Co-Synthesis of Loop Execution Patterns for Multihop Control Networks

Sumana Ghosh, Soumyajit Dey, and Pallab Dasgupta

***Abstract*—We propose pattern-based execution of control loops as a preferable alternative to traditional fully periodic execution for implementing a set of embedded control systems where the sensors, actuators, and control nodes are connected via a shared wireless network. We use a theoretically sound model for skipping loop executions without compromising stability as the basis for developing an SAT-based approach for synthesizing such schedulable loop execution patterns over multihop (wireless) control networks. We demonstrate superior control performance as compared to controllers designed under the fully periodic solutions.**

***Index Terms*—Communication schedule, multihop control network (MCN), network schedulability, SAT.**

I. Introduction

THE PROBLEM of executing a set of control loops over a shared wireless network is at the heart of the rapidly expanding family of wireless-based cyber-physical systems. As shown in Fig. 1, such systems may consist of a control node $\mathcal{C}$, responsible for controlling a set of plants, a set of sensor nodes $V_{\mathtt{S}}$, a set of actuator nodes $V_{\mathtt{A}}$, and several intermediate wireless nodes given by the set $V_{\mathtt{I}}$. Other notations are introduced later. Each control loop must execute periodically, and for each execution of a control loop, the wireless network must carry the sensor data from one or more sensor nodes to the control node, and actuation messages from the control node to one or more actuator nodes.

The quality of control is closely related with the period at which a control loop executes. In a multihop control network (MCN), the choice of the sampling periods of various control loops must take into cognizance the bandwidth limitations of the wireless network. It is possible that the sampling rates of the control loops chosen by a designer from control theoretic considerations generate infeasible traffic on the network, that is, all sense/actuation messages cannot be delivered on time by the network. Traditionally control designers use fully periodic execution patterns, and therefore in such situations they are forced to settle for an inferior controller which works with a lower sampling rate, thereby compromising the quality of control.

In this letter, we propose a novel alternative to the current practice. Instead of using the inferior controller, we propose to employ the original unschedulable sampling rate by intentional skipping of control law computation and communication in the form of a *well defined pattern of loop executions*. Systematic insertion of idle slots across multiple such control loops can help in co-scheduling the loops over the limited network bandwidth, which may not have been possible otherwise (using standard periodic loop execution). We employ an SAT-based methodology to synthesize the communication schedules for the traffic generated by such aperiodic patterns to be carried in real time by the wireless network. We use a sound control theoretic model for generating constraints for the SAT solver such that the generated scheduling solutions enjoy provable stability guarantees, even under packet drops as induced by the network. We present experimental evidence to show superior quality of control as compared to the inferior controller which the traditional designer has to settle for.

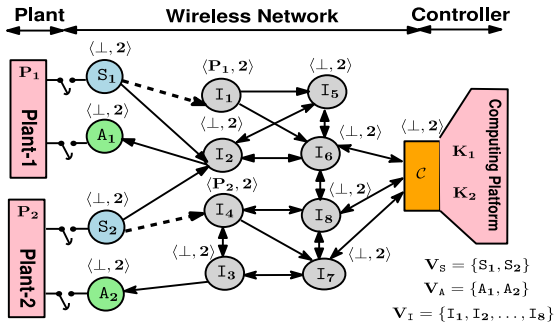

Fig. 1. Typical MCN.

This letter assumes significance in the backdrop of related work. The problems of channel scheduling [4], end-to-end delay analysis, and selection of optimal sampling rates for MCNs [7] are well studied. However, the work in [4] does not consider the co-design problem, as we do in this letter. The work in [7] assumes predefined communication schedules and fully periodic sampling rates for all the control loops, which therefore suffers from precisely the suboptimality problem that we address. The work in [6] finds the optimal packet delivery sequence, but does not consider wireless networks and the constraints that such networks induce. Research performed in [1] provides a compositional method for deriving a global switched system for an MCN and checking its stability under the assumption of a given communication schedule for each control loop. As a derivative, it is also possible to synthesize suitable communication schedules given the specification of individual control loops [3]. However, as admitted by Fiore *et al.* [4], their construction of an automaton capturing the admissible patterns suffers from scalability issues. On the other hand, inside our algorithmic framework we use the theory of [2], which provides computable lower bounds on the number of loop executions while guaranteeing stability. This

Manuscript received October 13, 2017; accepted November 14, 2017. Date of publication November 23, 2017; date of current version October 18, 2018. This manuscript was recommended for publication by H. Tomiyama. *(Corresponding author: Sumana Ghosh.)*

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India (e-mail: sumanaghosh@cse.iitkgp.ernet.in; soumya@cse.iitkgp.ernet.in; pallab@cse.iitkgp.ernet.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2017.2777506

enables us to choose the execution patterns without having to construct any automaton.

II. Formal Problem Statement

An MCN is a tuple $\mathcal{N} = \langle \mathcal{P}, \mathcal{K}, \mathcal{G} \rangle$ consisting of a set $\mathcal{P}$ of plants, a set $\mathcal{K}$ of controllers, and a network model $\mathcal{G}$. The following sections define the control model and the network model formally. At the end of this section we define the co-synthesis problem.

A. Plant-Controller Model

We use the vector $x = [x_p^\mathsf{T}, x_c^\mathsf{T}]^\mathsf{T}$ to represent the state vector of the closed loop control system, where the vectors x_p and x_c represent the state of the plant variables and the control variables respectively. The dynamics of physical plants when sampled with some fixed interval h can be expressed as discrete-time linear time-invariant (LTI) systems with equations: $x_p[k+1] = A_p x_p[k] + B_p u[k]$, $y[k] = C_p x_p[k]$, where the vectors $x_p[k]$, $y[k]$, and $u[k]$ define the plant state, the output, and the control input, respectively, at time $t = kh$, for some $k \in \mathbb{N}$ (that means, t is the real time at kth sample). The matrices A_p, B_p, and C_p describe the transition matrix, the input map, and the output map for the plant model, respectively.

As a running example, we consider a system of cart inverted pendulum [5]. Corresponding to the sampling interval $h = 70$ ms, the discrete time linearized dynamic matrices obtained for this system are as follows:

$$A_p = \begin{bmatrix} 1 & 0.07 & 01 & 0 \\ 0 & 0.96 & 0.03 & 0 \\ 0 & -0 & 1.02 & 0 \\ 0 & -0.03 & 0.55 & 1.02 \end{bmatrix}, B_p = \begin{bmatrix} 0 \\ 0.12 \\ 0 \\ 0.09 \end{bmatrix}, C_p^\mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

The feedback control software senses the plant output y and decides the control action by adjusting the control variables in u. The feedback control law is usually represented as an LTI system given as: $x_c[k+1] = A_c x_c[k] + B_c y[k]$, $u[k] = C_c x_c[k]$ where, $x_c[k]$ represents the state of the controller at time $t = kh$, and A_c, B_c, and C_c are the state transition matrix, the input map, and the output map for the controller, respectively. Thus, for an LTI plant $P = (A_p, B_p, C_p)$ with stabilizing controller $K = (A_c, B_c, C_c)$, the dynamics of the resulting closed loop $\Sigma = \langle P, K \rangle$ is given as

$$x[k+1] = \begin{bmatrix} A_p & B_p C_c \\ B_c C_p & A_c \end{bmatrix} x[k] = A_1 \; x[k]. \tag{1}$$

The notion of loop skipping, which is central to our approach, can be modeled as follows. If the controller does not execute in a sampling interval $[k, k+1]$, then the value of the control variables remain same (that is, we have $x_c[k+1] = x_c[k]$) and only the plant variables are updated. In this interval the closed loop dynamics is therefore $x[k+1] = A_0 \; x[k]$, where A_0 is same as A_1 except that A_c is replaced by the identity matrix, I, and B_c is replaced by the null matrix, O. An *execution pattern* for the controller can be equivalently represented by a sequence over $\{A_0, A_1\}$.

Formally, for a given control loop $\Sigma = \langle P, K \rangle$ together with its dynamic matrices $\{A_0, A_1\}$, we define a *loop execution pattern* as an infinite computation schedule generated by an infinitely repeating finite length string $s \in \{A_0, A_1\}^*$. Therefore $\forall k \in \mathbb{N}$, the dynamics of the closed loop system, $\langle P, K \rangle$, is defined as: $x[k+1] = s[k\%l]x[k]$, where l is the length of s. For example, according to the pattern $s = A_1A_1A_0A_0A_1A_0$, we have $x[6] = A_0A_1A_0A_0A_1A_1x[0]$. Note that, in the sampling intervals corresponding to positions where we have A_0, the network does not have to carry the sense and actuation messages for that control loop.

slots:	1	2	3	4	5	6		
CH-1	$(\mathtt{S}_1, \mathtt{I}_1)$	$(\mathtt{I}_1, \mathtt{I}_6)$	$(\mathtt{I}_6, \mathcal{C})$	$(\mathcal{C}, \mathtt{I}_6)$	$(\mathtt{I}_6, \mathtt{I}_2)$	$(\mathtt{I}_2, \mathtt{A}_1)$		
CH-2	$(\mathtt{S}_2, \mathtt{I}_4)$	$(\mathtt{I}_4, \mathtt{I}_8)$	$\perp$	$\perp$	$(\mathtt{I}_8, \mathcal{C})$	$(\mathcal{C}, \mathtt{I}_7)$	$(\mathtt{I}_7, \mathtt{I}_3)$	$(\mathtt{I}_3, \mathtt{A}_2)$

Fig. 2. Communication schedules.

B. Network Model

The network model $\mathcal{G}$ consists of a directed connectivity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the number of *channels*, M (as explained below). $\mathcal{V}$ represents the set of vertices, consisting of the set $V_\mathtt{S}$ of sensor nodes, the set $V_\mathtt{A}$ of actuator nodes, the control node $\mathcal{C}$, and the set $V_\mathtt{I}$ of intermediate nodes. An edge from nodes v_i to v_j exists if the network allows message transmission from v_i to v_j. Sensor nodes can only transmit and actuator nodes can only receive.

Our SAT-based modeling of MCN is well equipped for capturing standard industrial wireless control protocols, such as WirelessHART [8].

In WirelessHART, each channel is divided into time slots of length $\delta = 10$ ms. Each time slot on each channel allows exactly one transmission and its acknowledgment between a pair of nodes. The sensor and actuator messages generated by the recurrent loop execution patterns has to be routed across the network by using these transmission slots. A *communication schedule* defines the use of time slots on each channel by pairs of communicating nodes. In our case, the communication schedule is recurrent because the loop execution patterns generate recurrent traffic on the network.

Fig. 2 shows a part of the communication schedule with two plants. Conservatively assuming interference between any two transmissions using the same frequency, two channels are used—one for each plant. The recurrent channel occupancy schedules, ρ_1 and ρ_2 (for channels 1 and 2, respectively), are shown in the figure. In transmission slot 1, channel 1 is used by $\mathtt{S}_1$ to transmit to node $\mathtt{I}_1$ [denoted by $(\mathtt{S}_1, \mathtt{I}_1)$] while channel 2 is used by $\mathtt{S}_2$ to transmit to node $\mathtt{I}_4$. Empty transmission slots are shown as $\perp$, e.g., the sensor message from $\mathtt{S}_2$ waits at $\mathtt{I}_8$ as the node $\mathcal{C}$ is busy in communicating with $\mathtt{I}_6$ in slot 3. Since ρ_1 and ρ_2 have length 6 and 8, respectively, the global state of the network will recur after $\text{lcm}(6, 8) = 24$ slots.

Formally, for M channels, there are M recurrent channel occupancy schedules $\rho_1, \ldots, \rho_M$ of lengths $\pi_1, \ldots, \pi_M$. We define a communication schedule as a *synchronous parallel composition* of the channel occupancy schedules denoted as $\Omega = \rho_1 \| \cdots \| \rho_M$ having period $\Pi = \text{lcm}(\pi_1, \ldots, \pi_M)$.

Given the plant-controller model, $\langle \mathcal{P}, \mathcal{K} \rangle$, the network model, $\mathcal{G}$, and the specification of network induced packet drops, $\mathcal{D}$, our goal is to co-synthesize the following.

1) A recurrent execution pattern for each control loop. The pattern may not be uniform, that is, loop executions may be skipped intentionally at well defined recurring points in a uniform periodic pattern to obtain the chosen recurrent execution pattern. That is the set $\mathscr{S} = \langle s_1, \ldots, s_n \rangle$.

Algorithm 1: Generate Pattern-Based Schedule

Input: MCN $\mathcal{N} = \langle \mathcal{P}, \mathcal{K}, \mathcal{G} \rangle$, packet drop specification, $\mathcal{D}$:(d, k)
Output: $\langle \mathscr{S}, \Omega \rangle$

```
Compute ⟨r_1, ··· , r_n⟩;
for each of choice ⟨l_1, ··· , l_n⟩, l_i ∈ {2, ... , l_max} do
    ∀r_i ∈ ⟨r_1, ··· , r_n⟩, set q_i = ⌈l_i × r_i⌉ ;     // set #actuation
    q_min= [q_1, ··· , q_n]; q_max=[l_1, ··· , l_n];
    if q_max ≠ q_min then
        q=q_max ;          // Initially periodic execution
        while q ≥ q_min do
            Choose a control loop Σ_i such that q[i] > q_min[i];
            q[i]=q[i]-1 ;      // Allow one more loop skip
            Compute t_b = lcm(l_1 × h_1, ··· , l_n × h_n);
            if SolveSAT(N, q, q_max, t_b) then
                ⟨S, Ω⟩ ← output_pattern_and_schdule();
                if Check_Packet_Drop(S, (d, k)) then
                    ⟨S, Ω⟩ ← report_solution();
15 "Schedule does not exist for pattern length up to l_max";
```

2) A communication schedule, Ω, that realizes the timely routing of the messages on the wireless as required by the loop execution patterns of all the plants.

The co-synthesis problem needs to guarantee that the controllers are stable and realizable on the MCN even under network packet drops following the specification $\mathcal{D} = (d, k)$, which implies that the number of network induced packet drops is bounded by maximum of d drops in every k consecutive transmission window.

III. Co-Synthesis Methodology

The stability of periodic control loops in the presence of packet drops has been studied in [2], which provides a sufficient condition on the *rate of loop skipping* while guaranteeing stability. Following [2] according to our context we can say, given a control loop Σ with associated Schur stable closed loop dynamics A_1 and unstable open-loop dynamics A_0, the rate, r, of successful execution of the loop over an infinite horizon is bounded by $r_{\min} < r \leq 1$ where

$$r_{\min} = \frac{1}{1 - \gamma_1/\gamma_0}, \quad \gamma_0 > \gamma_1, \quad \gamma_1 < 1 \tag{2}$$

with $\gamma_j = 2\log_e \lambda_{\max}(A_j)$, $\lambda_{\max}(A_j)$ is the maximum eigenvalue of A_j, $j = 0, 1$. This means, if we choose a loop execution pattern of length l, we may allow up to $\lfloor l \times (1 - r_{\min}) \rfloor$ number of loop skipping without compromising the closed loop stability.

Algorithm 1 outlines the proposed approach. The algorithm starts with the fully periodic loop execution patterns for all the controllers. For the chosen set of loop execution patterns (assuming the maximum pattern length as $l_{\max}$), the function `SolveSAT()` use an SAT-based approach to determine whether a communication schedule exists corresponding to those loop execution patterns. If so, the witness for satisfiability yields the communication schedule. If not, then the algorithm inserts a suitable idle slot in the recurrent loop execution pattern of one controller while ensuring that the number of such loop skips never violates its minimum required execution rate, and reiterates the above steps.

We now explain the function `SolveSAT()`, where the formalism used is as follows. At a time t, a node v_j is in state, $v_j(t) = \langle P_i, \tau \rangle$, if it contains a sensor message generated by sensor of P_i at some time τ, where $\tau \leq t$. Likewise, $v_j(t) = \langle K_i, \tau \rangle$, if it contains an actuation message of the controller K_i generated by the control node $\mathcal{C}$ at some time τ. The state of the node v_j may also be of the form $v_j(t) = \langle \perp, \_ \rangle$ indicating that the node has no messages.

The state of the MCN at any time t is the collection of all node states and is given by $z(t) = \langle v_1(t), v_2(t), \ldots, v_{|\mathcal{V}|}(t) \rangle$, where $z(t)[v_j] = v_j(t)$. In the initial state $z(0)$, all nodes are empty. For example, in Fig. 1, the states of the nodes at $t = 2$ are shown besides the nodes. The state update rules for MCN states are as follows.

1) *Update Due to Sensor Message Generation:* A sensor node $\mathtt{S}_i$ generates sensory messages following the chosen loop execution pattern, s_i, for P_i. If $|s_i| = l$ then $\forall d \leq l$, if $s_i[d] = A_{1,i}$, then

$$z((ql + d)h_i)[\mathtt{S}_i] = \langle P_i, (ql + d)h_i \rangle, q \in \{0, 1, 2, \ldots\}.$$

2) *Update Due to Actuation Message Generation:* If $z(t)[\mathcal{C}] = \langle P_i, \tau \rangle$, then

$$z((d + 1)\delta)[\mathcal{C}] = \langle K_i, (d + 1)\delta \rangle, d\delta \leq t < (d + 1)\delta.$$

3) *Update Due to Message Forwarding:* For an edge $(v_i, v_j) \in \Omega(d)$, where $\Omega(d)$ represents the set of transmissions occurred at dth slot according to Ω, we have

$$z((d + 1)\delta)[v_i] = \langle \perp, \_ \rangle, z((d + 1)\delta)[v_j] = z(d\delta)[v_i].$$

Our SAT formulation also uses the following constraints.

1) *Transmission Constraint:* A node can send only to its neighbor, that is, $(v_i, v_j) \in \Omega(d) \Rightarrow v_j \in N(v_i)$.
2) *Mutual Exclusion Constraint:* A node cannot receive from multiple senders at the same time. If $(v_i, v_j) \in \Omega(d)$ then for every $v_m \neq v_i$, $(v_m, v_j) \notin \Omega(d)$.
3) *Conflict Constraint:* A node cannot act as sender and receiver at the same time step. If $(v_i, v_j) \in \Omega(d)$ then for every $v_m \neq v_i$, $(v_m, v_i) \notin \Omega(d)$.
4) *Validity Constraint:* A message is valid in the network only for its sampling interval and any node always contains such valid messages. Following Ω, if $z(t)[v_j] = \langle P_i, \tau \rangle / \langle K_i, \tau \rangle$, for any $v_j \in \mathcal{V}$ and any loop Σ_i, $1 \leq i \leq n$, then $t - \tau \leq h_i$.
5) *Existential Constraint:* A message can exist only in one node at a particular time instant. Following Ω, if $z(t)[v_j] = \langle P_i, \tau \rangle / \langle K_i, \tau \rangle$, then $\forall t, \forall v_m \in \mathcal{V}, m \neq j$, $z(t)[v_m] \neq \langle P_i, \tau \rangle / \langle K_i, \tau \rangle$.

A potential *recurrent schedule* Ω with period Π for the MCN defines a sequence of state transitions such that there exists a state $z(t)$ *reachable* from $z(0)$ using some possibly nonrecurrent transmissions followed by $z(t) \stackrel{\Omega}{\Rightarrow} z(t + \Pi\delta)$ such that in the MCN state $z(t)$ and $z(t+\Pi\delta)$, the sensor/actuation message at node v are exactly same, for any $v \in \mathcal{V}$.

For the chosen set of loop execution patterns: $\langle s_1, \ldots, s_n \rangle$, the recurring time, η, as $\eta = \text{lcm}(|s_1| \times h_1, \ldots, |s_n| \times h_n)$. Note that inside the interval $(t, t + \eta)$, the loop execution patterns for all control loops repeat an integer number of times. Given that all messages generated inside this interval need to be consumed in their causal order, a simple *producer–consumer*-based argument can be used to prove the bound: $\eta \leq \Pi\delta < \eta + \min\{h_1, \ldots, h_n\}$.

Now, we consider the effect of network induced packet drops. Note that, dropping a packet has the same effect as skipping an execution of the loop. Hence, we keep an additional verification pass [the function `Check_Packet_Drop()` in

line 13] to check whether the schedulable solutions returned by `SolveSAT()` meets the actual minimum rate $r_i, \forall i$ even under the packet drops induced following the specification (d, k). This is simply done by the following steps: 1) unroll the pattern $s_i \in \mathscr{S}$, up to the bound $L = \text{lcm}(l_i, k),\ \forall i$; 2) automatically introduce all possible packet drops over the schedulable combination of patterns, $\mathscr{S}$, using suitable SAT clauses; and 3) check for existence of more than $(l_i - \lceil r_i \times l_i \rceil)$ number of loop skips in all the sequences.

IV. Experimental Setup and Results

We demonstrate the proposed method on the system of two cart-inverted pendulums [5] sharing the MCN of Fig. 1. The control input u is the force that moves the cart horizontally and outputs are the horizontal displacement of the cart and angular displacement of the pendulum. We use settling time and peak overshoot limits of 5 s and 0.35 radians, respectively, as the performance criteria. Two systems differ by their parameters values. The values for the parameters (cart-mass, pendulum-mass, pendulum length, friction coefficient, and mass moment of inertia) are taken as (0.5, 0.1, 0.3, 0.25, and 0.03) and (0.6, 0.2, 0.3, 0.3, and 0.06), respectively.

Under these control design constraints, and in the absence of any limitation on the network bandwidth, a suitable choice of sampling periods for the two pendulums that guarantee satisfactory closed loop response are 70 and 80 ms, respectively. However, this design cannot be implemented on our MCN since the choice of sampling rates generates message traffic in a pattern for which no communication schedule exists (as determined using our SAT-based analysis).

The traditional alternative in such situations would be to reduce the periodic sampling rates. For this, we implement two existing methods of optimal rate selection through greedy heuristic and convex optimization as given in [7] under a certain choice of routing path for the loops. Both these methods report the schedulable optimal choice of sampling periods for the loops as 70 and 100 ms, respectively, for which we compute the optimal periodic controllers using standard linear quadratic regulator (LQR)-based control design technique.

Fig. 3 compares the output response of Pendulum-2 for *option a)* the superior but infeasible controller with period 80 ms, *option b)* the inferior but schedulable controller with period 100 ms [7], and *option c)* controller having period 80 ms with schedulable loop execution pattern as generated by our tool-flow. In all cases we consider that the network induced packet drops are bounded by maximum two drops in every sequence of 14 transmissions, i.e., the network packet drop specification in Algorithm 1 is (2, 14).

The degradation in peak overshoot and settling time for the feasible periodic solution (*option b*) is clearly evident when compared with the infeasible periodic solution (*option a*). In case of *option c*, Algorithm 1 synthesizes sets of schedulable loop executions patterns with sampling periods 70 and 80 ms, respectively, with occasional loop skipping instrumented inside as chosen by the SAT. For the pendulums, we choose the recurrent patterns, $A_0A_1A_1A_1A_1A_1A_1A_1$, and $A_1A_1A_1A_1A_1A_1A_0$ respectively among the set of schedulable solutions, based on least LQR cost. Algorithm 1 takes 21.34 s to generate this solution. The comparison of the response of Pendulum-2 following *option c*, with the response for the infeasible superior controller (*option a*) in Fig. 3 establishes that the schedule generated by Algorithm 1 with instrumented loop skipping performs just as well as the superior controller which was unschedulable.

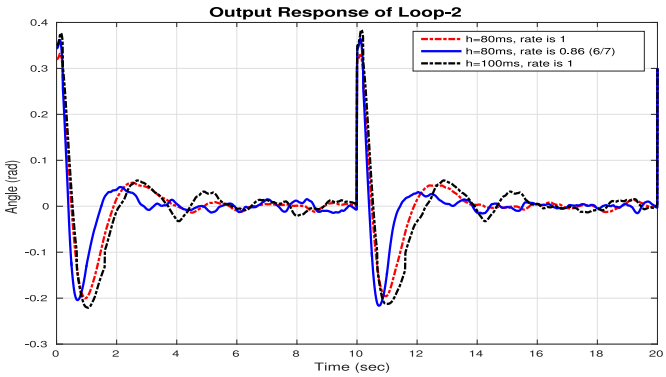

Fig. 3. Comparison of scheduling options (a), (b), and (c).

In the figure, all responses are evaluated under a disturbance scenario comprising of a Gaussian state noise with covariance $R = 0.4 \times B_p B_p^{\mathsf{T}}$ and a periodic spike of 0.3 rad with a period of 10 s. All the experiments have been performed on a 2.40 GHz Intel Xenon E5620 processor with 16 cores and 50 GB of memory, running 64-bit Ubuntu 12.04. We use the SMT solver Z3 for the SAT formulation and Z3py, a Z3 API in Python as a front-end modeling language. We have tested the scalability of our tool using other large network configurations considering: 1) 19 nodes, 55 edges, and 3 plants, and 2) 27 nodes, 68 edges, and 3 plants. In each case, the results (generated in <135 min in maximum) reported better control performance when compared with existing approaches [7]. In that way, our CAD tool-flow for automatically generating realizable MCN schedules provides a useful design aid to control engineers.

References

[1] R. Alur, A. D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss, "Modeling and analysis of multi-hop control networks," in *Proc. RTAS*, San Francisco, CA, USA, 2009, pp. 223–232.

[2] M. S. Branicky, S. M. Phillips, and W. Zhang, "Scheduling and feedback co-design for networked control systems," in *Proc. CDC*, vol. 2. Las Vegas, NV, USA, 2002, pp. 1211–1217.

[3] A. D'Innocenzo *et al.*, "Scalable scheduling algorithms for wireless networked control systems," in *Proc. CASE*, Bengaluru, India, 2009, pp. 409–414.

[4] G. Fiore, V. Ercoli, A. J. Isaksson, K. Landernäs, and M. D. Di Benedetto, "Multihop multi-channel scheduling for wireless control in wirelessHART networks," in *Proc. ETFA*, Sep. 2009, pp. 1–8.

[5] W. C. Messner and D. M. Tilbury. *Control Tutorials for MATLAB and Simulink: A Web-Based Approach*. Accessed: Aug. 10, 2017. [Online]. Available: http://ctms.engin.umich.edu/CTMS

[6] J. Ning, S. Yeqiong, and S.-L. Francoise, "Graceful degradation of the quality of control through data drop policy," in *Proc. ECC*, 2007, pp. 4324–4331.

[7] A. Saifullah *et al.*, "Near optimal rate selection for wireless control systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4, pp. 1–25, 2014.

[8] J. Song *et al.*, "WirelessHART: Applying wireless technology in real-time industrial process control," in *Proc. RTAS*, St. Louis, MO, USA, 2008, pp. 377–386.