

Online Period Selection for Wireless Control Systems

Venkata P. Modekurthy and Abusayeed Saifullah
Department of Computer Science, Wayne State University, Detroit, MI, USA

Abstract—Recent advancements in Industrial Internet-of-Things and cyber-physical systems through the development of wireless standards like WirelessHART and ISA100 for real-time and reliable communication paved the way for a new industrial trend called *Industry 4.0*. Industry 4.0 proposes to improve production efficiency by employing smart factories. One approach to improve the production efficiency is to predict the external disturbances and adjust the sampling periods accordingly. In a wireless control system, an online adjustment of the sampling periods can decrease the energy consumption of nodes, especially when the system is within a stable state. Although adjusting the sampling period is beneficial, the stability of the system may be compromised due to external disturbances through an increase in sampling period while decreasing the sampling period can impact the real-time performance and energy of the wireless network. Existing work on online sampling period selection assumes that the controller computes the periods and schedules, and repeatedly broadcasts the new sampling periods and schedules to all the nodes. Such an approach is highly energy consuming and can impact control performance. In contrast, to handle online period selection, we propose an autonomous scheduler based on game theory, where each node in the network generates the schedules locally and without any communication with the others. Our approach can handle any changes in the period and link qualities locally and on the fly. We also propose a heuristic for online sampling period assignment, where each node predicts the state and adjust the period locally. Our evaluation on a case study shows that the proposed approach consumes at least 59% less energy when compared to the state-of-the-art approach.

I. INTRODUCTION

Industrial Internet-of-Things (IoT) and industrial Cyber-Physical Systems (CPS) evolved from wireless standards like WirelessHART [2] and ISA100 [1] that facilitate low-power, flexible, and cost-efficient real-time communication for a broad range of applications like process control [9], smart manufacturing [34], smart grid [8], and data center power management [33], [32]. These wireless standards facilitate closed loop real-time communication between sensors and actuators, where each sensor measures the state of a plant and delivers it to a controller through a wireless network. The controller generates control commands based on the measured state and then sends the control commands to each actuator through a wireless network. To facilitate reliable and real-time communications in a shared wireless environment, industrial wireless standards employ a Time Division Multiple Access (TDMA) based Media Access Control (MAC) protocol with a high degree of redundancy.

Industrial IoT and CPS paved the way for a new industrial trend under the specification of Industry 4.0 [40]. It is a new concept of Internet technologies for industries to improve

production efficiency by employing smart factories [37]. One approach to enhance the production efficiency of the plants is to predict the external disturbances and to adjust the sampling period accordingly. Such an online sampling period selection enables the controller to counteract the disturbances promptly, efficiently, and without affecting the stability. Furthermore, an online sampling period selection reduces the energy consumption of the sensors and actuators.

Adjusting the sampling rate of a plant can have undesirable consequences in a wireless control system in which many control loops share the same wireless communication medium. Particularly, selecting a high sampling rate for one control loop may affect latency, real-time performance, and stability of the others. Selecting a low sampling rate can affect the stability of the plant. Therefore, sampling rates should balance real-time communication and control performance, requiring a control and communication co-design approach.

An online sampling rate assignment has to take into account the stability of each plant and the schedulability of each control loop in the network. Typically, existing work formulates this as a non-linear optimization problem with network schedulability and stability constraints [14], [10], [24]. The non-linear optimization problem is solved at the controller, and the period and schedule are disseminated to the nodes periodically. Such an approach can lead to high energy consumption and disrupt the sensing and actuation messages [21]. The work in [17] placed restrictions on the increase in the sampling period to avoid re-dissemination of a new schedule. Although intuitive, this approach can consume more energy due to the unavailability of a period meeting the restrictions. Another approach to avoid re-dissemination of a schedule is to use any existing distributed scheduler [21], [43], [42], [7]. Although these approaches can handle changes in the sampling period, they have the following limitations. First, they assume a simple link quality model, where the link quality at the current slot is independent of that observed in the past time slots. Second, dissemination of the sampling periods can cause energy overheads in the system.

To address the above limitations, we propose an autonomous period selection algorithm for wireless control systems. Our approach is based on selecting a maximum period to ensure a decrease in the Lyapunov function for stability. To handle online period selection, we also propose an autonomous scheduler for wireless control systems using game theory. The objective of the game is to maximize the number of successful transmissions by avoiding collision and transmissions during poor link quality. Our evaluation on a case study shows that

the proposed approach consumes at least 59% less energy compared to the holistic controller proposed in [17], which is the state-of-the-art approach.

In summary, we make the following contributions in this paper.

- We formulate the autonomous TDMA scheduler as a non-cooperative game. The players choose a strategy that yields maximum payoff for all players, and hence, each node converges to a schedule without communicating with the others. The payoff of the game represents the number of successful transmissions, and the objective of the game is to maximize the payoff. We also provide a linear-time sufficient schedulability analysis for the proposed autonomous scheduler.
- We propose an online sampling period assignment algorithm, where each node autonomously selects the sampling period of all control loops by predicting the state of the system. Based on the current state, it predicts the sampling period required to ensure a non-increasing Lyapunov function for stability.

The rest of the paper is organized as follows. Section II reviews related work. Section III describes the system model. Section IV describes the autonomous scheduler. Section V describes the dynamic period selection. Section VI presents the evaluation and Section VII concludes the paper.

II. RELATED WORK

Real-time routing, real-time scheduling, and schedulability analysis are investigated in [9], [20], [29], [19], [22], [6], [45], [34], [11], [21], [43], [42]. They assume a centralized or distributed scheduler, which consumes energy at the nodes. The work in [7] proposes an autonomous scheduler for sparse traffic scenarios to address this issue. However, it assumes a fixed sampling period. In contrast, our work proposes an autonomous period selection and an autonomous scheduler based on game theory. Although there exist several works that use game theory for wireless networks, they do not consider real-time scheduling [35].

Scheduling-control co-design for wired networked control systems was studied in [28], [12], [15], [3], [44]. In contrast, we consider a wireless network. Wireless control co-design was investigated for single-hop networks in [41], [26] (also see survey [23]), and for multi-hop networks in [38]. They assume each sensor samples the plant at fixed periods. Such static sampling is not a good approach for dynamic systems. The work in [4] proposes a self-triggered control with network schedulability constraints based on a virtual link capacity margin. It uses a simplistic network model where it does not avoid collisions in the network. In contrast, our autonomous scheduler avoids collisions.

The work in [14], [10] proposes stability under a self-triggered control with a centralized scheduler. A centralized schedule has to broadcast the entire schedule after every change in period. Thus, a centralized scheduler usually causes 1) high energy overheads and 2) frequent disruptions to the control operation. To address this limitation, the work

in [17] proposes a self-triggered control where the periods only increase in integer multiples of the initial sampling period. In cases where the actual increase in the period needed for stability is small, the approach in [17] does not change the period, thereby making no improvement in energy consumption. It does not decrease the sampling period when the network is underutilized.

All the approaches discussed above use a centralized period selection algorithm, which still requires frequent broadcast of sampling periods. In contrast, we propose an autonomous period selection algorithm where each node selects a period locally and without communicating with others. Furthermore, we propose an autonomous scheduler where nodes can adapt to the sampling period changes without any energy overheads.

III. SYSTEM MODEL

A. Network Model

In this paper, we consider an industrial wireless sensor-actuator network (WSAN), which consists of field devices, an access point, and a gateway. The *field devices* are wirelessly networked sensors and actuators. Each node contains a *half-duplex* omnidirectional radio transceiver that can receive from at most one sender at a time. Furthermore, a node cannot both transmit and receive at the same time. *Access point* provides a path between the wireless network and the gateway. The *network manager* and the controller remain at the *gateway*. The network employs feedback *control loops* between sensors and actuators. Sensors measure process variables and deliver them to the controller through the network. The controller sends control commands to the actuators, which then operate the control components to adjust physical processes. In this paper, we assume that the role of the network manager is to create network topology, observe the network operation, and determine offline admission control for new control loops/flows. Note that the network manager does not generate a schedule.

We consider a WSAN where all sensors/actuators directly communicate with the access point. That is, sensors, actuators, and the access point form a star topology. Many existing WirelessHART and ISA100 deployments use such single-hop topology for technical simplicity. Furthermore, recent low-power wide-area network technologies like LoRa [25], SigFox [5] and SNOW [31], [27], [30] all adopt a single-hop network topology. Transmissions in the network are scheduled based on a single channel TDMA protocol. Time in the network is globally synchronized and slotted. A transmission and its acknowledgment happen in one slot. Transmissions are scheduled based on the link quality. We use the signal to noise ratio (SNR) to determine the link quality.

B. Control System Model

We consider there are n real-time control loops in the system denoted by $F = \{F_1, F_2, \dots, F_n\}$. For the rest of the paper, we use the terms flow and control loop interchangeably. Each flow corresponds to a linear time-invariant (LTI)

control system, which is expressed in its discrete state space expression, as shown in Equation (1).

$$\begin{aligned} x_i(t+1) &= A_i x_i(t) + B_i u_i(t) + v_i(t) \\ y_i(t) &= C_i x_i(t) \end{aligned} \quad (1)$$

In Equation (1), $x_i(t)$ represents the state of control system i at t -th time slot, $u_i(t)$ represents the control input, $y_i(t)$ represents the observed output, and $v_i(t)$ represents the discrete time white Gaussian noise. We assume the pair (A_i, B_i) is controllable, and the pair (A_i, C_i) is observable.

The period and deadline of the flow F_i are represented by T_i and D_i , respectively. We consider an implicit deadline system, where the deadlines are equal to the periods, i.e., $D_i = T_i$. Since we consider the sampling period changes dynamically with the system, we also consider that the deadline of the packet also changes dynamically. Nevertheless, the relative deadline of the packet is the same as the current sampling period of the plant. For the rest of the paper, the periods and deadlines are specified in number of time slots.

For each plant, we use a model predictive controller with a cost function given by Equation (2), and a final state $x_f \equiv 0$.

$$\begin{aligned} J_i(t) &= \frac{1}{2} x_i(t+N)^T P_i^f x_i(t+N) + \\ &\sum_{z=t}^{t+N} \frac{1}{2} (x_i^T(z) P_i x_i(z) + u_i^T(z) Q_i u_i(z)) \end{aligned} \quad (2)$$

The cost function is the sum of the weighted average of the state and control input for every time slot in the duration of horizon (N), where P_i and Q_i represent the weight for the state and control input, respectively. In Equation (2), P_i^f is a positive definite matrix that satisfies the Lyapunov equation $A_i^t P_i^f A_i + Q_i^f = 0$, where Q_i^f is also a positive definite matrix. The cost function used in Equation (2) ensures closed loop asymptotic stability of the control loop i [18]. Note that the value of $u_i(t)$ changes only after T_i time slots and is constant in between.

We assume each control loop i has a *weight* w_i . The weight of control determines its importance in ensuring stability. The total cost (inclusive of all control loops) is given by Equation (3).

$$J(t) = \sum_{i \in [i, n]} w_i J_i(t) \quad (3)$$

Given the state space equations and control cost for all plants, we propose an online sampling period assignment that changes dynamically with the state of the system. To handle online sampling period assignment, we first propose an autonomous scheduler that does not incur energy overhead.

IV. LINK QUALITY BASED AUTONOMOUS SCHEDULER

Existing centralized and distributed schedulers for industrial standards like WirelessHART and ISA100 use real-time scheduling policies like EDF (earliest deadline first) and DM (deadline monotonic). However, these schedulers typically assume a simple link quality model where the packet reception rate (PRR) is constant at each time slot. Such a model is

not applicable in practice since link quality varies with time, and link quality at a current time slot depends on that of the previous time slots. In this paper, we consider a general link quality model where link quality varies with time. We use a small neural network to predict the link quality.

For the general link quality model, we develop a new autonomous scheduler where each node determines when to transmit a packet based on the deadline of the packet and link quality. Designing such an autonomous scheduler where all packets meet their deadlines is challenging since all nodes in the network have to compute the best time to transmit all packets, and agree on the schedule without communication. Another important challenge is that the best transmission time slot depends on the dynamically changing link quality.

We model the scheduling problem as a non-cooperative game where each node in the network chooses the Nash equilibrium strategy autonomously and without any coordination. The objective of each node playing the game is to maximize the number of successful transmissions by avoiding deadline misses, collisions, and transmissions on poor links. This formulation is flexible with changing periods while ensuring all nodes converge to the same schedule without communicating with other nodes in the network. Typically, a game can have multiple Nash equilibrium strategies. In the proposed approach, we can break the ties by choosing the strategy which schedules packets with the shortest absolute deadline first. Thus, the game-theoretic autonomous scheduler converges to a solution without any communication.

The proposed game-theoretic scheduler requires link quality information of all links and period/deadlines of all flows to determine the Nash equilibrium strategy (or transmission schedule). In this section, we assume that all nodes are aware of the sampling rate for all flows. We discuss the autonomous sampling rate selection in Section V. In this section, we first discuss the link quality prediction. We then discuss our game formulation and the existence of Nash equilibrium strategies for a single-slot-game. We then discuss how the nodes compute Nash equilibrium strategies in a multi-slot game. Finally, we present the schedulability analysis for the proposed network scheduler.

A. Link Quality Prediction

We use a data-driven prediction using neural networks (ANN), similar to the approach in [16]. A node uses ANN to predict the link quality of η subsequent time slots between itself and a receiver. In this section, we consider η is a design parameter. The choice of an ANN approach arises from the fact that the ANN approach can estimate link quality with greater accuracy [16]. Note that the work in [16] only compares the performance of different approaches but does not develop an autonomous scheduler or autonomous period selection.

The neural network takes the signal strength at the beginning of the time slot as the input. It generates the SNR values of packet transmission and of its acknowledgment for η subsequent time slots as output. A node considers the link quality at a time slot to be good if the SNR for packet transmission

and its acknowledgment are above a pre-determined threshold, and bad otherwise. Note that, the neural network generates the link qualities for η time slots of one link. In a single-hop network, most of the links are positively or negatively correlated to one another [39]. The nodes use κ -factor [39], which is cross-correlation index that depends on channel and power levels, to estimate the link quality of η subsequent time slots for all links in the network. Thus, the proposed approach is not computation-intensive and can execute on a low power embedded device.

We model the ANN as a three-layer neural network with one input layer, one hidden layer, and one output layer. The input layer consists of 1 neuron. The output layer consists of 2η neurons (one to estimate the SNR of packet reception and another to estimate the SNR of the acknowledgment). The hidden layer consists of $2\eta \times 5$ neurons. Note that the proposed neural network setup is similar to that used in [16]. We only increase the number of nodes in the hidden layer and the output layer to predict link quality information for η time slots.

To train the ANN, we use 1) signal strength observed on a channel before making a transmission, 2) SNR obtained at the receiver for each packet reception, and 3) SNR of the acknowledgment packet. Signal strength value at the beginning of the time slot is used as the input to the neural network. SNR values obtained at the receiver and SNR values of packet acknowledgment are used as reference outputs during training. The network manager computes and then broadcasts the weights of the neural network (after the training phase) and the correlation between links to all nodes.

During the data collection phase, a node collects the signal strength information for a short duration at the beginning of a time slot. It then makes η consecutive transmissions to the receiver. Each transmission is followed by an acknowledgment by the receiver. Nodes in the network use a round-robin approach to avoid interference from collisions. Since the autonomous scheduler is designed to avoid collisions, the SNR values from simultaneous transmissions are not required during data collection. The nodes use packet transmissions or piggybacked acknowledgments to send signal strength and the SNR information to the base station.

In this link prediction model, computation-intensive operations take place on the controller, and hence, nodes in the network do not incur significant computation overheads. Furthermore, the controller can train the ANN during design time, which does not interfere with the control command computations. The experimental evaluation performed in [16] using training information from 20000 packets show that the average prediction error for a neural network approach is 0.18.

B. Game Formulation for Autonomous Scheduler

We formulate the game similar to the prisoner's dilemma where we assume all nodes to be non-cooperative. That is, nodes do not communicate with each other to generate a schedule. They use local information to come up with a strategy to maximize their payoff. In our game, we consider each node with a packet as a player of the game.

In a time slot, each node has two strategies: 1) *to transmit* and 2) *not to transmit*. The payoff obtained at each time slot is defined in Equation (4). The objective of the game is to maximize the payoff, which is defined in terms of the number of successful transmissions. The payoff function rewards a successful transmission and penalizes the wastage of energy due to transmission failures in each time slot. It also penalizes a deadline miss for each packet. If a node chooses *not to transmit* a packet, then the payoff is 0. If it chooses *to transmit* a packet and the packet transmission is successful, the payoff is 1. If it chooses *to transmit* a packet and the packet transmission is not successful, the payoff is -1 . A packet transmission may fail due to two reasons: 1) collision from simultaneous transmissions or 2) transmission on a poor link (low SNR).

$$\text{payoff} = \begin{cases} 1, & \text{if a packet transmission is successful} \\ 0, & \text{if node decides not to transmit a packet} \\ -1, & \text{if a packet transmission is not successful} \\ -2, & \text{if a packet misses its deadline} \end{cases} \quad (4)$$

Given the link quality information and sampling period for all control loops, we discuss how a node computes a strategy that leads to a Nash equilibrium for a single-slot-game. We then extend the analysis to a multi-slot-game and show how a node computes the strategy that leads to Nash equilibrium.

C. Single Time Slot Game

To better explain the payoff function and Nash Equilibrium for a single time slot game, we provide a simple example with two nodes: node A and node B. A similar approach can be used to compute Nash equilibrium when the number of nodes is greater than 2. Based on the link quality, we can consider three different scenarios. 1) Link quality for node A (the link between the access point and node A) and link quality for node B (the link between the access point and node B) are bad. 2) Link quality for node A is bad, but link quality for node B is good. 3) Link quality for node A and link quality for node B are good. For the first scenario, Fig. 1(a) shows the payoff values obtained for each strategy played by node A and B. If node A chooses to transmit, the destination does not receive the packet due to poor link quality, and hence, it receives a payoff of -1 . However, if node A chooses not to transmit, it receives a payoff of 0. Thus, the best strategy for node A is *not to transmit* a packet. Similarly, the best strategy for node B is *not to transmit* a packet. Thus, the Nash Equilibrium strategy in this case for both nodes is *not to transmit* a packet.

For the second scenario, when link quality for A is bad, Fig. 1(b) shows the payoff values obtained for each strategy. As shown in Fig. 1(b), the best strategy for node A is *not to transmit* a packet, and hence obtain a payoff of 0. Since node A always chooses *not to transmit* a packet, the best strategy for node B is to transmit a packet and receive a payoff of 1. The Nash equilibrium strategy, in this case, is for node B *to transmit* a packet and for node A is *not to transmit* a packet.

For the third scenario, when link quality for node A and B is good, Fig. 1(c) shows the payoff values obtained for each strategy. As shown in Fig. 1(c), when both nodes choose *to*

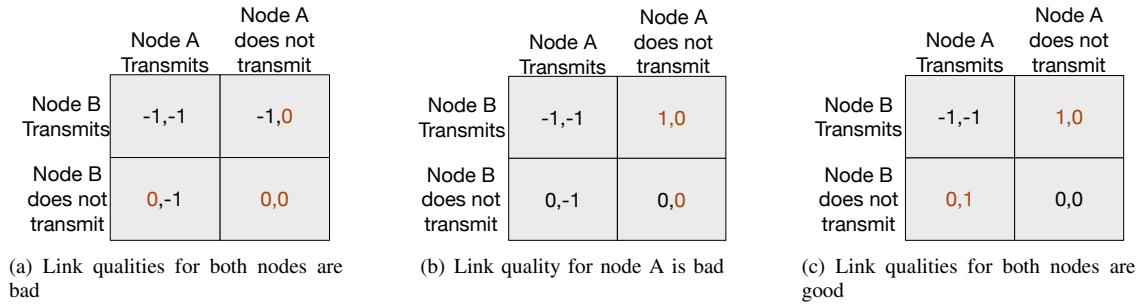


Fig. 1. Payoff values for each strategy in a game with two players.

transmit, it results in a collision, and hence, node A and B receive a payoff of -1 . Similarly, when both nodes choose *not to transmit*, it results in a wastage of time slot for both nodes, and hence, they receive a payoff of 0. When node B chooses *to transmit* and node A chooses *not to transmit*, node B can make a successful transmission. Here, node B gets a payoff of 1, and node A gets a payoff of 0 (since its time slot is wasted). Thus, the best strategy of node B is *to transmit* when node A selects *not to transmit*, and the best strategy of node B is *not to transmit* when node A selects *to transmit*. Thus, there are two Nash equilibria for this scenario. A tie can be broken here based on the shortest absolute deadline. The node with the smallest absolute deadline chooses *to transmit* a packet while the other chooses *not to transmit* a packet.

To explain the effect of deadlines and link qualities on the Nash equilibrium strategies, we consider a fourth scenario. In this scenario, we consider node B's link quality is good, and its packet deadline is 2. We consider node A's link quality is bad, and its deadline is 1. Node A has to transmit in the first slot to meet the deadline. In this situation, if node A chooses to transmit a packet, there is a chance the packet transmission is not successful, and it receives a payoff of -3 (-2 for a deadline miss and -1 for a unsuccessful transmission). If the packet transmission is successful, then node A gets a payoff of 1. However, if node A decides not to transmit a packet, it gets a payoff of -2 . In this case, node A chooses *to transmit* and node B chooses *not to transmit*. Since a node receives a payoff of -2 for every deadline miss, it tries to make at least one transmission for every packet.

In a two player game where both nodes have the same deadlines, there are scenarios where there are two Nash equilibrium strategies. In these scenarios, a mixed strategy Nash equilibrium analysis shows that the probability of switching between the 2 Nash equilibrium strategies is 0.5. We can extend this result to a n player game where the probability of switching between the n Nash equilibrium strategies is $\frac{1}{n}$. An equal probability for all nodes shows that the choice of the payoff values results in a fair usage of network resources between all the players.

D. Multiple Time Slot Game

In our autonomous scheduler, each node can predict the link quality of η time slots. Therefore, each node plays the game for η time slots. The total payoff is computed as the sum of payoffs received in each time slot. For the multi-slot-

game, we assume that each node plays a sub-game perfect equilibrium strategy (which is a Nash equilibrium strategy in every time slot). In case of multiple sub-game perfect equilibrium strategy, a tie can be broken here based on the shortest absolute deadline. In a multi-slot-game, we assume that each flow requires a total of γ (where $\gamma \geq 2$) time slots for both uplink and downlink communications. Of the available γ slots, two are used for transmissions and the rest are used for re-transmissions in case of failures.

In our game, the subgame perfect equilibrium for a η time slot game depends on the link quality conditions and deadlines for each packet. Due to a large number of scenarios (considering different link qualities and deadlines), it can be difficult to theoretically analyze all possible scenarios and come up with a single algorithm that can always generate a Nash equilibrium strategy. Therefore, we propose to use the insights from the single-slot-game to reduce the number of strategies explored.

E. Schedulability Analysis

This section presents the schedulability analysis for the proposed autonomous scheduler. The schedulability analysis determines if the selection of a set of periods will result in all packets meeting their deadlines or not.

Theorem 1. *Under the proposed autonomous scheduler, the set of flows $F = \{F_1, F_2 \dots F_n\}$ with periods (equal to deadlines) $T_1, T_2, \dots T_n$ can meet their deadlines given $p_i T_i$ represents the maximum number of poor link qualities experienced by a flow F_i when Equation (5) is satisfied, where $\mu_i = \frac{\gamma}{T_i(1-p_i)}$.*

$$\sum_{i=1}^n \mu_i \leq 1 \quad (5)$$

Proof. We first show the schedulability analysis when the link quality is good for all links and all slots. We then model the time slots with poor link qualities as blocking time in non-preemptive uni-processor scheduling. We use the non-preemptive schedulability test for the proposed autonomous scheduler.

When the link quality is good for all time slots, each node selects a unique transmission time slot for each packet of a flow based on the shortest absolute deadline (since all link qualities are good). This schedule is the same as the EDF scheduler. Since preemptive uni-processor schedulers consider

that a task preempts at the boundaries of the time unit and not in between, the autonomous scheduler (when the link quality is good) maps to a preemptive scheduler. The schedulability test for an implicit deadline task set under preemptive EDF scheduler is $\sum_{i=1}^n \mu_i^G \leq 1$, where $\mu_i^G = \frac{c_i}{T_i} = \frac{\gamma}{T_i}$.

In the proposed autonomous scheduler, a transmission happens only on a good link quality to avoid wastage of energy. When the link quality for a flow with a shorter absolute deadline is bad, then its transmission is blocked until the link quality is good. Meanwhile, a flow with larger absolute deadline with a good link quality can make a transmission. A similar blocking can be observed in a non-preemptive scheduler where a task with a shorter absolute deadline is blocked by the execution of a larger absolute deadline. Thus, we can map the proposed autonomous scheduler to a non-preemptive EDF scheduler. The schedulability test for non-preemptive uni-processor scheduler is given by $\sum_{i=1}^n \mu_i \leq 1$, where δ is the maximum blocking time, and

$$\mu_i = \frac{C_i}{T_i - \delta} = \frac{\gamma}{T_i - p_i T_i} = \frac{\gamma}{T_i(1 - p_i)}$$

Thus, the schedulability test for the proposed autonomous scheduler under the assumption that $p_i T_i$ represents the maximum number of poor link qualities experienced by a flow F_i is given by Equation (5). \square

V. ONLINE SAMPLING PERIOD SELECTION

Here, we first formulate the centralized version of an online sampling period selection problem. We then present the autonomous sampling period acclimation algorithm, where each node selects the sampling period without communicating with others.

A. Formulation of a Centralized Online Sampling Period Selection Problem

In this paper, we change the sampling period after every τ time slots instead of changing the sampling period after every interval to ensure low overhead. Given the state space equations for each flow and schedulability condition, the online sampling period assignment problem is formulated in Problem (6). To capture the stability of all control loops, we use the controller cost function given by Equation (3) as the objective. We use the schedulability analysis given by Equation (5) as a constraint. Note that the period of flow F_i changes after every τ time slots but remains constant within an interval $(k\tau, k\tau + \tau)$, where $k = 1, 2, \dots, \infty$. Thus, any period selected by Problem (6) results in a schedulable network scenario for the interval of length τ .

$$\begin{aligned} & \arg \min_{T_i \forall i} J(t) \\ & \text{subject to } \sum_{i=1}^n \mu_i \leq 1 \end{aligned} \quad (6)$$

Due to the high complexity of the objective function, finding a solution to Problem (6) can take a long time. Furthermore, the controller must disseminate the sampling periods obtained

by solving Problem 6 to all the nodes in the network, which causes high energy overhead at the nodes. To address this problem, we propose a sampling period acclimation heuristic, which is presented in the following sections.

B. Sampling Period Acclimation

This section presents an approach for sampling period acclimation, where the sampling periods of all plants are varied every τ time slots. Algorithm 1 presents the sampling period acclimation algorithm. In the proposed approach, the central manager initializes all flows with a sampling period. The initial period assignment has to take into account the schedulability and stability of the plants. Since an optimization function considering both constraints can be computation intensive for a low power embedded device, we choose to execute the initial sampling period assignment at the controller. The initial sampling period assignment can act as a fail-safe assignment when all the flows experience high disturbance. Note that the controller executes the initial sampling period assignment only once (i.e., during the system initialization).

Algorithm 1: Sampling Period Acclimation Algorithm

```

Input :  $T_i[initial]$  = Initial-Sampling-Period-Assignment()
Output: Sampling Period  $T_i \forall i$  in  $[0, n]$ 
 $x_i(t)$  = State-Estimation()
 $x_i(t + \tau)$  = State-Estimation()
 $V_i(t) = x_i^T(t)P_i^L x_i(t)$ 
 $V_i(t + \tau) = x_i^T(t + \tau)P_i^L x_i(t + \tau)$ 
 $Inc, Dec$  = Segregate-Flows()
for  $i$  in  $Inc$  do
  |  $T_i$  = Increase-Sampling-Period()
end
for  $i$  in  $Dec$  do
  |  $T_i$  = Decrease-Sampling-Period()
end

```

After every τ time slots, the sampling periods for all plants are re-computed by each node autonomously. A node first estimates the current states and future states after τ time slots (with the current sampling period) of all plants in the network. Based on the current states and the estimated future states, the plants and their respective flows are segregated based on whether their sampling periods increase or decrease. Each sensor first computes the sampling period of all plants with an increase in their sampling period. Each sensor then computes the sampling period of all plants with a decrease in their sampling period. The intuition behind computing increase and decrease in the sampling period separately is to reduce the overhead of computation. An increase in the sampling period may impact the stability of a system. However, it does not impact the schedulability of the flows on the network. Similarly, a decrease in the sampling period impacts the schedulability of the flows, but it may improve the stability of the plant. Thus, splitting the computation of sampling periods can reduce the overhead of calculations on each sensor. The initial sampling period assignment, state estimations, segregation of plants, and the approaches for increasing/decreasing sampling period are described in detail in the following sections.

C. Initial Sampling Period Assignment

The initial sampling period assignment must ensure the stability of plants and schedulability of their respective flows on the network. Stability of a plant can be maintained when the sampling period is within the bounded range of maximum ($T_i[\text{max}]$) and minimum ($T_i[\text{min}]$) sampling periods. Schedulability of all flows can be ensured when the total utilization is no greater than 1, i.e., $\sum_i^N \mu_i \leq 1$. We use these two constraints to develop a heuristic for assigning initial sampling periods.

Algorithm 2 illustrates our heuristic for sampling period assignment. It takes as input the minimum and maximum sampling periods for each flow that ensures acceptable control performance. It starts by assigning the minimum sampling period to all flows. If all flows in the network are schedulable with the selected sampling periods, it stops. If a flow is not schedulable, it iterates over the ordered set of flows (decreasing order of the weighted individual control cost). The maximum sampling period $T_i[\text{max}]$ is selected as the new sampling period for a flow. The algorithm stops when a sampling period assignment leads to a schedulable network. In some rare cases, the algorithm can stop when all flows are assigned their maximum sampling periods, and the network is still unschedulable. In this case, the plants are not stabilizable as there is no feasible sampling period assignment with acceptable control performance. Once the algorithm stops, the central manager broadcasts the sampling period to all nodes.

Algorithm 2: Initial Sampling Period Assignment

Input : $T_i[\text{min}]$ & $T_i[\text{max}] \forall i$ in $[0, n]$
Output: Sampling Period $T_i \forall i$ in $[0, n]$
for i **in** $[0, n]$ **do**
 $T_i = T_i[\text{min}]$
 $\mu_i = \frac{\gamma}{T_i}$
end
 $i = n$
while $\sum_{i=1}^n \mu_i > 1$ **do**
 $T_i = T_i[\text{max}]$
 $\mu_i = \frac{\gamma}{T_i}$
 $i = i - 1$
end

D. State Estimation and Correction

State estimation has been extensively studied in control theory [36]. Kalman filtering is a popular approach for state estimation, which can predict the state of the plant with high accuracy [36]. Kalman filter uses the past state estimation error to accurately predict the current and future state of the plant. Such an algorithm requires the state space representation, current state of the system, control input applied at each time slot, and the noise experienced at each sample to compute the error in state estimate. Since Kalman filtering is mainly used by the controller to compensate for packet delay or loss, the assumption about the availability of the aforementioned data is valid. However, in the proposed approach, a sensor estimates the state of all plants, which does not have all the required data. A sensor can obtain the initial state of each plant and the state space representation during the network initialization

phase. Nevertheless, a sensor does not have the control input applied, and the noise experienced at each time slot. To address this issue, we propose the following approach to estimate the state of the plant.

Each sensor assumes that all plants use a linear controller given by $u_i(t) = k_i x_i(t)$, where k_i is a constant value. The value of k_i satisfies the Lyapunov stability constraint

$$(A_i + B_i K_i)^T P_i^L (A_i + B_i K_i) - P_i^L + Q_i^L = 0 \quad (7)$$

where, P_i^L and Q_i^L are positive definite matrices. A sensor can obtain the values of K_i for all flows, during network initialization. Thus, a sensor can predict the control input applied during the previous step. We use the variance of white Gaussian noise as the past noise in the system. Since the variance is fixed, all nodes use the same value of noise to estimate the state of the plant.

In the proposed approach, the noise estimation in a plant is assumed to be constant. However, in practice, the noise experienced by each plant can be random. The random nature of noise can lead to errors in state estimation. A sensor monitoring the plant associated with flow F_i can compute the state estimation error for the plant based on Equation (8), where $x_i^{\text{obt}}(t)$ is the measured value of the state of plant associated with flow F_i , and $x_i^{\text{est}}(t)$ is the estimated state of plant associated with flow F_i .

$$\xi(t) = x_i^{\text{obt}}(t) - x_i^{\text{est}}(t) \quad (8)$$

In some cases, the error in state estimation can be significantly large, and plant associated with flow F_i may require a lower sampling period. Since all nodes are not aware of the state estimation errors for all flows, a sensor that monitors the plant (with large errors) must broadcast the state estimation error to all nodes. To enable this communication, we reserve ν (where $\nu \ll \tau$) number of time slots in every τ time slots (typically the last ν slots). A sensor with a positive error can use this ν time slots to transmit the error using CSMA-CA protocol. We choose $\xi(t) > 0$ as the condition to transmit the state estimation error for its simplicity. However, the system designer is free to choose any policy. In practice, the random noise should not be high for all plants at the same time, and hence, a small value of ν should be sufficient to communicate the errors without collision. Note that CSMA-CA protocol is only used within the ν time slots to transmit the observed states of the plants with high state estimation errors. All other communications happen using the autonomous TDMA scheduler proposed in Section IV and their real-time performance is not affected by these ν time slots.

Note that, in the proposed approach, all sensors receive the state space equations, initial states, and control inputs of all plants from the controller. Since the initial state, control input calculation, and noise calculation for all plants are the same at all sensors, all nodes converge to the same the state estimation for all plants.

E. Segregating the Flows

Once the state estimation for current time slot (t) is estimated, a similar approach as mentioned in Section V-D can

be used to estimate the state at $t + \tau^{\text{th}}$ time slot using the current sampling period. Given the state of the system $x(t + \tau)$ and $x(t)$, we compute the Lyapunov function for the linear controller at time t and $t + \tau$ (given by $V(t)$ and $V(t + \tau)$, respectively) using Equation (9). We use the summation over the interval τ to accurately represent the trend of the Lyapunov function.

$$V(t) = \sum_{z=t-\tau}^t x_i^T(z) P_i^L x_i(z) \quad (9)$$

We want $V(t)$ to be non-increasing in time for stability. To ensure a decreasing Lyapunov function, we place a tighter restriction on $V(t)$, given by Equation (10).

$$V_i(t + \tau) \leq \beta V_i(t) \quad (10)$$

If the above equation is satisfied with the initial sampling period or the last used sampling period, then there is a steady decrease in the Lyapunov function, and the plant should be reaching the final state. In these situations, the sampling period can be higher than the last used sampling period or the initial sampling period. However, if the above equation is not satisfied, then it implies that the plant may experience some noise which can affect the stability of the system. In this case, the sampling period of the system should be equal to or lower than the initial sampling period.

Given the two outcomes of Inequality (10), the plants can be segregated into two categories: plants with an increase in the sampling period and plants with a decrease in the sampling period. The sampling period is first computed for all plants with an increase in the sampling period and followed by all plants with a decrease in the sampling period. The approach to increase and decrease sampling period is discussed in the following sections.

For the simplicity in explanation, we use a uniform value of β in this paper. However, the system designer can choose to use multiple values of β depending on the state of the plant. One example is to choose a value of β close to 0 when the system is in an unstable region or approaching a stable region, and close to 1 when the system is within a stable region. Note that our approach only provides a best-effort approach to ensure the stability and closed loop performance of the plants. We choose to avoid imposing restrictions on the plant model to ensure stability as the restrictions reduce the applicability of the proposed approach.

F. Increasing Sampling Period

When the state of a plant is within (or close to) a stable region, then the sampling period of that plant can be increased beyond the initial sampling period. However, a substantial increase in the sampling period may cause the state of the plant to move to an unstable region. Therefore, we formulate the increase in sampling period as an optimization problem. The objective of the problem is to find the maximum sampling rate, which satisfies the constraint specified in Equation (10).

Equation (11) formulates the sampling period maximization problem.

$$\begin{aligned} & \text{maximize} && T_i \\ & \text{subject to} && V_i(t + \tau) \leq \beta V_i(t) \end{aligned} \quad (11)$$

Our heuristic for solving the optimization problem is to use multiplicative increase and additive decrease of T_i (starting from the initial sampling period) and select the maximum T_i that satisfies the constraint. Note that the sampling period $T_i[\text{max}]$ or $T_i[\text{min}]$ is required when the plant is in an unstable region and moving towards a stable region. When the plant is within a stable region, sampling the state of the plant at $T_i[\text{max}]$ can cause energy overheads. When the plant is in a stable region, the plant can be sampled at a period higher than the $T_i[\text{max}]$ while ensuring the plant does not enter an unstable region. Thus, we use τ as an upper bound of T_i instead of $T_i[\text{max}]$.

Since all flows in the network are schedulable with the initial period assignment, any increase in the period will only lower the utilization and does not affect the schedulability of the system. Thus, we do not consider schedulability as a constraint for the optimization problem.

G. Decreasing Sampling Period

The decrease in the sampling period of a plant affects the schedulability of a system. However, it does not affect the stability of a plant. Therefore, to counteract the plant noise for the next τ time slots, each node first assigns initial period to the flow. Since the initial period is within the $T_i[\text{min}]$ and $T_i[\text{max}]$, the sampling period should always ensure stability. After the first assignment, a node computes the total utilization for all flows in the network. If the total utilization is no greater than 1, then periods can be decreased from the current period assignment. The available slack in the utilization (i.e., $1 - \sum_{i=1}^n \mu_i$) is divided equally among all the flows with a decrease in the sampling period. For example in a system with two flows with decreasing periods and the total utilization with $T_i[\text{max}]$ assignment is 0.8, the utilization of each flow ($\frac{\tau}{T_i}$) is incremented by 0.1.

VI. EVALUATION

We have evaluated the proposed sampling period acclimation (SPA) algorithm and autonomous scheduler against the holistic controller proposed in [17]. The holistic controller computes the maximum increase in the sampling period that ensures a decreasing Lyapunov function. We have compared the two approaches in terms of average energy consumed per node in the network. We define *energy consumption* as the average energy consumed per node in milli-Joules required to transmit the state and control information from sensor to actuator. We used the product of the number of transmissions and energy consumed for each communication (0.22mJ with 5.5ms Tx time) to estimate energy consumption per node. We also show the system response, i.e., the state of the system and control cost over time under SPA algorithm.

We executed the proposed SPA algorithm on Matlab and the proposed autonomous scheduler on TOSSIM [13]. Our

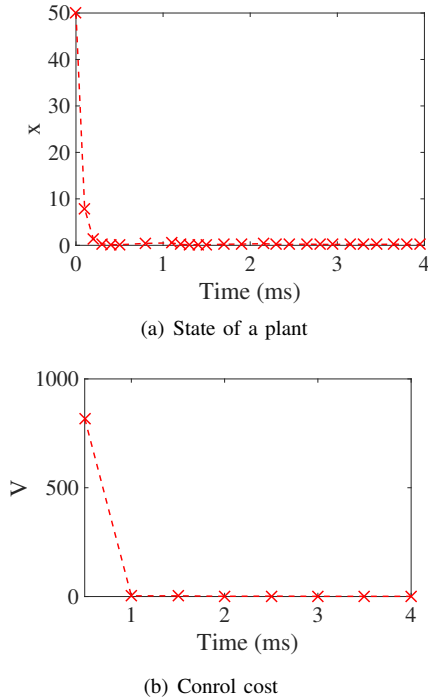


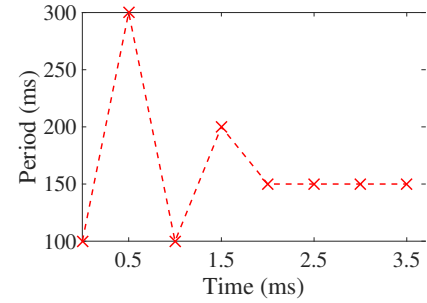
Fig. 2. System response under SPA

setup consisted of 5 plants. The state space equation of a plant is expressed as $x(t+1) = Ax(t) + Bu(t) + w$, where $A = 1.05$, $B = 1$, $x(0) = 50$, and $w = 0.05$. For simplicity, we assume all plants have the same coefficients. For a fair comparison between the proposed approach and holistic controller, we choose a linear controller with $K = -0.95$. The controller satisfies the Lyapunov stability constraint, expressed in Equation 7, where $P^L = 1$, and $Q^L = 3$. However, the proposed approach can also work with other controllers like the model predictive controller. We choose two values of β . When the state of the system is above 1, $\beta = 0.5$, and when state of the system is below 1, $\beta = 1$. The initial sampling period of all the plants was fixed at $100ms$. Since the period was selected to be $100ms$, we selected $\tau = 500ms$ since it allows for a few iterations between the initial assignment and first sampling period change.

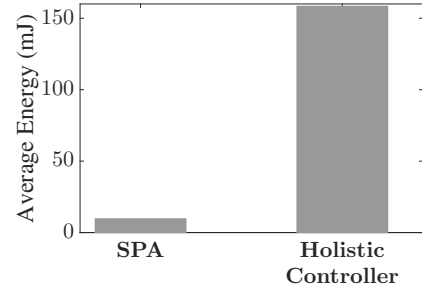
We used a star topology of 11 nodes; 5 sensors, 5 actuators, and 1 controller, with the controller in the center of the star topology. We used an autonomous TDMA medium access protocol, as proposed in the paper. Time in the network was divided into $10ms$ time slots. For a fair comparison between the proposed approach and the holistic controller, we assume the link quality to be good for all links, and each node uses 2 time slots to make a successful transmission. The MAC protocol of both the proposed approach and the holistic controller is similar to that in WirelessHART without graph routing and redundant transmissions. Since the link quality was assumed to be good, the utilization of each flow was given by $\mu_i = \frac{2}{T_i}$, and the schedulability test was given by $\sum_i \mu_i \leq 1$.

Fig. 2 shows the control cost and state of the system as time increases from $10ms$ to $4s$. We have observed that under

SPA, there was a steady decrease in the control cost as well as the state of the system. We have observed that the stabilized around 0.8 due to the presence of a fixed noise. We have observed that at $0.5s$, the state of the system was below 1, and the control cost was close to 800 . Since the control cost was so high, SPA algorithm selected a period of $300ms$ for all flows. The high period of $300ms$ resulted in a significant accumulation of noise. Due to the high noise, we observed that the plant could not be stabilized by any period greater than $100ms$. Thus, there is a sharp decrease in the period at $1s$. A similar effect was observed from $1.5s$ to $2s$, which accounts for the sharp increase and decrease in the period. After $2s$, we observed that the system was in a stable state, and control cost remained constant, and hence, the period was constant.



(a) Period selection by SPA



(b) Average energy consumption

Fig. 3. Performance of SPA

For a fair comparison with the holistic controller, we assume the holistic controller also changes the period every $\tau = 500ms$, and a similar cost metric was used. Note that, in both the approaches, the future control cost ($V(t+\tau)$) is intended to decrease by a fraction of the previous control cost. Since the holistic controller can only increase the period in multiples of 2, the closest value of the period is selected and used for the schedule. The holistic scheduler uses a flooding mechanism to transmit packets in the network, and hence the average energy consumption at the nodes is around $158.4mJ$ after a $4s$ interval. However, in the proposed approach, the average energy consumption is $9.68mJ$, which is a 92% decrease in energy consumption when compared to the holistic controller. If the holistic controller used a single hop communication without any flooding, the average energy consumption is $23.7mJ$, which shows that SPA consumes 59% less energy. These results show that SPA algorithm minimizes the energy consumption by at least 59% when compared to the holistic

controller while ensuring the control cost decreases, and the system moves closer to a stable state.

VII. CONCLUSION

In this paper, we have proposed an autonomous scheduler which maximizes the number of successful transmission in the network using game theory. In the proposed autonomous scheduler, each node avoids collisions of packets and transmissions under poor link quality. We have also derived an efficient utilization based schedulability test to determine the schedulability of all flows on the network. We have then proposed a heuristic for sampling period acclimation in wireless control systems. Our evaluation on a case study showed that the proposed approach consumes at least 59% less energy when compared to the state-of-the-art approach. Our autonomous scheduler and sampling period selection algorithm are proposed for a single hop network. In the future, we plan to extend these algorithms for multi-hop networks.

ACKNOWLEDGMENT

This work was supported by NSF through grants CNS-1565751 and CAREER-1846126, and by the Graduate School of Wayne State University.

REFERENCES

- [1] ISA100, 2007. <https://www.isa.org/isa100/>.
- [2] WirelessHART, 2007. <https://fieldcommgroup.org/technologies/hart>.
- [3] S. Akashi, H. Ishii, and A. Cetinkaya. Self-triggered control with tradeoffs in communication and computation. *Automatica*, 94, 2018.
- [4] 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 *ICCPs'12*.
- [5] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi. Long-range communications in unlicensed bands: The rising stars in the iot and smart city scenarios. *IEEE W. Comm.*, 23(5), 2016.
- [6] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert. 6tisch: deterministic ip-enabled industrial internet (of things). *IEEE Communications Magazine*, 52(12), 2014.
- [7] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: Robust mesh networks through autonomously scheduled tsch. In *SenSys'15*.
- [8] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke. Smart grid technologies: communication technologies and standards. *IEEE Trans. on Ind. Inf.*, 7(4), 2011.
- [9] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon. Reliable and real-time communication in industrial wireless mesh networks. In *RTAS'11*.
- [10] E. Henriksson, D. E. Quevedo, E. G. Peters, H. Sandberg, and K. H. Johansson. Multiple-loop self-triggered model predictive control for network scheduling and control. *Trans. on Cont. Sys. Tech.*, 23(6), 2015.
- [11] X. Jin, F. Kong, L. Kong, W. Liu, and P. Zeng. Reliability and temporality optimization for multiple coexisting wirelesshart networks in industrial environments. *IEEE Trans. on Ind. Elec.*, 2017.
- [12] M. Kishida. Event-triggered control with self-triggered sampling for discrete-time uncertain systems. *IEEE Trans. on Aut. Cont.*, 2018.
- [13] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *SenSys*, 2003.
- [14] H. Li, W. Yan, and Y. Shi. Triggering and control co-design in self-triggered model predictive control of constrained systems: With guaranteed performance. *IEEE Trans. on Aut. Cont.*, 2018.
- [15] C. Liu, H. Li, J. Gao, and D. Xu. Robust self-triggered min-max model predictive control for discrete-time nonlinear systems. *Automatica*, 2018.
- [16] T. Liu and A. E. Cerpa. Data-driven link quality prediction using link features. *ACM Transactions on Sensor Networks (TOSN)*, 10(2), 2014.
- [17] Y. Ma and C. Lu. Efficient holistic control over industrial wireless sensor-actuator networks. In *ICII'18*.
- [18] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 2000.
- [19] V. P. Modekurthy, D. Ismail, M. Rahman, and A. Saifullah. A utilization-based approach for schedulability analysis in wireless control systems. In *ICII'18*.
- [20] V. P. Modekurthy, A. Saifullah, and S. Madria. Distributed graph routing for wirelesshart networks. In *ICDCN'18*.
- [21] V. P. Modekurthy, A. Saifullah, and S. Madria. Distributedhart: A distributed real-time scheduling system for wirelesshart networks.
- [22] M. R. Palattella, P. Thubert, X. Vilajosana, T. Watteyne, Q. Wang, and T. Engel. 6tisch wireless industrial networks: Determinism meets ipv6. In *Internet of Things*. Springer, 2014.
- [23] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson. Wireless network design for control systems: A survey. *IEEE Comm. Sur. & Tut.*, 20(2), 2018.
- [24] C. Peng, D. Yue, and M. Fei. A higher energy-efficient sampling scheme for networked control systems over ieee 802.15. 4 wireless networks. *IEEE Trans. Ind. Inf.*, 12(5), 2016.
- [25] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettisalo. On the coverage of lpwans: range evaluation and channel attenuation model for lora technology. In *ITST'15*.
- [26] E. G. W. Peters, D. E. Quevedo, and M. Fu. Co-design for control and scheduling over wireless industrial control networks. In *CDC'15*.
- [27] M. Rahman, D. Ismail, V. P. Modekurthy, and A. Saifullah. Implementation of lpwan over white spaces for practical deployment. *IoTDI'19*.
- [28] I. Saha, S. Baruah, and R. Majumdar. Dynamic scheduling for networked control systems. In *HSCC'15*.
- [29] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen. Schedulability analysis under graph routing in WirelessHART networks. In *RTSS'15*.
- [30] A. Saifullah, M. Rahman, D. Ismail, C. Lu, J. Liu, and R. Chandra. Enabling reliable, asynchronous, and bidirectional communication in sensor networks over white spaces. In *SenSys'17*.
- [31] A. Saifullah, M. Rahman, D. Ismail, C. Lu, J. Liu, and R. Chandra. Low-power wide-area network over white spaces. *IEEE/ACM Transactions on Networking*, 26(4), 2018.
- [32] A. Saifullah, S. Sankar, J. Liu, C. Lu, R. Chandra, and B. Priyantha. Capnet: Exploiting wireless sensor networks for data center power capping. *ACM Trans. on Sen. Net.*, 15(1), 2018.
- [33] A. Saifullah, S. Sankar, J. Liu, C. Lu, B. Priyantha, and R. Chandra. CapNet: A real-time wireless management network for data center power capping. In *RTSS '14*, 2014.
- [34] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. Real-time scheduling for WirelessHART networks. In *RTSS'10*.
- [35] N. Samian, Z. A. Zukarnain, W. K. Seah, A. Abdullah, and Z. M. Hanapi. Cooperation stimulation mechanisms for wireless multihop networks: A survey. *Jour. of Net. and Comp. App.*, 54, 2015.
- [36] D. Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.
- [37] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Trans. on Ind. Inf.*, 2018.
- [38] F. Smarra, A. D'Innocenzo, and M. D. D. Benedetto. Optimal co-design of control, scheduling and routing in multi-hop control networks. In *CDC'12*.
- [39] K. Srinivasan, M. Jain, J. I. Choi, T. Azim, E. S. Kim, P. Levis, and B. Krishnamachari. The k-factor: Inferring protocol performance using inter-link reception correlation. In *Mobicom'10*.
- [40] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Ind. Elec. Magazine*, 11(1), 2017.
- [41] H. Xu and S. Jagannathan. Distributed joint optimal network scheduling and controller design for wireless networks. In *Computer Networks & Communications (NetCom)*, pages 147–162. Springer, 2013.
- [42] T. Zhang, T. Gong, C. Gu, H. Ji, S. Han, Q. Deng, and X. S. Hu. Distributed dynamic packet scheduling for handling disturbances in real-time wireless networks. In *RTAS*. IEEE, 2017.
- [43] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu. Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks. *IEEE Trans. on Mob. Comp.*, 2018.
- [44] X.-M. Zhang, Q.-L. Han, and B.-L. Zhang. An overview and deep investigation on sampled-data-based event-triggered control and filtering for networked systems. *IEEE Trans. on Ind. Inf.*, 13(1), 2017.
- [45] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele. Adaptive real-time communication for wireless cyber-physical systems. *ACM Trans. on CPS*, 1(2), 2017.