### 3.1 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 $\mathcal{V} = V_P \cup V_I \cup C$. Here, $V_\mathrm{P} = \{P_1, \cdots, P_n\}$ denotes the set of plant nodes. Each plant node comprises the sensor and actuator units connected locally to the physical plant (see Figure 3). 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. $C$ 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 $V_\mathrm{I}$ 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 $C$ 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 3, 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 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:

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

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:

$$u[k] = Kx[k] + Fr, \tag{2}$$

where $K$ is the feedback gain, $F$ is the feedforward gain, and $r$ is the reference output. By combining Eqs. (1) and (2), we get the closed-loop system model as follows:

$$x[k+1] = (A + BK)x[k] + Fr = A_{cl}x[k] + Fr. \tag{3}$$

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) [4], can be employed to determine $K$, while the final value theorem [10] 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.

## 4 A MOTIVATIONAL EXAMPLE

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 [36] 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 $r = 30$ 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 $h = 50$ 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 2, 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.

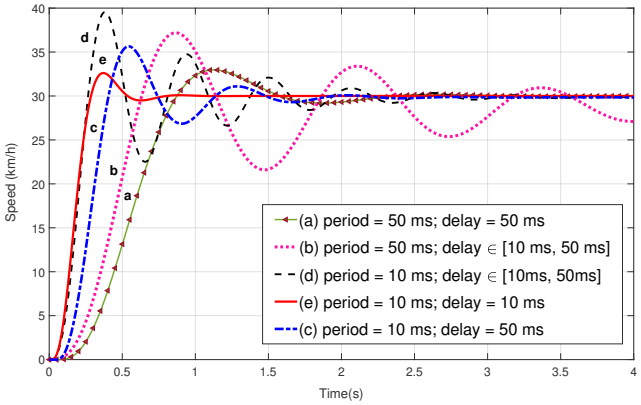


**Figure 2: Effect of variable delay in system response.**

**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 2 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 $d = 10$ ms. The red solid line in Figure 2 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 $h = 10$ 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.

# 5 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, $\hat{x}[k']$ 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 3 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 $\check{\delta}$ and a maximum delay $\hat{\delta}$ in the considered network.

## 5.1 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 ($\delta_f$). Based on $\delta_f$, 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 $\delta_f$ 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:

$$\check{k}_a = k + \lceil (\delta_f + \tau_c + \check{\delta}_r)/h \rceil \tag{4}$$

where $\tau_c$ 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:

$$\hat{k}_a = k + \lceil (\delta_f + \tau_c + \hat{\delta}_r)/h \rceil \tag{5}$$

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. (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 [15].

$$\hat{x}[k'] = A^{\Delta} x[k] + \sum_{j=0}^{\Delta-1} A^{\Delta-1-j} B u[k+j] \tag{6}$$

Following Eq. (6), for each such predicted actuation instant $k' \in \{\check{k}_a, \check{k}_a + 1, \cdots, \hat{k}_a\}$, the predictor estimates the plant-state $\hat{x}[k']$

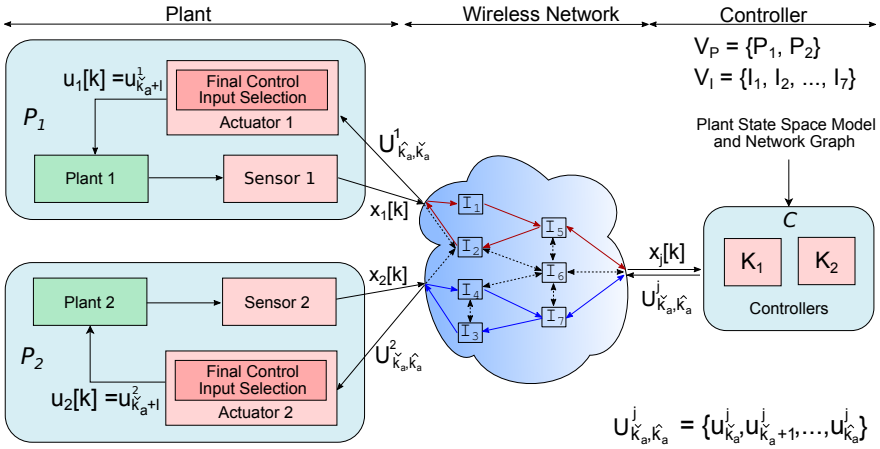


**Figure 3: Proposed proactive feedback scheme in a multi-hop wireless network.**

based on the last measured state valuation $x[k]$ and all the intermediate control inputs $\{u_{\check{k}_a}, u_{\check{k}_a+1}, \cdots, u_{k'-1}\}$.

## 5.2 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. (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:

$$\hat{x}[k'+1] = A\hat{x}[k'] + Bu[k']. \tag{7}$$

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 $\hat{k}_2 = 6$. 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 $U^k = \{u^k_{\check{k}_a}, u^k_{\check{k}_a+1}, \cdots, u^k_{\hat{k}_a}\}$.

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.

## 5.3 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 input $U^k$ for reaching the actuator be $\Delta$ 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 $\hat{x}[k']$. Here, $\hat{x}[k']$ is calculated using Eq. (6) based on the plant state $x[k]$, which is sensed at the $k$-th sampling instant.

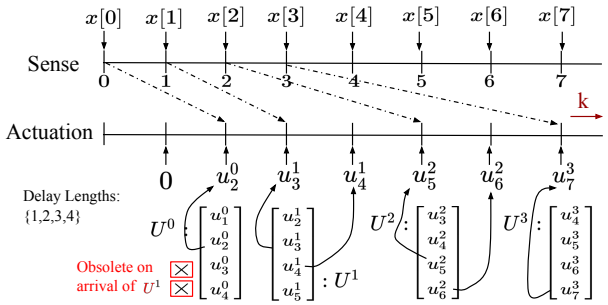


**Figure 4: Implementing zero delay in actuation.**

An illustrative example is given in Figure 4. 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.

# 6 EXPERIMENTAL EVALUATION

To evaluate the behavior of our proposed technique in real-world, we present experimental results obtained on a CPS testbed. We next

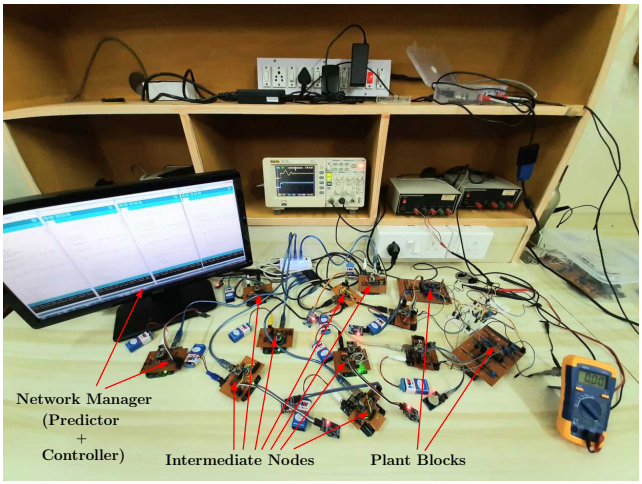


**Figure 5: The MCN testbed.**

describe our experimental setup, i.e., the CPS testbed, and then give experimental results.

## 6.1 Experimental Setup

We used the hardware/software co-designed testbed as depicted in Figure 5 in our experiments, which is built using off-the-shelf radio hardware. It provides a 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 3. The forward and return paths are colored red for loop 1 and blue for loop 2. Figure 5 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.

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 [19]. 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.

**Plant Node:** In our testbed, Double Integrator Circuits (DICs) form the physical plants, however, the hardware implementation of other

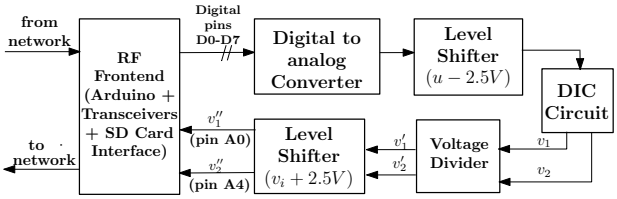


**Figure 6: Hardware details of the physical plant node.**

physical plants can be easily integrated in the future. We consider DICs that have the following discrete time dynamics [18].

$$x_p[t+1] = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} x_p[t] + \begin{bmatrix} -1 \\ 0.5 \end{bmatrix} u[t] \tag{8}$$

$$y[t] = \begin{bmatrix} 0 & 1 \end{bmatrix} x_p[t] \tag{9}$$

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. 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 [32], a Real Time Clock (RTC) module [27], and a SD memory card. The outline of the plant node is delineated in Fig. 6. During the sensing stage, plant state measurements are sent to the microcontroller through the voltage divider and lower level shifter. 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 [4], 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 through the microcontroller + RF + RTC interface, the predictor estimates the 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 [31], our bare-bone 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 3 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 3, 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.

## 6.2 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.

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. 6.1) 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 3 (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.

*6.2.1* ***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 [7]", the sampling periods of both DICs are chosen as 50 ms. Due to a higher sampling period, this approach suffers from a lower performance.

*FSFD:* In the "fixed-sampling and fixed-delay based control scheme [45]", 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 [5, 30]", 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 7 compares the output responses of the plant 1 using our proposed strategy and these aforementioned control techniques. As it is evident in Figure 7, 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.

*6.2.2* ***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 to the system matrices $A$ and $B$ (cf. Eq. (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 8 depicts

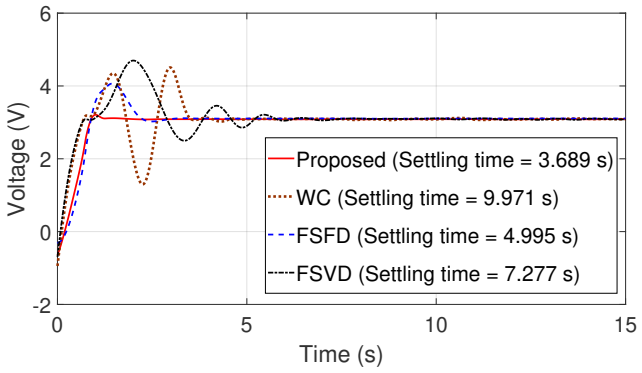


**Figure 7: Response of plant 1 under different schemes.**

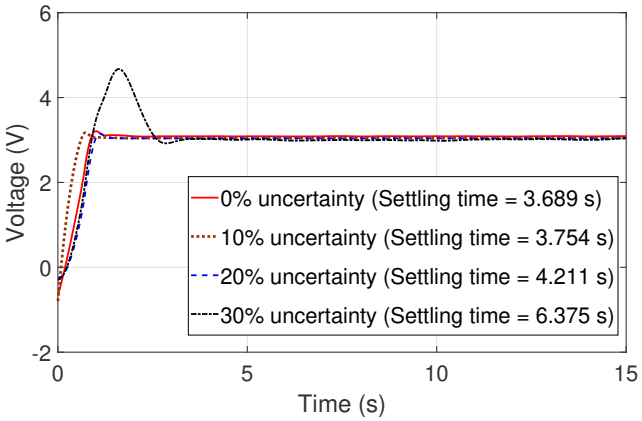


**Figure 8: Impact of model uncertainties on plant 1.**

the response of plant 1 and also reports the settling time when we consider $\gamma_1$ 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 8, 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 7) and FSVD (7.277 s, cf. Figure 7) 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.

*6.2.3* ***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 3), 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 8 shows the responses 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.

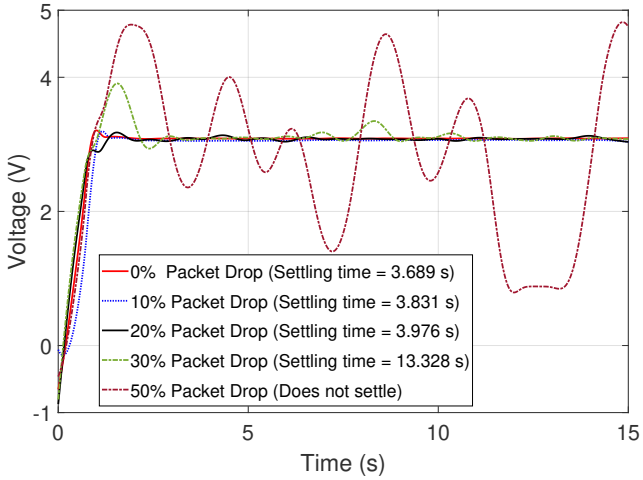


**Figure 9: Impact of packet drops on plant 1.**

*6.2.4* ***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 3) in the routing path of plant 2. The comparisons of settling times for the aforementioned experiments are reported in Table 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.

## 7 CONCLUSIONS AND FUTURE WORK

This paper presents a framework 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. We also present a hardware-software co-designed testbed for evaluating the proposed technique. The testbed can specify and implement different wireless CPS with negligible effort. Our proposed approach relies on transmitting multiple control inputs in each packet. Clearly, this increases the

**Table 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 |

channel utilization, such that packets collide more frequently when different devices are in range. An extension of this work will be analyzing the network and then choosing the appropriate sampling period for the controller based on the network load. 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.

## ACKNOWLEDGMENT

This work was supported by the DFG Project #387044055 and the NSF Award #2038960.

## REFERENCES

[1] G. Alldredge, M. S. Branicky, and V. Liberatore. 2008. Play-back buffers in networked control systems: Evaluation and design. In *American Control Conference (ACC)*.
[2] R. Alur, A. D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss. 2009. Modeling and analysis of multi-hop control networks. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*.
[3] R. Alur, A. D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss. 2011. Compositional modeling and analysis of multi-hop control networks. *IEEE Trans. Automat. Control* 56, 10 (2011), 2345–2357.
[4] K. J Åström and B. Wittenmark. 1997. *Computer-controlled systems*. Prentice-Hall, Inc.
[5] J. Bai, E. P. Eyisi, F. Qiu, Y. Xue, and X. D. Koutsoukos. 2012. Optimal cross-layer design of sampling rate adaptation and network scheduling for wireless networked control systems. In *International Conference on Cyber-Physical Systems (ICCPS)*.
[6] M. Balszun, D. Roy, L. Zhang, W. Chang, and S. Chakraborty. 2017. Effectively utilizing elastic resources in networked control systems. In *Embedded and Real-Time Computing Systems and Applications (RTCSA)*.
[7] D. Baumann, F. Mager, R. Jacob, L. Thiele, M. Zimmerling, and S. Trimpe. 2019. Fast feedback control over multi-hop wireless networks with mode changes and stability guarantees. *ACM Transaction on Cyber-Physical Systems)* 4, 2, Article 18 (2019), 32 pages.
[8] E. Bini and A. Cervin. 2008. Delay-aware period assignment in control systems. In *Real-Time Systems Symposium (RTSS)*.
[9] W. Chang, L. Zhang, D. Roy, and S. Chakraborty. 2017. *Control/architecture codesign for cyber-physical systems*. Springer Netherlands.
[10] C. Chen. 1994. *System and signal analysis*. The Oxford Series in Electrical and Computer Engineering,Oxford University Press, UK.
[11] LAN/MAN Standards Committee et al. 2003. Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE-SA Standards Board* (2003).
[12] A. D'Innocenzo, M. D. Di Benedetto, and E. Serra. 2013. Fault tolerant control of multi-hop control networks. *IEEE Trans. Automat. Control* 58, 6 (2013), 1377–1389.
[13] H. Gao and T. Chen. 2007. New results on stability of discrete-time systems with time-varying state delay. *IEEE Trans. Automat. Control* 52, 2 (2007), 328–334.
[14] P. Garcia, P. Castillo, R. Lozano, and P. Albertos. 2006. Robustness with respect to delay uncertainties of a predictor-observer based discrete-time controller. In *Conference on Decision and Control (CDC)*.
[15] P. Garcia, A. Gonzalez, P. Castillo, R. Lozano, and P. Albertos. 2012. Robustness of a discrete-time predictor-based controller for time-varying measurement delay. *Control Engineering Practice* 20, 2 (2012), 102 – 110.
[16] S. Ghosh, S. Dey, and P. Dasgupta. 2018. Co-synthesis of loop execution patterns for multi-hop control networks. *IEEE Embedded System Letters (LES)* 10, 4 (2018), 111–114.
[17] S. Ghosh, S. Dey, and P. Dasgupta. 2020. Pattern guided integrated scheduling and routing in multi-hop control networks. *ACM Transaction on Embedded Computing Systems* 19, 2, Article 9 (2020), 28 pages.
[18] S. Ghosh, S. Dutta, S. Dey, and P. Dasgupta. 2017. A structured methodology for pattern based adaptive scheduling in embedded control. *ACM Transactions on Embedded Computing Systems* 16, 5s (2017), 189:1–189:22.
[19] S. Ghosh, A. Mondal, P. H. Kindt, P. Sharma, Y. Agarwal, S. Dey, A. K. Deb, and S. Chakraborty. 2020. A programmable open architecture testbed for CPS education. *IEEE Design & Test* 37, 6 (2020), 31–38.
[20] D. Goswami, R. Schneider, and S. Chakraborty. 2011. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*.
[21] D. Goswami, R. Schneider, and S. Chakraborty. 2011. Re-engineering cyber-physical control applications for hybrid communication protocols. In *Design, Automation & Test in Europe (DATE)*.
[22] D. Goswami, R. Schneider, and S. Chakraborty. 2014. Relaxing signal delay constraints in distributed embedded controllers. *IEEE Transactions on Control Systems Technology* 22, 6 (2014), 2337–2345.
[23] R. Jacob, L. Zhang, M. Zimmerling, J. Beutel, S. Chakraborty, and L. Thiele. 2020. The time-triggered wireless architecture. In *Euromicro Conference on Real-Time Systems (ECRTS)*.
[24] R. Jacob, M. Zimmerling, P. Huang, J. Beutel, and L. Thiele. 2016. End-to-end real-time guarantees in wireless cyber-physical systems. In *Real-Time Systems Symposium (RTSS)*.
[25] X. Jin, A. Saifullah, C. Lu, and P. Zeng. 2019. Real-time scheduling for event-triggered and time-triggered flows in industrial wireless sensor-actuator networks. In *International Conference on Computer Communications (INFOCOM)*.
[26] A. N. Kim, F. Hekland, S. Petersen, and P. Doyle. 2008. When HART goes wireless: Understanding and implementing the WirelessHART standard. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.
[27] lastinuteengineers.com. 2020. Interface DS3231: Precision RTC module with Arduino. https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial. (2020).
[28] Y. Ma, D. Gunatilaka, B. Li, H. Gonzalez, and C. Lu. 2018. Holistic Cyber-Physical Management for Dependable Wireless Control Systems. *ACM Transactions on Cyber-Physical Systems* 3, 1 (2018), 3:1–3:25.
[29] F. Mager, D. Baumann, R. Jacob, L. Thiele, S. Trimpe, and M. Zimmerling. 2019. Feedback control goes wireless: Guaranteed stability over low-power multi-hop networks. In *International Conference on Cyber-Physical Systems (ICCPS)*.
[30] R. Mangharam and M. Pajic. 2013. Distributed control for cyber-physical systems. *Journal of the Indian Institute of Science* 93, 3 (2013), 353–387.
[31] Nordic Semiconductor. 2019. nRF Connect SDK. https://developer.nordicsemi.com/nRF_Connect_SDK/doc/1.0.0/nrf/ug_esb.html. (2019). [Online; accessed 11-Nov-2019].
[32] Nordic Semiconductor. 2019. nRF24 Series. https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF24-series. (2019). [Online; accessed 11-Nov-2019].
[33] D. Roy, W. Chang, S. K. Mitter, and S. Chakraborty. 2019. Tighter dimensioning of heterogeneous multi-resource autonomous CPS with control performance guarantees. In *Design Automation Conference (DAC)*.
[34] D. Roy, S. Ghosh, Q. Zhu, M. Caccamo, and S. Chakraborty. 2020. GoodSpread: Criticality-aware static scheduling of CPS with multi-QoS resources. In *Real-Time Systems Symposium (RTSS)*.
[35] D. Roy, C. Hobbs, J. Anderson, M. Caccamo, and S. Chakraborty. 2021. Timing debugging for cyber-physical systems. In *Design, Automation and Test in Europe (DATE)*.
[36] D. Roy, L. Zhang, W. Chang, D. Goswami, and S. Chakraborty. 2016. Multi-objective co-optimization of FlexRay-based distributed control systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*.
[37] D. Roy, L. Zhang, W. Chang, S. K. Mitter, and S. Chakraborty. 2018. Semantics-preserving cosynthesis of cyber-physical systems. *Proc. IEEE* 106, 1 (2018), 171–200.
[38] A. Saifullah, C. Wu, P. B. Tiwari, Y. Xu, Y. Fu, C. Lu, and Y. Chen. 2014. Near optimal rate selection for wireless control systems. *ACM Transactions on Embedded Computing Systems* 13, 4s (2014), 128:1–128:25.
[39] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. 2011. End-to-end communication delay analysis in wirelesshart networks. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*.
[40] R. Schneider, D. Goswami, S. Zafar, S. Chakraborty, and M. Lukasiewycz. 2011. Constraint-driven synthesis and tool-support for FlexRay-based automotive control systems. In *International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS)*.
[41] L. Shi, L. Xie, and R. M. Murray. 2009. Kalman filtering over a packet-delaying network: A probabilistic approach. *Automatica* 45, 9 (2009), 2134 – 2140.
[42] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. 2004. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control (TAC)* 49, 9 (2004), 1453–1464.
[43] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt. 2008. WirelessHART: Applying wireless technology in real-time industrial process control. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*.
[44] H. Voit, A. Annaswamy, R. Schneider, D. Goswami, and S. Chakraborty. 2012. Adaptive switching controllers for systems with hybrid communication protocols. In *American Control Conference (ACC)*.
[45] W. Wang, D. Mosse, D. Cole, and J. G. Pickel. 2018. Dynamic wireless network reconfiguration for control system applied to a nuclear reactor case study. In *International Conference on Real-Time Networks and Systems (RTNS)*.
[46] G. Weiss, A. D'Innocenzo, R. Alur, K. H. Johansson, and G. J. Pappas. 2009. Robust stability of multi-hop control networks. In *Conference on Decision and Control (CDC) held jointly with Chinese Control Conference (CCC)*.