Control over Low-Power Wide-Area Networks

Aakriti Jain†
Wayne State University
Detroit, USA

Prashant Modekurthy†
University of Nevada Las Vegas
Las Vegas, USA

Abusayeed Saifullah
Wayne State University
Detroit, USA

***Abstract*—There has been a growing interest to adopt low-power wide-area network technology, specially LoRa, for industrial control applications. Its machine-to-machine communication capabilities can enable management of large-area applications (e.g., oil fields over hundreds of km^2) or process plants that are often positioned far from the central operations center, at inconvenient or hazardous locations in difficult terrain or offshore. While recent works have studied real-time communication over LoRa, they have not considered optimizing control performance. To optimize control performance, industrial automation needs a *co-design* of real-time scheduling and control. Such a co-design, in general, is highly challenging due to complex dependencies between control performance, plant dynamics, and real-time communication. Existing co-design approaches for other wireless domains are not applicable to LoRa network. LoRa nodes are extremely power-constrained hindering frequent communication and scale. In this paper, we propose a highly energy-efficient and scalable framework for real-time scheduling and control co-design for a LoRa network. By taking into account LoRa characteristics, the co-design approach entails state-aware communication and control to dynamically update the sampling rates while meeting real-time constraints. To minimize communication and synchronization overhead, the co-design is decomposed through a partitioned scheduling. We consider co-design in each partition of the control loops by developing a new schedulability condition. The proposed scheduling-control co-design solution dynamically determines the sampling rates of the sensors to optimize control performance. Simulations based on NS-3 and a custom control script show that our co-design approach minimizes control cost at least by 80% as compared to the baselines.**

***Index Terms*—Networked Control system, IIoT, CPS, LPWAN, LoRa, Real-Time Communication, Model Predictive Control, Stability, Scheduling**

I. Introduction

In industrial domain, Internet of Things (IoT) and cyber-physical systems (CPS) are emerging in large-scale and wide-area applications. For example, oil fields can extend over hundreds of km^2 (e.g., the East Texas Oil-field has an area of 74×8km^2) requiring numerous sensors and actuators for their management [1]. Emerson is planning to deploy 10,000 sensors and actuators in an oil-field in Texas [2]. Traditional industrial wireless solutions based on short-range technologies such as WirelessHART [3] and ISA100.11a [4] form multi-hop mesh networks, posing a huge challenge to support such large-scale and wide-area deployments. Therefore, many recent works have proposed to adopt *Low-Power Wide-Area Network (LPWAN)* for industrial control applications [5]–[7]. In parallel, there has been a growing interest for LP-WANs in industries, as shown in a recent survey conducted by ON World and the International Society of Automation (ISA) that 57% of industrial IoT professionals prefer LP-WAN solutions [8]. Thanks to their long range (km) at low-power (milliwatts), LPWANs can enable machine-to-machine communication thereby greatly benefiting the applications like management of pipelines (that can be hundreds of km long), silo level, oil refineries, and process plants that are often positioned far from the central operations center, at inconvenient or hazardous locations in difficult terrain or offshore.

LoRa (Long Range) is a leading LPWAN technology that is commercially available across the world with more than 600 known use cases and over 50 million devices deployed on every inhabited continent [9]. Existing works like [5], [7] have specifically considered LoRa as a communication backbone for industrial control applications, specially for the regions like North America, where LoRa spectrum duty-cycling is not required. ABI research predicts that more than 50% of all LPWAN connections will be based on LoRa by 2026 due to it is flexible for outdoor and indoor applications [10]. Works [6], [7] have proposed real-time communication over LoRa to enable control applications. However, these works have not considered control performance optimization on LoRa networks. In this paper, we bridge the gap between real-time communication and control applications and address control performance optimization for a LoRa network.

Optimizing control performance over a LoRa network is highly challenging. To optimize control performance, industrial automation needs a *co-design* of real-time scheduling and control. Such a co-design, in general, is highly challenging due to complex dependencies between control performance, plant dynamics, and real-time communication. Existing co-design approaches [11]–[15] for other wireless domains are not applicable to LoRa network as LoRa nodes are extremely power-constrained. A LoRa node's battery-life is expected to be of several years as changing batteries too often in numerous devices and in remote or inaccessible places is not a practical option. While energy-efficiency is a general requirement and challenge in LPWANs, it becomes more complicated in LoRa when combined with real-time requirement. Specifically, the nodes' communications have to be minimal which makes real-time communication extremely challenging.

The fundamental building blocks of any industrial automation system are feedback control loops that largely rely on real-time communication between sensors and actuators. Enabling closed-loop communication under severe energy constraints of the nodes is quite challenging. One key method of conserving

†Co-primary author

979-8-3503-6927-4/24/$31.00 ©2024 IEEE
DOI 10.1109/ICCPS61052.2024.00024

energy of the LoRa devices is to reduce communication while doing critical control operation, which could be achieved by dynamically selecting the sampling rates for the control loops. However, the selection of sampling rates for these loops is crucial to achieve a balance between control performance and real-time communication. Opting for a low sampling rate can degrade control performance, while a high rate can introduce substantial communication delays, leading to subpar performance. Furthermore, the large-scale deployment of LPWAN/LoRa poses a challenge on the scalability and stability of the network.

In this work, we explore scheduling-control co-design to maintain control performance at low energy cost. We address the above challenges by proposing a highly energy-efficient and scalable framework for real-time scheduling and control co-design for a LoRa network. By taking LoRa characteristics into account, the co-design approach entails state-aware communication and control to dynamically update the sampling rates while meeting real-time constraints and ensuring stability. Using *Model Predictive Control (MPC)* theory, it samples the system at a slow rate in steady state and faster during the transients. To minimize communication and synchronization overhead, the co-design is decomposed through a partitioned scheduling after which scheduling-control co-design is done inside each partition of the control loop. The partitioning is done by adopting RTPL, a partitioned real-time scheduling algorithm for LoRa proposed in [7]. We develop the co-design in each partition subject to a schedulability constraint which, to our knowledge, is the first sufficient schedulability condition for real-time protocol on a LoRa network supporting concurrent transmission and reception of packets on distinct channels. The co-design solution dynamically determines the sampling rates of the sensors to optimize control performance.

The paper is organized as follows. Section II presents the necessary background and system model. Section III reviews related work. Section IV presents the proposed scheduling-control co-design. Section V evaluates the proposed co-design, and Section VI concludes the paper.

II. System model and Background

In this section, we present a brief overview of LoRa and the RTPL MAC protocol. We also present the system model, consisting of a network model and a physical control system.

A. An Overview of LoRaWAN

LoRa is a leading LPWAN technology. Typically, LoRaWAN is known as the MAC layer of the LoRa stack, while LoRa is considered as the physical (PHY) layer. A LoRaWAN architecture consists of three devices: gateways, end devices (nodes), and a network server. Nodes are sensors or actuators that can communicate directly with a gateway over long distance (several kilometers) and have a half-duplex radio, enabling transmission or reception of a packet at a time but not both. A LoRaWAN network can consist of multiple gateways that relay information to the network server through a local LAN/Internet. The network manager and the controller are responsible for the overall operation of the network. Unlike the battery powered nodes, the gateway is line-powered, Internet-connected and has high computation capabilities.

LoRa adopts a proprietary *Chirp Spread Spectrum (CSS)* for communication that makes it less sensitive to noise and interference. A LoRa transceiver entails five transmission parameters that can be adjusted during runtime: transmission power, carrier frequency (channel), *spreading factor (SF)*, bandwidth (BW), and coding rate (CR). These parameters are crucial role in determining the transmission duration, energy consumption, reliability, and range of the communication. SF ranges between 7-12 and represents the number of bits encoded in a symbol. A higher spreading factor implies a lower-bit rate, longer transmission time, a higher Signal-to-Noise Ratio (SNR), increased receiver sensitivity, and longer range. Depending on the SF and bandwidth, LoRa achieves data rates between 0.3 kbps and 27 kbps.

B. Network Model

We consider an LPWAN based on LoRaWAN. A *communication path* in a LoRaWAN network is a combination of a SF and a channel. LoRaWAN uses *orthogonal* SFs [16], where two communications paths are said to be *orthogonal* to each other if they have different channels or different SFs. An *uplink communication path (UCP)* is a unique combinations of uplink channels and spreading factors on which a gateway can simultaneously receive packets. Similarly, a *downlink communication path (DCP)* is a unique combination of downlink channels and spreading factors on which a gateway can simultaneously transmit packets.

Sensors are responsible for measuring process variables and transmitting them to the controller on a UCP. The controller, in turn, sends control commands to the actuators on a DCP. Actuators apply the control command to modify physical processes. We refer to the closed loop communication from a sensor to actuator via a controller as a control loop. A LoRa gateway is equipped with $(1 + m')$ radios. One radio enables simultaneous reception across m UCPs, while the remaining radios enable concurrent transmissions across m' DCPs.

Each control loop is allocated multiple time slots on the uplink and downlink communication path. The system designer determines the number of time slots allocated to a control loop on a communication path, considering the control loop's reliability requirement and noise conditions on the communication path. We define $\mathbb{C}_i^{up}(UCP_j)$ as the communication time requirement on the uplink communication path UCP_j for control loop ℓ_i, referring to the number of time slots allocated for control ℓ_i on the communication path UCP_j. Similarly, we define $\mathbb{C}_i^{down}(DCP_j)$ as the communication time requirement for control loop ℓ_i on the downlink communication path DCP_j.

We adopt the *RTPL partitioned scheduling* from a recent work [7]. In RTPL, control loops are partitioned and each partition is assigned a UCP and DCP. Control loops are scheduled within a partition using earliest deadline first (EDF).

C. Physical Control System

We consider a LoRa network with n wireless control loops $\ell_1, \ell_2, \cdots, \ell_n$, where ℓ_i represents the feedback control loop between a sensor node s_i and actuator a_i through a controller at the gateway. For a control loop ℓ_i, the controller computes control command $u_i(k)$ upon receiving state information $x_i(k)$ from sensor s_i, and then sends it to the actuator a_i. The controller also computes the sampling period for each loop.

We model the physical plant as a linear time-invariant system (LTI). We use an LTI representation since it models a wide variety of systems with satisfactory accuracy. A control loop ℓ_i is represented in the discrete state space as follows.

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k); \quad y_i(k) = C_i x_i(k),$$

Here, $x_i(k)$ is the state of the control loop ℓ_i at time step k. The control input is denoted as $u_i(k)$, while the observed output is represented by $y_i(k)$. $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m}$, and $C_i \in \mathbb{R}^{p\times n}$. For each control loop ℓ_i, we assume that the pair (A_i,B_i) is controllable, and that the pair (A_i,C_i) is observable. Since each control loop is controllable, an optimal controller such as model predictive controller ensures all control loops reach their desired state and remain in the desired state.

We denote the sampling period of ℓ_i by p_i and the deadline of ℓ_i by d_i. We consider an implicit deadline system, i.e., the period of control loop equals its deadline ($p_i = d_i$). If a control loop ℓ_i meets its deadline d_i for all of its packets, then it is said to be *schedulable*. A set of control loops is schedulable, when each control loop in the set is schedulable.

D. RTPL Overview

RTPL is a low-overhead real-time communication protocol for LoRa that leverages the concurrent reception capabilities of LoRa proposed in [7]. It adopts a partitioned scheduling approach, dividing control loops into partitions. The control loops in each partition uses one uplink and one downlink communication path with the same SF. Communications of the control loops inside a partition happen only on that partition's uplink and downlink communication paths.

Within a partition, the nodes adopt a time division multiple access (TDMA) protocol, dividing the network time into slots. The time slot length is defined as the time to transmit a packet on the smallest spreading factor. Packet transmission lengths vary with the SFs. Thus, a control loop requires multiple time slots to transmit a packet on higher SFs. Within a partition, the control loops follow the EDF scheduling. The period and communication time requirement for uplink and downlink communication for all control loops is disseminated to all nodes within a partition. Nodes use EDF scheduling to transmit and receive packets.

RTPL adopts a worst-fit partitioning algorithm. The algorithm iterates over the list of control loops. For each control loop, the partitioning algorithm identifies a partition with the largest remaining utilization after allocation. If the control loop is schedulable on that partition, it assigns the control loop to that partition. If it cannot be assigned to any partition, the algorithm decides the case as unscheduleable and stops.

RTPL adopts a utilization-based schedulability test to determine the schedulability within a partition. Considering a control loop ℓ_i has a period of T_i and a deadline $d_i = T_i$, the utilization of ℓ_i on the partition with UCP_j and DCP_j is

$$\mu_i(UCP_j) = \frac{\mathbb{C}_i^{up}(UCP_j) + \mathbb{C}_i^{down}(DCP_j)}{T_i}.$$

Considering $\pi(UCP_j)$ represents the set of control loops allocated to that parition, the schedulability test within the partition is given by the following equation.

$$\sum_{\ell_i \in \pi(UCP_j)} \mu_i(UCP_j) \leq 1$$

III. Related Work

Several recent studies have suggested using LoRa for industrial control applications [5]–[7] and real-time communication [6], [7]. [6] introduces a medium access strategy to support real-time flows in LoRa-based networks, while [7] presents a partitioned real-time scheduling strategy called RTPL for LoRa. [17] proposes a strategy to enhance task schedulability by optimum period selection. In contrast to these works, our approach involves dynamically updating sampling rates based on a control cost formulation, which jointly determines the control signal and sampling period.

Prior studies on scheduling and control co-design [11]–[15], [18], [19] have predominantly focused on traditional short-range wireless networks, leaving the potential of LPWAN/LoRa unexplored. These studies assume fixed sampling periods for each sensor, which can be power-consuming, particularly when the system is in a stable state. In contrast, our proposed state-aware system model aims to optimize control performance while ensuring real-time performance.

The study in [20] examines a sufficient condition for the existence of a stabilizing controller under a fixed upper bound on the probability of packet failure. Typically, such a fixed upper bound is derived for worst-case scenarios, leading to significant overheads and high energy consumption at the sensors. To mitigate this issue, the approach in [21] adopts an event-based control strategy, where a sensor transmits packets only when an event occurs. However, this method requires continuous monitoring of the system state to check the triggering condition, leading to high energy consumption.

To address the energy consumption limitation, [22], [23] propose stability under a self-triggered control without schedulability constraints. However, [24] proposes a self-triggered control based on a virtual link capacity margin, which is not applicable to LoRa, even under a single channel. [25] suggests a self-triggered control for a TDMA network scheduler, but this approach relies on single-channel transmission with packet flooding, leading to high energy consumption. Energy conservation at the controller is explored in [26]. but a similar approach is not suitable for conserving energy at sensors. Additionally, [5] has investigated closed-loop communication in LoRa networks to facilitate event-triggered control. However, this study does not sufficiently address energy constraints, as it requires the actuators to remain in

continuous listen mode, and it does not leverage the potential of LoRa's concurrent reception on orthogonal SFs.

IV. Design of Scheduling-Control Co-design Framework

In this section, we present an energy-efficient and scalable real-time scheduling and control co-design framework for LoRa. We first formulate the co-design problem, for a single control loop, as an optimization problem solved using a model predictive controller. The insights from the single control loop system drive the design for a multiple control loop system.

Typically, industrial control systems employ static sampling, wherein a sensor measures the state periodically at a fixed rate. A static sampling rate negatively impacts the control performance and energy consumption at each node. For example, when a control loop is not in a stable state, sampling the system more frequently improves control performance, enabling a smaller settling time, i.e., reaching a stable state faster. When a control loop is in a stable state, sampling the system less frequently reduces energy consumption at the nodes. LPWAN devices are energy-constrained, and minimizing energy consumption is crucial for their use in control systems. We propose a co-design of scheduled communication and control with dynamic sampling, wherein the sampling rate of a control loop is updated based on the current state.

A. An Overview of the Scheduling-Control Co-design

Sampling rate selection renders opposing impacts on control performance and real-time guarantees. Control performance benefits from high sampling rates, but high sampling rates may violate real-time requirements due to limited capacity/bandwidth of the network. Lower sampling rates have higher chances of ensuring real-time guarantees, but they lead to poor control performance. Thus, sampling rate selection must balance control performance and real-time guarantees.

To balance control performance and real-time guarantees, we formulate the scheduling-control co-design as an optimization problem using model predictive control theory [27], [28]. The model predictive controller identifies a sequence of control commands and sampling rate for each control loop, resulting in a minimum control cost. Additionally, it offers the capability to pre-compute system output $y(k)$ and control signal $u(k)$ to avoid sudden changes and delay in system response. The optimization problem includes a constraint (schedulability condition), ensuring real-time guarantees of end-to-end communication under the selected periods (inverse of the rates). Upon generating the optimal periods, the control loops are partitioned using the RTPL algorithm and the communications within each partition follow the EDF policy. Fig. 1 shows the full framework of the co-design.

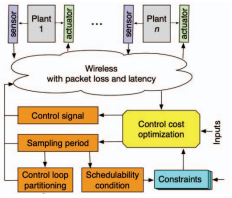


Figure 1. Scheduling-Control Co-Design Framework

After partitioning, to minimize the computation and communication overhead, we propose to execute the constrained optimization problem of finding periods and control commands for each control loop after a **sync period**. A network designer may tune the sync period prior to network deployment. The sync period constitutes multiple sampling instances of each control loop, i.e., the sync period (τ) is significantly greater than the sampling period of each control loop, $\tau > p_i \ \forall \ell_i$. After each sync period, the controller solves the constrained optimization to compute the control command and period of all control loops. The network manager then broadcasts the updated periods to all nodes and synchronizes their time. This constitutes our proposed dynamic sampling approach.

Specifically, the dynamic sampling approach functions as follows: Nodes wake-up periodically after each sync period to receive new sampling periods. Upon receiving the updated periods, nodes compute the next transmission time according to the RTPL scheduler. The default mode of the sensors is to sleep. Sensors wake up before the transmission time to sense the plant's state and transmit the message to the controller. Upon receiving the new state, the controller executes the unconstrained control cost optimization to compute (only) the control command. The controller sends the new control command to the actuator, which applies the control command on the plant. The default mode of the actuator is to apply the received control command on the plant at each time step with its radio off.

The controller and network manager operate in discrete time. For the network manager, the length of a downlink packet transmission on the smallest SF and its acknowledgment defines the time slot size. The plant model defines the controller's time step size. We consider a single time step within the model of a plant comprises of multiple time slots of the network. Precisely, one time step of the plant ℓ_i contains $\mathbb{C}_i^{up}(UCP_j) + \mathbb{C}_i^{down}(DCP_j)$ (the sum of uplink and downlink communication time requirement) network time slots. The dissimilar step sizes facilitates the application of control command during the same time step as the sensor senses the state, thereby simplifying the stability analysis. Following the dissimilar step sizes, we define the period of the control loop ℓ_i according to the controller time step as p_i, $T_i = p_i \times (\mathbb{C}_i^{up}(UCP_j) + \mathbb{C}_i^{down}(DCP_j))$, where T_i is the period in network time slots.

For simplicity in explanation, we first formulate the co-design problem for a single control loop. We use the insights of this problem to formulate and propose an approach to address the scheduling-control co-design of multiple control loops.

B. Formulating Scheduling-Control Co-Design for a Single Control Loop

The goal of the controller is to move the state of the system to 0 (a zero matrix of the appropriate dimension). To meet the goal, we formulate the controller's cost function to trade off the traditional quadratic control cost and sampling cost similar to [23]. A control loop ℓ_i operates with a period of $p_i \in P_i$, where $P_i = [1, 2, \cdots, \gamma_i] \subset \mathbb{N}^+$. The model predictive controller generates a sequence of control commands $U_i = \{u_i(k), u_i(k+p_i), \cdots\}$. $Q_i > 0$ and $R_i > 0$ are symmetric matrices of appropriate dimensions indicating

weight. Considering $\sigma_i \in \mathbb{R}^+$ is a design variable used to trade off sampling cost against control cost, the cost function of the controller is expressed by Eq. (1).

$$J_i\big(x_i(k), p_i, U_i\big) = \frac{\sigma_i}{p_i} + \sum_{j=0}^{\infty} \Big(\|x_i(k+j)\|^2_{Q_i} + \|u_i(k+j)\|^2_{R_i} \Big) \quad (1)$$

We formulate the cost function for an infinite-horizon controller to ensure the period selection ensures asymptotic stability at the origin. During runtime, the controller minimizes the cost for a finite horizon. In a network with a single control loop, there is no network contention. Thus, the sensor of control loop ℓ_i samples the state periodically with a period p_i, starting at time step 0, until the sync period τ. After the sync period τ, we assume the control loop ℓ_i operates periodically with a period ϕ_i. Therefore, the control cost can be expanded as follows.

$$\begin{aligned} J_i\big(x_i(k), p_i, U_i\big) = \frac{\sigma_i}{p_i} + \\ \Big(\sum_{r=0}^{\lceil \frac{\tau}{p_i} \rceil} \sum_{j=0}^{p_i - 1} \big(\|x_i(k + r.p_i + j)\|^2_{Q_i} + \ \|u_i(k + r.p_i)\|^2_{R_i} \big) \Big) + \\ \Big(\sum_{r=0}^{\infty} \sum_{j=0}^{\phi_i - 1} \big(\|x_i(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i + j)\|^2_{Q_i} + \\ \|u_i(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i)\|^2_{R_i} \big) \Big) \end{aligned} \quad (2)$$

In a single control loop system, the smallest period of a control loop of one time step ensures real-time guarantees on the network, since the time step of the plant is large enough to accommodate both uplink and downlink communication on the network. Thus, the single loop problem formulation minimizes the cost function in Eq. (2), ignoring scheduling constraints. The goal of this formulation is to compute a series of control commands and the sampling period for the first sync period that result in a minimum cost as shown below.

$$\textbf{Minimize}_{\{p_i, U_i(p_i)\}} J_i\Big(x_i(k), p_i, U_i(p_i)\Big) \quad (3)$$

Although the controller executes the above optimization problem only at sync periods, the computation overhead for finding a solution is high. To minimize the overhead, we split the problem into two sub-problems. The first sub-problem aims to compute the sequence of control commands for an arbitrary period p_i. We then use the control commands to simplify the sub-problem of period selection. Thus the scheduling-control co-design problem formulation is written as shown below.

$$\textbf{Minimize}_{\{p_i\}} \Big(\textbf{Minimize}_{\{U_i(p_i)\}} J_i\Big(x_i(k), p_i, U_i(p_i)\Big) \Big) \quad (4)$$

The control command for $J_i(x_i(k), p_i, U_i(p_i))$ can be computed using a dynamic programming approach as mentioned in [29]. To compute the control command, we define the following notations. $\lambda(A)$ denotes the eigenvalues of A.

$$A_i^{(j)} = A_i^j, \qquad Q_i^{(j)} = Q_i + \sum_{q=2}^{j} {A_i^{(q-1)}}^T Q_i A_i^{(q-1)}$$

$$B_i^{(j)} = \sum_{q=0}^{j-1} A_i^q B_i, \qquad N_i^{(j)} = \sum_{q=2}^{j} {A_i^{(q-1)}}^T Q_i B_i^{(q-1)}$$

$$R_i^{(j)} = R_i + \sum_{q=2}^{j} \big({B_i^{(q-1)}}^T Q_i B_i^{(q-1)} + R_i \big)$$

Lemma 1. *For a system with $Q > 0$, $R > 0$, and $(A_i^{(p_i)}, B_i^{(p_i)})$ is controllable, then minimizing the control cost function for an infinite-horizon optimization problem*

$$\begin{aligned} \textbf{Minimize}_{\{U_i(p_i)\}} \sum_{r=0}^{\infty} \sum_{j=0}^{\phi_i - 1} \big(\|x_i(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i + j)\|^2_{Q_i} + \\ \|u_i(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i)\|^2_{R_i} \big) \end{aligned}$$

results in a control command

$$u_i\Big(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i\Big) = -V^{(\phi_i)} x_i\Big(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i\Big) \forall r \in \mathbb{N}^+$$

where $V_i^{(\phi_i)}$ is given below

$$\begin{aligned} V_i^{(\phi_i)} = \Big(R_i^{(\phi_i)} + {B_i^{(\phi_i)}}^T W_i^{(\phi_i)} B_i^{(\phi_i)} \Big)^{-1} \\ \Big({A_i^{(\phi_i)}}^T W_i^{(\phi_i)} B_i^{(\phi_i)} + N_i^{(\phi_i)} \Big)^T \end{aligned} \quad (5)$$

and $P_i^{(\phi_i)}$ is obtained by solving the Riccati equation below

$$\begin{aligned} W_i^{(\phi_i)} = & Q_i^{(\phi_i))} + {A_i^{(\phi_i))}}^T W_i^{(\phi_i))} A_i^{(\phi_i))} \\ & - \Big({A_i^{(\phi_i))}}^T W_i^{(\phi_i))} B_i^{(\phi_i))} + N_i^{(\phi_i))} \Big) V_i^{(\phi_i))} \end{aligned} \quad (6)$$

Proof. The proof follows the proof for infinite-horizon optimal control problem in [23]. □

We extend Lemma 1 to provide control commands for all samples.

Lemma 2. *For a system with $Q > 0$, $R > 0$, and $(A_i^{(p_i)}, B_i^{(p_i)})$ is controllable, then the objective function of the finding the control command can be expressed as follows*

$$\textbf{Minimize}_{\{U_i(p_i)\}} J_i\Big(x_i(k), p_i, U_i(p_i)\Big) = \frac{\sigma_i}{p_i} + \|x_i(k)\|^2_{W_i^{(p_i)}} \quad (7)$$

where

$$\begin{aligned} W_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)} &= Q_i^{(\lceil \frac{\tau}{p_i} \rceil p_i))} + {A_i^{(\lceil \frac{\tau}{p_i} \rceil p_i))}}^T W_i^{(\phi_i))} A_i^{(\lceil \frac{\tau}{p_i} \rceil p_i))} \\ &- \Big({A_i^{(\lceil \frac{\tau}{p_i} \rceil p_i))}}^T W_i^{(\phi_i))} B_i^{(\lceil \frac{\tau}{p_i} \rceil p_i))} + N_i^{(\lceil \frac{\tau}{p_i} \rceil p_i))} \Big) V_i^{(\lceil \frac{\tau}{p_i} \rceil p_i))} \\ V_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)} &= \Big(R_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)} + {B_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)}}^T W_i^{(\phi_i)} B_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)} \Big)^{-1} \\ & \Big({A_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)}}^T W_i^{(\phi_i)} B_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)} + N_i^{(\lceil \frac{\tau}{p_i} \rceil p_i)} \Big)^T, \\ W_i^{(j.p_i)} &= Q_i^{(j.p_i))} + {A_i^{(j.p_i))}}^T W_i^{((1+j)p_i))} A_i^{(j.p_i))} \\ &- \Big({A_i^{(j.p_i))}}^T W_i^{((1+j)p_i))} B_i^{(j.p_i))} + N_i^{(j.p_i))} \Big) V_i^{(j.p_i))} \\ V_i^{(j.p_i)} &= \Big(R_i^{(j.p_i)} + {B_i^{(j.p_i)}}^T W_i^{((1+j)p_i)} B_i^{(j.p_i)} \Big)^{-1} \\ & \Big({A_i^{(j.p_i)}}^T W_i^{((1+j)p_i)} B_i^{(j.p_i)} + N_i^{(j.p_i)} \Big)^T \end{aligned}$$

$\forall j \in [1, \lceil \frac{\tau}{p_i} \rceil - 1]$, and $W_i^{(\phi_i)}$ is given by Lemma 1. The optimal control command for the above formulation is expressed as shown below.

$$u_i\Big(k + j.p_i\Big) = -V^{(j.p_i)} x_i\Big(k + j.p_i\Big) \qquad \forall j \in [1, \lceil \frac{\tau}{p_i} \rceil]$$

$$u_i\Big(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i\Big) = -V^{(\phi_i)} x_i\Big(k + \lceil \frac{\tau}{p_i} \rceil p_i + r.\phi_i\Big) \ \forall r \in \mathbb{N}^+$$

Proof. Upon applying Lemma 1 to the cost function, the cost function reduces to a finite-horizon optimal control, solved using the recurring Riccati equation for $W_i^{(p_i)}$ [29]. □

Applying Lemma 2, the objective function of the scheduling-control co-design can be reduced to Eq. (8).

$$\textbf{Minimize}_{\{p_i\}} \frac{\sigma_i}{p_i} + \|x_i(k)\|^2_{W_i^{(p_i)}} \quad (8)$$

The algorithm for finding the suitable period for Eq. (8) involves computing the cost for each feasible period $p_i \in P_i$ and finding the period that results in the smallest cost. The above algorithm has to test several period values before termination, incurring large computation overhead. To reduce computation overhead, we reduce the search space of the feasible periods to those that ensure stability.

Theorem 1 (Stability of a Control Loop)**.** *For a system with $Q > 0$, $R > 0$, and $(A_i^{(p_i)}, B_i^{(p_i)})$ is controllable, then choosing $p_i \in P_i \subset \mathbb{N}^+$,*

$$\phi_i = \max\{p_i | p_i \in P_i, \forall \lambda \in \lambda(A_i)\ \lambda^{p_i} \neq 1\ if\ \lambda \neq 1\} \quad (9)$$

and using the dynamic sampling approach results in asymptotic stability at the origin, i.e.,

$$\lim_{k\to\infty} x(k) = 0$$

where γ_i is the maximum $p_i \in P_i$, $\epsilon \in (0, 1]$ and $p_i \in P_i$ fulfills

$$(A^{(p_i)} - B^{(p_i)}V^{(p_i)})^T W^{(\phi_i)}(A^{(p_i)} - B^{(p_i)}V^{(p_i)}) \leq (1-\epsilon)W(p_i). \quad (10)$$

Direct proof. The difference between the cost function at time step k and time $k + p_i$ is bounded, leading to the system state at $k \to \infty$ to be 0. This proof follows the stability proof in [23]. □

The stability proof provides a mathematical expression of computing the set of feasible periods using Eq. (10). The set of feasible periods P_i and Riccati solutions for each period W^{p_i} are computed during network deployment and stored in memory. The period selection algorithm iterates over the set of feasible periods and selects the period that results in smallest cost for $J_i(x_i(k), p_i, U(p_i))$.

C. Formulating Scheduling-Control Co-Design for Multiple Control Loops

In this section, we plan to adopt the period selection algorithm from the single control loop to develop a period selection algorithm for multiple control loops. For a network with multiple control loops, RTPL MAC schedules the packets in the network. The use of RTPL MAC introduces two primary challenges in adopting the aforementioned period selection. First, the period selection algorithm should ensure real-time guarantees in RTPL. Second, the use of EDF scheduling in RTPL implies that sensors measure system state at some time step in the interval $[k, k + p_i)$. Thus, the time difference between two sampling instances can be greater than p_i.

In RTPL, considering a sampling period p_i for control loop ℓ_i, the maximum interval between any two sampling instances is $2p_i$. The cost function of control loop ℓ_i assuming the control loop samples with a fixed sampling interval of $\hat{p}_i = 2p_i$ is shown below in Eq. (11). The cost function in Eq. (11) introduces pessimism since the control loop samples the system more frequently (decreasing the actual control cost). Thus, Eq. (11) serves an upper bound of the cost function. We propose to adopt Eq. (11) for calculating the list of feasible periods.

$$\begin{aligned} J_i\big(x_i(k), \hat{p}_i, U_i\big) &= \frac{\sigma_i}{\hat{p}_i} + \\ &\Big(\sum_{r=0}^{\lceil \frac{\tau}{\hat{p}_i} \rceil} \sum_{j=0}^{\hat{p}_i - 1} \big(\|x_i(k + r.\hat{p}_i + j)\|^2_{Q_i} + \ \|u_i(k + r.\hat{p}_i)\|^2_{R_i}\big)\Big) + \\ &\Big(\sum_{r=0}^{\infty} \sum_{j=0}^{\phi_i - 1} \big(\|x_i(k + \lceil \frac{\tau}{\hat{p}_i} \rceil \hat{p}_i + r.\phi_i + j)\|^2_{Q_i} + \\ &\|u_i(k + \lceil \frac{\tau}{\hat{p}_i} \rceil \hat{p}_i + r.\phi_i)\|^2_{R_i}\big)\Big) \end{aligned} \quad (11)$$

Considering L_i represents the maximum latency (or response time) of control loop ℓ_i, $L_i < p_i \forall \ell_i$ serves as a schedulability condition for all control loops. The scheduling-control co-design of multiple control loops is shown below

$$\begin{aligned} &\textbf{Minimize}_{\{\hat{p}_i, U_i(\hat{p}_i)\}} \sum_{\forall \ell_i} J_i\Big(x_i(k), \hat{p}_i, U_i(\hat{p}_i)\Big) \\ &\textbf{s. t.} \quad L_i \leq \frac{\hat{p}_i}{2} \ \ \forall \ \ i \end{aligned} \quad (12)$$

where $J_i(x_i(k), \hat{p}_i, U_i(p_i))$ is given by Eq. (11).

In the co-design formulation, the cost function of the multiple control loop case is a sum of individual control loops, and the cost function for a control loop is identical to that of the single control loop case. Thus, adopting Lemma 2, the scheduling-control co-design formulation can be re-written as shown below.

$$\textbf{Minimize}_{\hat{p}_i} \sum_{\forall \ell_i} \frac{\sigma_i}{\hat{p}_i} + \|x_i(k)\|^2_{W_i^{(\hat{p}_i)}} \qquad \textbf{s. t.} \quad L_i \leq \frac{\hat{p}_i}{2} \ \ \forall \ \ i$$

The stability proof of the above equation follows Theorem 1. The feasible periods $\hat{p}_i \in P_i$ that result in asymptotic stability at origin must obey Eq. (13).

$$\begin{aligned} &(A^{(\hat{p}_i)} - B^{(\hat{p})}V^{(\hat{p})})^T W^{(\phi_i)}(A^{(\hat{p})} - B^{(\hat{p})}V^{(\hat{p})}) \leq \\ &(1-\epsilon)W(\hat{p}). \end{aligned} \quad (13)$$

Since the control loop operates at $p_i = \frac{\hat{p}_i}{2}$, the feasible periods are limited to even numbers. Given the set of feasible periods, the period selection algorithm can try all possible combination of periods to find the period selection that minimizes the total cost of the plant and ensures schedulability.

Ensuring the schedulability of all control loops for a set of periods is time-consuming. In the next few sections, we propose a polynomial time solution for testing schedulability.

D. Schedulability Analysis within a Partition

Schedulability analysis for industrial wireless networks has been well-studied for half-duplex radios. These works adopt existing processor schedulability analysis by mapping the communication on a channel to the execution of a task on a processor. For the analysis, a unique communication path is analogous to a core on a processor, and packet transmission on a channel in one time slot is equivalent to a task execution on a core for one time unit. Existing works assume (1) uplink and downlink communication happen on the same communication path and (2) nodes and gateway can transmit to or receive from at most one node. LoRa uses separate uplink and downlink communication channels, and the gateway simultaneously receives and transmits packets, precluding the direct adoption of existing schedulability analysis.

RTPL adopts an EDF scheduler for scheduling packet transmissions on the uplink and downlink communication paths. Thus, high-priority packets can delay a low-priority packet on both the uplink and downlink communication path. Typically, analyzing the interference on the uplink and downlink communication paths relies on a response-time analysis. Response-time calculation is a pseudo-polynomial time algorithm, that complicates the proposed period selection algorithm. To simplify the period selection algorithm, we propose a sufficient utilization-based schedulability condition for RTPL.

To develop a sufficient utilization-based test, we analyze RTPL scheduling under the assumption that the ordering of packet transmissions remains the same for the uplink and downlink communication, which we refer to as RTPL-Uplink-EDF. Specifically, in RTPL-Uplink-EDF, the uplink communication schedule follows an EDF schedule, while the downlink schedule follows the uplink schedule with added offsets (instead of a local EDF). The offsets in the downlink schedule account for the time delay to successfully receive an uplink packet and generate a control command before transmitting a downlink packet. RTPL-Uplink-EDF scheduler purports to restrict the interference from high-priority packets onto one communication path, facilitating a mapping to the processor domain for a utilization-based test.

With the ordering preserved between uplink and downlink communication, we estimate the worst-case offset between a control loop's packet transmission on the uplink and downlink channel. We then map the offset in the downlink schedule to a ghost task and communication on the downlink schedule to a uniprocessor schedulability analysis to generate a sufficient schedulability test.

Lemma 3. *A set of control loops $\pi(UCP_j)$ assigned to uplink communication path UCP_j and downlink communication path DCP_j are said to be schedulable under RTPL scheduling if they are schedulable under RTPL-Uplink-EDF.*

Proof. A packet transmission schedule generated by RTPL-Uplink-EDF is transformable to RTPL schedule by a series of packet transmission swaps, similar to EDF optimality proof. Thus, a task set schedulable under RTPL-Uplink-EDF is also schedulable under RTPL-EDF. □

Upon restricting the ordering of packets for downlink communication, we define critical interval for RTPL scheduling

Definition 1 (Critical Interval of RTPL)**.** In RTPL and RTPL-Uplink-EDF, the critical interval is defined as the interval starting with an uplink transmission that does not overlap with any downlink communication and ends in an idle time slot with no uplink or downlink communication or the start of the next critical interval.

Critical interval of RTPL scheduler is similar to a critical instance in the real-time processor scheduling theory. Following Definition 1, the start of a critical interval may follow the last downlink communication of the previous critical interval. Since there is no overlap between critical intervals, packets from one critical interval cannot delay a packet of another critical interval. Therefore, a critical interval bounds the delay experienced by a packet.

Observation 1. *[Uplink communication in the first time slot of the critical interval] An uplink communication marks the start of a critical interval.*

Observation 2. *[No downlink Communication in the first time slot of the critical interval] At the start of the critical interval, there is no downlink communication. The first downlink communication happens after the uplink communication.*

Within a critical interval, the first downlink communication starts after the first uplink communication completes. The downlink communication schedule within a critical interval is offset by the communication time requirement of the first uplink communication.

Observation 3. *[Upper Bound of the Offset for Downlink Communication within a Critical Interval] The maximum offset for downlink communication is the maximum uplink communication time requirement of all control loops, i.e.,* $\max_i \mathbb{C}_i^{up}$.

For RTPL and RTPL-Uplink-EDF, a control loop's communication time requirement comprises of communication time requirement on the uplink channel ($\mathbb{C}_i^{up}$) and communication time requirement on the downlink channel ($\mathbb{C}_i^{down}$). The total communication time requirement is computed as shown below.

$$\mathbb{C}_i = \mathbb{C}_i^{up} + \mathbb{C}_i^{down}$$

Due to the parallel uplink and downlink communication, the total communication time requirement of control loop ℓ_i does not contribute to the length of the critical interval. Instead, the maximum of the uplink and downlink communication time requirement contributes to the length of the critical interval.

Lemma 4. *A set of control loops $\pi(UCP_j)$ assigned to uplink communication path UCP_j and downlink communication path DCP_j are said to be schedulable under RTPL-Uplink-EDF scheduling if* $\frac{\max_i \mathbb{C}_i^{up}}{\min T_i - 1} + \sum_{\forall \tau_i \in \tau} \frac{\max \mathbb{C}_i^{up}, \mathbb{C}_i^{down}}{T_i} < 1$

Direct Proof. To develop a schedulability test, we map the RTPL-Uplink-EDF schedule's critical instant to a uniprocessor system's critical interval. The combination of uplink and downlink communication paths represents one core. In processor mapping, a unit time of the processor corresponds to the minimum number of time slots required for uplink communication on the partition. A control loop ℓ_i allocated to the partition ($\pi(UCP_j)$) maps to a task execution τ_i executing on a core. The period of task τ_i is the same as the period of ℓ_i. The worst-case execution time of τ_i is equal to the maximum of the uplink and downlink communication time requirements, i.e., $WCET_i = \max(\mathbb{C}_i^{up}, \mathbb{C}_i^{down})$.

For RTPL-Uplink-EDF, we model the offset for downlink communication as a ghost task τ_{ghost} that delays all communications. From Observation 3, the ghost task's worst-case execution time is given by $WCET_{ghost} = \max_i \mathbb{C}_i^{up}$. The ghost task executes every critical instant, which is the smallest

period of all control loops. The ghost task must execute with the highest priority within the critical interval. Therefore, the ghost task must execute with a period $T_{ghost} = (\min_i T_i) - 1$.

Given the mapping and uniprocessor EDF utilization-based schedulability test, the schedulability of RTPL-Uplink-EDF is given by the following equation.

$$\frac{\max_i \mathbb{C}_i^{up}}{\min_i T_i - 1} + \sum_{\forall \tau_i \in \tau} \frac{\max(\mathbb{C}_i^{up}, \mathbb{C}_i^{down})}{T_i} < 1$$

□

Theorem 2. *A set of control loops $\pi(UCP_j)$ assigned to uplink communication path UCP_j and downlink communication path DCP_j are said to be schedulable under RTPL scheduling if*

$$\frac{\max_i \mathbb{C}_i^{up}}{\min_i T_i - 1} + \sum_{\forall \tau_i \in \tau} \frac{\max(\mathbb{C}_i^{up}, \mathbb{C}_i^{down})}{T_i} < 1$$

Proof. The proof of this theorem follows Lemma 4 and Lemma 3. □

E. Schedulability for RTPL

RTPL adopts a worst-fit paritioning algorithm for parition-ing control loops. The worst-fit approach checks whether a set of control loops are schedulable within a partition using Theorem 2. Only upon ensuring schedulability, control loops are assigned to a partition. Additionally, the worst-fit heuristic is a polynomial time solution. Thus, worst-fit heuristic provides a sufficient test of schedulability, wherein a set of control loops are schedulable if the heuristic finds a partitioning.

F. Period Selection Algorithm

Selecting and dynamically adjusting the periods of control loops is critical to the scheduling-control co-design framework. The selected periods must guarantee schedulability (Sec. IV-E) and successful partitioning of the loops. In this section, we present an algorithm for selecting periods for control loops that can ensure stability and schedulability of all control loops.

During deployment, a set of feasible periods $p_i \in P_i$ that ensures stability are determined for each control loop using Equation (13). From these feasible periods, the one that leads to the lowest control cost (Eq. (11)) is chosen as the period for the control loop. If the selected periods result in a schedulability of control loops, they are assigned to partitions. However, if the loops cannot be scheduled, periods of control loops are iteratively updated to find a valid period assignment that ensures stability and schedulability of all control loops.

In each iteration, the algorithm modifies the period of all control loops by incrementing them by a small value denoted as the *update size (Z)* , selected by the system designer. The control cost for each control loop is recomputed using the modified periods, resulting in a *new control cost*. The difference between the new and old control costs, denoted as Δ, is calculated for each loop. The algorithm identifies one control loop whose period adjustment results in the smallest increase in the control cost. The period adjustment of the identified control loop is accepted, and all other adjustments are rejected. Upon updating one control loop's period, the schedulability of the system is verified. If the system is schedulable, the algorithm terminates. Otherwise, the algorithm iteratively updates the period until it finds a valid period assignment. Note that the algorithm may fail to find a valid period assignment when each control loop's maximum feasible period (ensuring stability) does not ensure schedulability. In such case, the system designer may redesign the network to ensure stability and schedulability.

G. Handling Re-Partitioning of the Nodes

Addition or deletion of nodes to the network, or a change in sampling rates may trigger the partitioning algorithm. Moreover, disseminating partitioning information to all control loops frequently will consume huge amount of energy. It becomes highly time consuming specially when the number of nodes is large. To achieve scalability and energy-efficiency of the nodes, we propose to run the optimization less frequently, periodically after a long interval, called **schedule epoch**. Nodes are re-partitioned after each epoch period, right after executing the optimization problem and thus any change in network dynamics is taken care of periodically. Additionally, executing optimization less frequently ensures maximum sleep time of the loops to save energy. As a result, a longer schedule epoch length leads to lower sampling rates and lower energy consumption. Conversely, shorter epoch lengths increase power consumption due to the more frequent need to solve the optimization problem.

A schedule epoch and a sync period serve distinct functions. A schedule epoch is a longer interval after which the optimization problem is *globally* executed for all control loops, leading to re-partitioning of loops. In contrast, a sync period is a shorter interval after which only the optimization is done to compute periods and control inputs of loops within a partition, without any re-partitioning. Multiple sync periods can occur within a schedule epoch, but the partitioning algorithm is only run at the start of each epoch. After the initial execution, the partitioning algorithm is not run again following any period update after the sync period which ensures the energy efficiency of the nodes. The lengths for both the schedule epoch and sync period can be determined through simulation.

H. Handling Time-Synchronisation of the Nodes

In the scheduling-control co-design framework, only the nodes in a partition need to be time-synchronized. The sync period helps in synchronizing the time of all nodes. After each sync period, the controller computes (and broadcasts) periods of (to) all control loops. All the nodes within a partition wake up after the sync period to receive the updated periods. We propose to broadcast the packets with a longer preamble to allow the nodes to wake up before the start of actual transmission. The length of the preamble may be tuned based on the synchronization requirements and the number of nodes.

V. SIMULATION

In this section, we evaluate our control co-design approach through simulation.

Setup: We use the NS-3 LoRaWAN module[30] to model the network in our simulation and a custom script to emulate

physical control system. We used up to 350 nodes and a single gateway. The locations of the nodes were randomly selected within a range of 5 km from the gateway. The nodes and the gateway use 8 channels with center frequencies between 902.3 and 903.7 MHz and bandwidth of 125 kHz. To comply with US regulations, we exclude SF11 and SF12 from our usage. We use 36 UCPs and 16 DCPs. We set the design variables $A = 1$, $B = 1$, $P = 1.1571$, $Q = 1$ and σ= 1. We set γ_i to 50s i.e., $P_i = [1s, 2s, \cdots, 50s]$ and the update size Z to 1s. The total run time of the simulation is calculated by multiplying the total number of schedule epochs taken during the simulation run, the number of sync periods within each schedule epoch, and the duration of a single sync period.

Baselines: We compare the proposed scheduling-control co-design approach against two approaches namely Fixed Periods and Holistic Control. In the *Fixed Periods* approach, the periods are not updated, i.e., the sampling periods are fixed for the entire simulation. In the *Holistic Control* approach, proposed in [25], the *Rate adaptation* algorithm adapts the sampling rate of controllers dynamically based on the control performance, measured by the Lyapunov function V. It compares the Lyapunov function to two thresholds- an increasing threshold $V_I th$, and a more stringent decreasing threshold $V_D th$. If V stays below $V_D th$ for a time interval, indicating good control performance, the period doubles to save energy. If V exceeds $V_I th$, the period halves (with the initial period being the smallest feasible value) to improve control.

Metrics: We evaluate the control cost over time with different numbers of nodes, schedule epochs, and sync periods. Furthermore, we evaluate the *energy consumption* of the nodes which is defined as the amount of energy expended by each node to transmit the state and control information from the sensor to the actuator and is measured in milli-Joules.

A. Results under varying number of nodes

In this simulation, we set the total number of schedule epochs to 2, the number of sync periods within each schedule epoch to 10 and the duration of a single sync period to 100s. Initially, we set the number of nodes to 50 and assess the control cost. Subsequently, we increase the number of nodes to 200 and 350, and evaluate the control cost for a total simulation time of 2000 seconds.

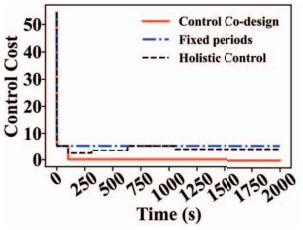
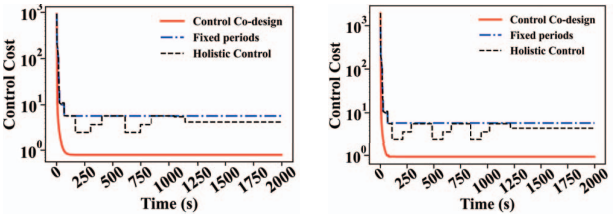


Figure 2. Results under varying number of nodes (50 nodes)

1) 50 nodes: In Fig. 2, for our co-design approach, we notice a swift decline in control cost within the initial 100s, stabilizing at a constant value of 0.20, indicating fast system convergence. In contrast, the control cost for the fixed periods approach stabilizes at 5.47, significantly higher than our approach. This is due to the nodes using fixed periods throughout without accounting for changes in the system state, leading to poor control performance. Additionally, we observe that the control cost for the holistic control approach fluctuates between 200s and 700s as the system sampling rates multiple times based on the threshold to achieve stability. It then gradually decreases, with the system becoming stable after 1400s with a control cost of 4.20.



(a) 200 nodes (b) 350 nodes
Figure 3. Results under varying number of nodes.

2) 200 and 350 nodes: Now, we vary the number of nodes to 200 and 350 and evaluate the control cost in Fig. 3 under the same setup. As the number of nodes in the network increases, we notice a corresponding spike in the initial control cost. Therefore, to better visualize the range of values, we plot the control cost on a logarithmic scale. In Fig. 3(a), the initial cost for 200 nodes is 931.08 for our approach and 938.05 for both fixed periods and holistic control. In Fig. 3(b) the initial cost for 350 nodes is 1917.48 for our approach and 1988.09 for both fixed periods and holistic control. We observe a similar trend of control cost where it gradually decreases after first 100s. In fact all the approaches tend to stabilize after this point by trying to increase the inter-sampling times. Our approach achieves stability with a very low control cost of 0.80 and 0.93 for 200 and 350 nodes respectively. This consistent performance demonstrates that our approach maintains a lower control cost compared to the two baseline approaches. This indicates that our co-design approach can deliver good control performance even at a large scale as it relies on dynamic sampling based on system state. On average, our approach outperforms the baseline approaches by 80%.

3) Energy consumption of 200 nodes: In this simulation, we present the energy consumption of 200 nodes under the same setup in Fig. 4. Our co-design approach aims to reduce the energy consumption of each node while dynamically updating the sampling rates, which also aids in improving network lifetime. Introducing disturbance in the system by adding impulse noise of 0.02 magnitude at every 5s, we observe that the median energy consumed for our approach is 15.73035 mJ (half of the nodes consume less, and half consume more). For holistic control, the median energy consumption per packet is 26.8804 mJ, indicating higher average energy consumption compared to our approach. Our approach shows some outliers at approximately 100 mJ, indicating some nodes consume significantly more energy than others. In contrast, holistic control has several outliers, all above the upper quartile, suggesting multiple nodes with significantly higher energy consumption. Fixed periods has the highest median energy consumption at

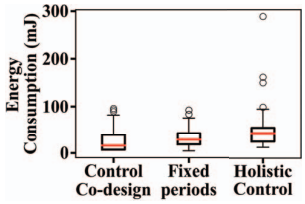
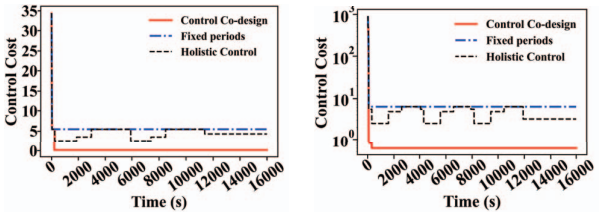


Figure 4. Energy consumption



(a) 50 nodes (b) 200 nodes

Figure 5. Results under varying schedule epoch and sync period

28.4076 mJ but with relatively few outliers above 100 mJ. Overall, our approach is more energy-efficient, achieving good control performance with less energy consumption, even in the presence of noise. Our approach performs approximately 44.61% better than the fixed periods approach and 41.49% better than the holistic control approach based on median.

B. Results under varying Schedule Epochs and sync periods

In this simulation, we vary the number of schedule epochs and the length of sync periods to evaluate the control cost in Fig. 5. We change the total number of schedule epochs to 4, set the number of sync periods within each schedule epoch to 20, and the duration of a single sync period to 200s. We evaluate the control cost for 50 nodes for a total simulation time of 16000s in Fig. 5(a). We observe that increasing the number of sync periods leads to a reduced initial cost compared to previous setups, as dynamic sampling occurs more frequently but still remains under control. The costs for each approach gradually decrease as the system stabilizes, with our approach stabilizing at a very low control cost of 0.20, the fixed periods approach at 5.47, and the holistic control approach oscillating between high and low control costs before stabilizing at 4.30.

In Fig. 5(b), we evaluate the control cost for 200 nodes. We observe a spike as the number of loops increases. To better visualize the range of values, we plot the control cost on a logarithmic scale. Initially, the control cost is high for each approach: 920.08 for fixed periods, 913.02 for holistic control, and 600.00 for our approach. Our approach quickly stabilizes with a cost of only 0.60, even over a long run time, due to dynamically updating periods based on the system state after each schedule epoch and sync period. However, fixed periods stabilize at cost of 6.10, while holistic control at 4.78. This result demonstrates the effectiveness of our approach of periodically updating sampling periods after sync periods and schedule epochs for achieving good control performance.

VI. Conclusion

In this paper, we have proposed a highly energy-efficient and scalable framework for real-time scheduling and control co-design for a LoRa network. To our knowledge, this is the first work on control performance optimization over a LoRa network. Simulations based on NS-3 show that our approach minimizes control cost at least by 80% as compared to the considerable baselines.

Acknowledgement

The work was supported by NSF through grants CNS-2301757, CAREER-2306486, CNS-2306745, CNS-2211640 and by ONR through grant N00014-23-1-2151.

References

[1] E. T. O. Field, https://en.wikipedia.org/wiki/East_Texas_Oil_Field.
[2] "WirelessHART sys. eng. guide," https://www.emerson.com/documents/automation/emerson-wirelesshart-system-engineering-guide-en-41252.pdf.
[3] "WirelessHART," https://www.fieldcommgroup.org/technologies/hart.
[4] "ISA100:," http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891.
[5] L. Bhatia, I. Tomić, A. Fu, M. Breza, and J. A. McCann, "Control communication co-design for wide area cyber-physical systems," *ACM TCPS'21*.
[6] L. Leonardi, F. Battaglia, and L. Lo Bello, "Rt-lora: A medium access strategy to support real-time flows over lora-based networks for industrial iot applications," *IEEE IoT J.*, 2019.
[7] S. Fahmida, V. P. Modekurthy, D. Ismail, A. Jain, and A. Saifullah, "Real-time communication over lora networks," in *IEEE/ACM IoTDI'22*.
[8] M. Hatler, D. Gurganious, and J. Kreegar, *Industrial LPWAN – A Market Dynamics Report*, https://www.onworld.com/iLPWAN/index.html.
[9] "LoRaWAN," https://www.lora-alliance.org.
[10] "LoRa design guide," https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf.
[11] 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 Trans. Embed. Comput. Syst.*, 2014.
[12] Y. Ma, J. Guo, Y. Wang, A. Chakrabarty, H. Ahn, P. Orlik, X. Guan, and C. Lu, "Optimal dynamic transmission scheduling for wireless networked control systems," *IEEE Trans. on Cont. Sys. Tech.*, 2022.
[13] F. Mager, D. Baumann, C. Herrmann, S. Trimpe, and M. Zimmerling, "Scaling beyond bandwidth limitations: Wireless control with stability guarantees under overload," *ACM TCPS*, 2022.
[14] V. P. Modekurthy and A. Saifullah, "Online period selection for wireless control systems," in *IEEE ICII'19*.
[15] Y. Ma, Y. Wang, S. Di Cairano, T. Koike-Akino, J. Guo, P. Orlik, X. Guan, and C. Lu, "Smart actuation for end-edge industrial control systems," *IEEE Trans. on Automation Sci. and Eng.*, 2022.
[16] https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/.
[17] D. Roy, C. Hobbs, J. H. Anderson, M. Caccamo, and S. Chakraborty, "Timing debugging for cyber-physical systems," in *DATE'21)*.
[18] Y.-Q. Xia, Y.-L. Gao, L.-P. Yan, and M.-Y. Fu, "Recent progress in networked control systems—a survey," *International J. of Automation and Computing*, 2015.
[19] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Comm. Surv. & Tut.*, 2018.
[20] K. Tsumura, H. Ishii, and H. Hoshina, "Tradeoffs between quantization and packet loss in networked control of linear systems," *Automatica*, 2009.
[21] J. Araújo, M. Mazo, A. Anta, P. Tabuada, and K. H. Johansson, "System architectures, protocols and algorithms for aperiodic wireless control systems," *IEEE TII*, 2014.
[22] M. Mazo and P. Tabuada, "Input-to-state stability of self-triggered control systems," in *IEEE CDC/CCC 2009*, pp. 928–933.
[23] E. Henriksson, D. E. Quevedo, E. G. W. Peters, H. Sandberg, and K. H. Johansson, "Multiple-loop self-triggered model predictive control for network scheduling and control," *IEEE Trans. on Cont. Syst. Tech.*, 2015.
[24] 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 *IEEE/ACM ICCPS'12*.
[25] Y. Ma and C. Lu, "Efficient holistic control over industrial wireless sensor-actuator networks," in *IEEE ICII'18*, pp. 89–98.
[26] Y. V. Pant, H. Abbas, K. Mohta, T. X. Nghiem, J. Devietti, and R. Mangharam, "Co-design of anytime computation and robust control," in *IEEE RTSS'15*, pp. 43–52.
[27] J. M. Maciejowski, *Predictive Control With Constraints*. Prentice Hall, 2002.
[28] J. H. Lee, "Model predictive control: Review of the three decades of development," *International J. of Cont., Auto. and Sys.*, 2011.
[29] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012.
[30] D. Magrin, M. Capuzzo, and A. Zanella, "A thorough study of lorawan performance under different parameter settings," *IEEE IoT J.*, 2019.