University of Pennsylvania
ScholarlyCommons

Database Research Group (CIS) | Department of Computer & Information Science

4-13-2009

Modeling and Analysis of Multi-hop Control Networks

Alur Rajeev
University of Pennsylvania, alur@cis.upenn.edu

Alessandro D'Innocenzo
University of Pennsylvania

Karl H. Johansson
University of Pennsylvania

George James Pappas
University of Pennsylvania, pappasg@seas.upenn.edu

Gera Weiss
University of Pennsylvania, gera@seas.upenn.edu

Copyright 2009 IEEE. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.
Reprinted from:
Alur, R.; D'Innocenzo, A.; Johansson, K.H.; Pappas, G.J.; Weiss, G., "Modeling and Analysis of Multi-hop Control Networks," Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE , vol., no., pp.223-232, 13-16 April 2009
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4840583&isnumber=4840552

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/db_research/44
For more information, please contact repository@pobox.upenn.edu.

Modeling and Analysis of Multi-hop Control Networks

Abstract
We propose a mathematical framework, inspired by the Wireless HART specification, for modeling and analyzing multi-hop communication networks. The framework is designed for systems consisting of multiple control loops closed over a multi-hop communication network. We separate control, topology, routing, and scheduling and propose formal syntax and semantics for the dynamics of the composed system. The main technical contribution of the paper is an explicit translation of multi-hop control networks to switched systems. We describe a Mathematica notebook that automates the translation of multihop control networks to switched systems, and use this tool to show how techniques for analysis of switched systems can be used to address control and networking co-design challenges.

Keywords
radiocommunication, telecommunication network routing, telecommunication network topology, control co-design, formal syntax, mathematica notebook, multihop communication networks, multihop control networks, networking co-design, switched systems, wireless HART specification

Comments
Copyright 2009 IEEE. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

Reprinted from:
Alur, R.; D'Innocenzo, A.; Johansson, K.H.; Pappas, G.J.; Weiss, G., "Modeling and Analysis of Multi-hop Control Networks," Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE , vol., no., pp.223-232, 13-16 April 2009

URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4840583&isnumber=4840552

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/db_research/44

Modeling and Analysis of Multi-Hop Control Networks

Rajeev Alur[1], Alessandro D'Innocenzo[1,2], Karl H. Johansson[3], George J. Pappas[1], Gera Weiss[1]
[1]University of Pennsylvania, Philadelphia PA
[2]University of L'Aquila, Italy
[3]Royal Institute of Technology, Stockholm, Sweden

***Abstract*—We propose a mathematical framework, inspired by the WirelessHART specification, for modeling and analysing multi-hop communication networks. The framework is designed for systems consisting of multiple control loops closed over a multi-hop communication network. We separate control, topology, routing, and scheduling and propose formal syntax and semantics for the dynamics of the composed system. The main technical contribution of the paper is an explicit translation of multi-hop control networks to switched systems. We describe a Mathematica notebook that automates the translation of multi-hop control networks to switched systems, and use this tool to show how techniques for analysis of switched systems can be used to address control and networking co-design challenges.**

I. Introduction

Wireless communication is emerging in control applications with main advantages being reduced installation costs and increased flexibility, as well as ease of maintenance, debugging and diagnostics. Control with wireless technologies typically involves multiple communication hops for conveying information from sensors to the controller and from the controller to actuators. While offering many advantages, the use of multi-hop networks for control is a challenge when it comes to predictability. Motivated by this challenge, we propose a formal modeling and analysis approach for multi-hop control networks.

The challenges in designing and analyzing multi-hop control networks are best explained by considering the recently developed WirelessHART standard as an example (see Section V). The standard allows designers of wireless control networks to distribute a synchronous communication schedule to all nodes in a wireless control network. More specifically, time is divided into slots of fixed length (10ms) and a schedule is an assignment of nodes to send data in each slot. The standard specifies a syntax for defining schedules and a mechanism to apply them. However, the issue of designing schedules remains a challenge for the engineers, and is currently done using heuristics rules. To allow systematic methods for computing and validating schedules, a clear formulation of the affect of schedules on control performance is needed.

In this paper, we propose a formal model for analyzing the joint dynamics induced by scheduling, routing and control. Specifically, given a description of these separate aspects of the system, we define a switched system that models the dynamics of the composed multi-hop control network. The usefulness of the model is demonstrated by confirming that it is compatible with the WirelessHART specification and by showing that it can be used to co-design control and scheduling. For example, using an experimental tool presented in this paper, we show that it is possible to resolve design parameters of a controller by representing the dynamics of a multi-hop control network symbolically (Section VI). As another example, we show that our model allows compositional analysis based on the method developed in [1], [2] (Section VI).

The paper is structured as follows. In section II we present the structure of multi-hop control networks. Section III describes a formal syntax for specifying such systems and Section IV gives formal semantics to that syntax. Then, in Section V we discuss the relevance of the proposed modeling approach to WirelessHART and in Section VI we present an experimental tool that employs formal models of multi-hop control networks to controller and scheduler design and to verification. Section VII contains concluding remarks and directions for future research.

Related Work

When discussing the interaction of network and control parameters, most research focuses on scheduling message and sampling time assignment for sensors/actuators and controllers interconnected by wired common-bus network [3]–[6], while what is needed for modeling and analysing protocols such as WirelessHART is an integrated framework for analysing/co-designing control, routing, topology, and scheduling.

To our knowledge, the only formal model of wireless sensor/actuator network is reported in [7]. In this paper, a simulation environment that facilitates simulation of computer nodes and communication networks interacting with the continuous-time dynamics of the real world is presented. The main difference between the work presented in [7] and the one presented here is that here we propose a formal mathematical model that allows more than just simulation. For example, we show that our approach allows systematic mathematical design techniques such as sensitivity and compositional analysis.

This work is also related to the growing research body on switched system (see e.g. [8], [9]). As we show in this paper, a multi-hop control network can be abstracted as a switched system. While generic approaches that ignore the specific structure of the switched system are applicable, we provide a detailed model that identifies the contribution of specific constituents to the dynamics. For example, the elaborated

1080-1812/09 $25.00 © 2009 IEEE
DOI 10.1109/RTAS.2009.40

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

model allows us to apply the approach proposed in [1], [2] for analyzing each control loop separately in a compositional manner.

II. Multi-Hop Control Networks

A Multi-hop control network, consists of a set of plants, a controller, and nodes that communicate sensing and actuation data from plants to controllers and back. The control scheme

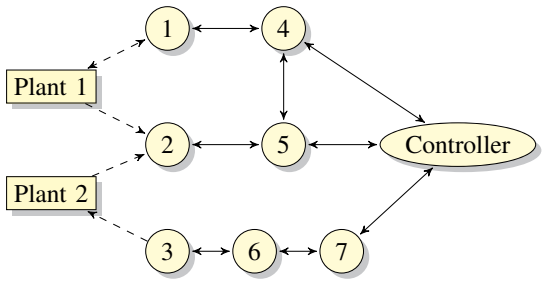


Fig. 1. An example of a multi-hop control network. Circles represent nodes with wireless communication capabilities, solid lines represent radio connectivity and dashed lines represent actuation/sensing.

is illustrated in Figure 1, where seven wireless nodes are used to measure information from two plants, send the information to a controller and then pass it back and actuate the plants. We assume that each node has radio and memory capabilities, in order to receive, store and transmit data. Each plant is considered as a dynamical system with a finite number of scalar output signals (observable outputs) and scalar input signals (control signals). Nodes in the network interact with the plants through these signals, namely they measure the observable outputs and provide actuation for the control inputs (dotted arrows). For example, in Figure 1, node 1 has both sensing and actuation capabilities for Plant 1 (bidirectional dotted arrow); node 2 has only sensing capabilities for both Plant 1 and Plant 2 (unidirectional dotted arrows to node 2 from both Plant 1 and Plant 2); and node 3 has only actuation capabilities for Plant 2 (unidirectional dotted arrow from node 3 to Plant 2). In order to close the control loop, measured data is sent from sensors to the controller through the wireless network. The computation of the control signal is performed in the controller, and control commands are sent from the controller back to the actuators. The solid arrows that connect the nodes model radio connectivity, i.e., a solid arrow is drawn from node v_1 to node v_2 if and only if node v_2 can receive signals transmitted by node v_1.

As detailed in Section III, we propose to describe multi-hop control networks by: (1) A mathematical model of the control loops, each consisting of a plant and a dynamic (stateful) feedback algorithm for controlling it. (2) The topology of the network, the location of the sensors/actuators, and the routing strategy. Note that the feedback algorithms for all the plants are typically executed by a single computer (controller), but we choose to model them with the plant. In this text, when we focus on one control loop, we will identify the controller with the control algorithm and say that a control loop consists of a plant and a controller, as is done in many control texts.

In Section IV, we formalize the semantics of multi-hop control networks by defining a switched system. The semantics of the model reflect data flow, as follows. A state of the system is a snapshot of data stored in the nodes. Transitions consist of copying data from nodes to nodes and of transformations of the control algorithms and plants states. Because transitions are governed by schedules, we propose to model the system as a discrete-time switched system where the switching signal is the communication and computation schedules. This model allows analysis of multi-hop control networks using the growing arsenal of techniques from the switched systems theory (see e.g. [9]).

III. Syntax

We propose the following formal syntax for describing a multi-hop control network. See the subsections that follow the definition for a more detailed description of each part.

***Definition** 1:* A multi-hop control network is a tuple $\mathcal{N} = \langle \mathcal{D}, \mathcal{G}, \Omega, \mathcal{R} \rangle$, where:

- $\mathcal{D} = \{\langle\langle A_i, B_i, C_i\rangle, \langle \tilde{A}_i, \tilde{B}_i, \tilde{C}_i\rangle\rangle\}_{i=1}^{p}$ models the control loops. Each control loop in D is modeled by a pair of triplets of matrices. The first triplet in each pair defines the dynamics of the plant and the second triplet defines the dynamic of the control algorithm, both in terms of matrices of Linear Time Invariant (LTI) systems. The number of columns in B_i must be the same as the number of rows in $\tilde{C}_i$, which is the number of inputs to the plant. Similarly, the number of rows in C_i must be the same as the number of columns in $\tilde{B}_i$, which is the number of measurable outputs from the plant. Let $\mathbb{I} = \cup_{i=1}^{p}\{y_{i,1}\ldots, y_{i,m_i}\}$ be the set of input signals for the plants, where m_i is the number of columns in B_i (rows in $\tilde{C}_i$). Let $\mathbb{O} = \cup_{i=1}^{p}\{u_{i,1}\ldots, u_{i,l_i}\}$ be the set of output signals from the plants, where l_i is the number of rows in C_i (columns in $\tilde{B}_i$).
- $\mathcal{G} = \langle V, E\rangle$ is a directed graph that models the radio connectivity of the network, where vertices are nodes of the network, and an edge from v_1 to v_2 means that v_2 can receive messages transmitted by v_1. We denote with v_c the special node of V that corresponds to the controller. Let $\mathcal{P}$ be the set of simple paths in $\mathcal{G}$ that start or end with the controller.;
- $\Omega\colon \mathbb{I}\cup\mathbb{O} \to V$ assigns to every input and output signal the node that implements, respectively, sensing or actuation;
- $\mathcal{R}\colon \mathbb{I}\cup\mathbb{O} \to 2^{\mathcal{P}}$ is a map, which associates to each input (resp. output) signal a set of allowed simple paths from (resp. to) the controller. We require that all elements in $\mathcal{R}(y)$ (resp. $\mathcal{R}(u)$) start (resp. end) with $\Omega(y)$ (resp.$\Omega(u)$) and end (resp. start) with the controller, for all $y \in \mathbb{I}$ (resp. $u \in \mathbb{O}$).

A. Control Loops

The variable $\mathcal{D}$, in the above definition, models the dynamics of the controlled plant and of the feedback algorithm

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

associated with it using the matrices $A_i, B_i, C_i, \tilde{A}_i, \tilde{B}_i$ and $\tilde{C}_i$. The meaning of this model is illustrated in Figure 2. Namely, each triplet $\langle A_i, B_i, C_i \rangle$ models an LTI plant and each triplet $\langle \tilde{A}_i, \tilde{B}_i, \tilde{C}_i \rangle$ models an LTI feedback block, interconnected with the plant in the usual way.

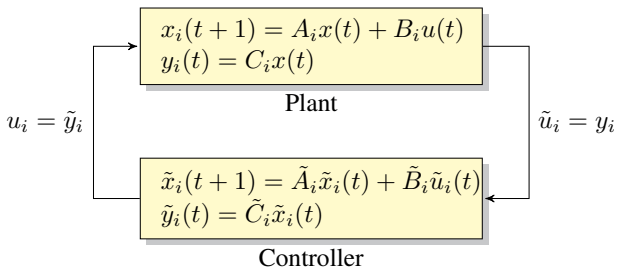


Fig. 2. A model of one control loop.

Note that the figure depicts a direct interconnection of the plant with the controller while, in reality, the wireless network introduces both measurement and actuation delays. We will model these delays later, based on the topology of the wireless network and the communication and computation schedules.

B. Radio connectivity graph

The graph $\mathcal{G}$, in the definition of a multi-hop control network, models the ability of nodes in the wireless network to receive signals sent by others. Formally, the vertices of the graph are the nodes in the network and a directed edge from node v_1 to node v_2 exists if and only if v_2 can receive signals sent by v_1. For example, the radio connectivity graph for the multi-hop control networks in Figure 1 is depicted in Figure 3.

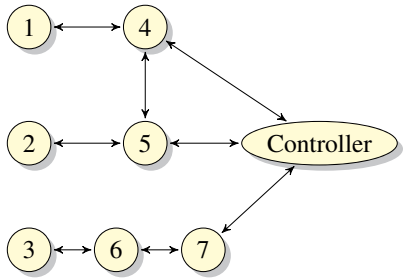


Fig. 3. Radio connectivity graph. Vertices are nodes in the wireless network. A directed edge from v_1 to v_2 says that v_2 can receive signals from v_1.

Plants are not present in the radio connectivity graph because they are not active nodes in the wireless network.

C. Sensors and actuators

The function $\Omega \colon \mathbb{I} \cup \mathbb{O} \to V$ formally defines which nodes of the network are sensors and/or actuators. Moreover, it associates sensors and actuators to the components of the output and input signals of the plant. As an example, in the system illustrated in Figure 4 the function Ω is depicted with dotted arrows, and is formally defined by $\Omega(u_{11}) = 1, \Omega(u_{21}) = 3, \Omega(y_{11}) = 1, \Omega(y_{12}) = \Omega(y_{21}) = 2$. where $\mathbb{I} = \{u_{11}, u_{21}\}$ and $\mathbb{O} = \{y_{11}, y_{12}, y_{21}\}$.

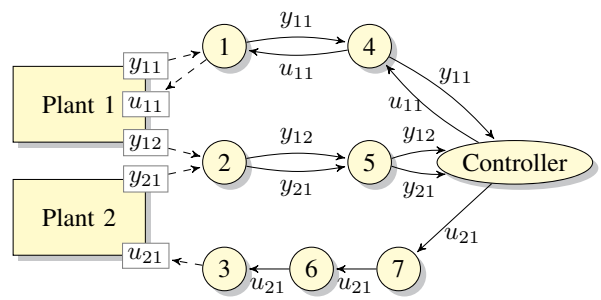


Fig. 4. A static routing, expressed as a set of paths from sensors to the controller and from the controller to actuators.

The signal $u_{i,j}$ (resp. $y_{i,j}$) denotes the jth output (resp. input) of the ith plant. With this naming convention, Ω maps rows (resp. columns) of the B matrices (resp. C matrices) to nodes of the wireless network. Specifically, if $\Omega(y_{i,j}) = k$ and $c_{i,j}$ is the jth row of C_i then the data at node k is $c_{i,j} x_i$, where x_i is the state of the ith plant. Similarly, if $\Omega(u_{i,j}) = k$ and $b_{i,j}$ is the jth column of B_i then the scalar at node k is multiplied by $b_{i,j}$ and added to x_i (every time step). These equations formalize the dynamics of the sensors and the actuators.

D. Routing

A (static) routing in a multi-hop control network is a set of acyclic paths from sensors to the controller and from the controller to actuators. For example, in Figure 4, each sensor is connected to the controller by one path and the controller is connected to each actuator by a path.

We propose two possible use cases with routing. The first use case is when the designer of the network specifies static routing as a set of allowed paths for each pair sensor-controller and controller-actuator. In this case, data can only flow along the specified paths. The second use case is when no explicit routing is specified, namely the user does not define $\mathcal{R}$. In this case, we assume a default routing $\mathcal{R}$ by considering the set of all acyclic paths from each sensor to the controller, and from the controller to each actuator.

IV. Semantics

In an ideal control loop, the input and output signals of plants and controllers are directly interconnected, namely $u(t) = \tilde{y}(t), y(t) = \tilde{u}(t)$, as depicted in Figure 2 above. However, when a multi-hop network is used to transport measured data from sensors to the controller, and actuation data from the controller to actuators, the semantics of the closed loop system need to incorporate the delays induced by the network. In particular, we need to define (i) how the measured and control data flow through the network (communication schedule), and (ii) how the controller computes the control commands (computation schedule).

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

A. Memory Slots

As the dynamics of multi-hop control networks are based on modeling information flow from sensors to the controller and from the controller to actuators, the first step towards formal semantics is an identification of the memory slots (registers) that hold that information. Specifically, each node of the network has a memory slot for each input or output signal designated for keeping the information passed to the node regarding the signal. Formally, the vertices of the memory slots graph are pairs $\langle v, \sigma \rangle$ where v is a node and σ is a signal. The edges of the memory slots graph reflect information flow channels. Specifically, there is an edge from $\langle v_1, \sigma \rangle$ to $\langle v_2, \sigma \rangle$ iff $v_1 = v_2$ or if v_1 and v_2 are consecutive nodes on some routing path of the signal σ. Namely, an edge in the graph shows where the information in each memory slot can flow – it can stay in the same memory slot or be moved to a consecutive one.

***Definition** 2:* Given a multi-hop control network with network topology $\mathcal{G} = \langle V, E \rangle$ and routing $\mathcal{R}\colon \mathbb{I} \cup \mathbb{O} \to 2^{\mathcal{P}}$, we define the graph $\mathcal{G}_{\mathcal{R}} = \langle V_{\mathcal{R}}, E_{\text{self}} \cup E_{\text{route}} \rangle$ where $V_{\mathcal{R}} = V \times (\mathbb{I} \cup \mathbb{O})$, $E_{\text{self}} = \{\langle \langle v, \sigma \rangle, \langle v, \sigma \rangle \rangle \colon v \in V, \sigma \in \mathbb{I} \cup \mathbb{O}\}$, and $E_{\text{route}} = \{\langle \langle v_1, \sigma \rangle, \langle v_2, \sigma \rangle \rangle \colon \sigma \in \mathbb{I} \cup \mathbb{O}, v_1$ and v_2 are consecutive on some $r \in \mathcal{R}(\sigma)\}$.

To avoid handling unneeded memory slots, we consider (without loss of generality) only the sub graph of $\mathcal{G}_{\mathcal{R}}$ reachable from/to the controller.

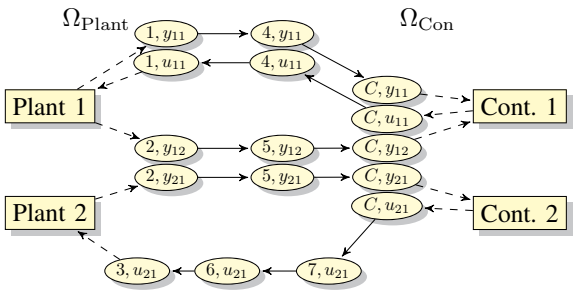


Fig. 5. The graph $\mathcal{G}_{\mathcal{R}}$ obtained by splitting each node to memory slots according to the routing scheme. The self loops are omitted for clearness of the picture.

The function Ω, defined in Section III-D above, which maps each input/output signal to a node, can be automatically extended to the function Ω_{Plant} which maps signals to memory slots (because each memory slot is mapped to a path which maps to a unique signal). Similarly, we will also use the function Ω_{Con} that maps signals of the controller to memory slots. These functions are depicted in Figure 5.

B. Controllers as Switched Systems

In Section III-A we defined controllers as linear time invariant dynamical systems. Semantically, however, we think of them as linear switched systems. The main reason for this generalization is to allow controllers to collect data before the actual control computation is executed. In particular, according to the dependence between each control signal and the measured data, we want to allow that any element of the control vector can be computed separately, when the relevant subset of measurements is ready. This requires coordinating (co-scheduling) computations and communication. Another motivation for modeling the controllers as switched systems is to allow analysis of systems with limited computational resources, where controllers need to operate in "light" mode some of the time, e.g. because other control loops need CPU resources. By defining the dynamics of the controller as switched system, we also allow modelling control techniques such as Kalman filters and Luenberger observers. To accommodate for such generalizations, all the analysis methods that we propose in this paper are independent of the structure of the controller (number of modes, dimensions, etc.).

Similar to what we proposed for routing, we propose two use-cases for handling conversion of controllers to switched systems. The first use-case is when an explicit model of the controller as a switched system is provided, and the second use-case is when only a linear time-invariant model of the controller is specified. The first case requires more modeling efforts, but it allows more general analysis of communication/computation co-scheduling.

For the second case, used in all of the example in this paper, we propose an implicit transformation of the controller from a time invariant system to a switched system as follows. Let $\langle \tilde{A}, \tilde{B}, \tilde{C} \rangle$ be a formulation of a feedback algorithm as a linear time invariant system. We define a switched system with the two modes $M = \{\text{Idle}, \text{Active}\}$. The Idle mode is defined by the matrices $\tilde{A}(\text{Idle}) := \mathbf{1}$ (identity matrix), $\tilde{B}(\text{Idle}) := \mathbf{0}$ (zero matrix) and $\tilde{C}(\text{Idle}) := \tilde{C}$. The Active mode is defined by $\tilde{A}(\text{Active}) = \tilde{A}$, $\tilde{B}(\text{Active}) := \tilde{B}$ and $\tilde{C}(\text{Active}) := \tilde{C}$. This definition models that the computation of the state variables of the controller does not have to be applied at every step, and that the state variables remain constant while the computation is not scheduled.

Mode switches of the controller are coordinated by the computation schedule described in the following section.

C. Scheduling

We propose a formal syntax for describing communication and computation schedule for a multi-hop control network:

***Definition** 3:* Given a multi-hop control network $\mathcal{N}$, let $\mathcal{G}_{\mathcal{R}} = \langle V_{\mathcal{R}}, E_{\mathcal{R}} \rangle$ be the memory slots graph as defined above.

- A communication schedule is a function $\eta\colon \mathbb{N} \to 2^{E_{\mathcal{R}}}$, that associates to each time t a set of edges of the memory slot graph. The intended meaning of this schedule is that $\langle v_1, v_2 \rangle \in \eta(t)$ iff at time t the content of the memory slot v_1 is copied to the memory slot v_2 (i.e. the physical node that maintains v_1 sends data to the physical node that maintains v_2). We require that if $\langle v_1, v_2 \rangle \in \eta(t)$ then for every $v_3 \neq v_1$, $\langle v_3, v_2 \rangle \notin \eta(t)$. Namely we do not allow assignment of two values to the same memory slot.
- A computation schedule for the ith control loop (corresponding to the ith entry in $\mathcal{D}$ of Definition 1) is a function $\mu_i\colon \mathbb{N} \to M_i$ where M_i is the set of modes of the switched-system that model the controller of the ith

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

control loop, as described in Section IV-B. The meaning of this function is that $\mu_i(t)$ defines the mode of the controller at time t.

In Section VI, below, we present a compositional analysis based on representing sets of communication schedules as regular languages over the alphabet $2^{E_{\mathfrak{R}}}$. In this context, one can also represent the set of feasible schedules in the same form. For example, if the transmission of data from node v_1 to node v_2 uses a mutually exclusive resource (e.g. radio frequency) shared with the transmission of data from node v_3 to node v_4 then the set of feasible schedules should be a sub-language of $\{S \subset E_{\mathfrak{R}} \colon \langle v_1, v_2\rangle \notin S \vee \langle v_3, v_4\rangle \notin S\}^*$ (where * is the Kleene star).

D. Multi-Hop Control Networks as Switched Systems

Based on the syntactical definition of a multi-hop control network and the schedules, we now define dynamics as switched systems. To allow compositional analysis, we model each control loop separately (plant, controller, and the data flow between them). Let i be the identifier of that control loop (corresponding to its index in the array $\mathcal{D}$ in Definition 1). We use the descriptions of the plant and the control algorithm as LTI systems, modeled by the matrices $\langle A_i, B_i, C_i\rangle$ and $\langle \tilde{A}_i, \tilde{B}_i, \tilde{C}_i\rangle$ respectively. Recall that, in Section IV-B, we transformed the control algorithm to a switched linear system with the parametrized matrices $\langle \tilde{A}_i(\cdot), \tilde{B}_i(\cdot), \tilde{C}_i(\cdot)\rangle$. The state of the switched system that models the control loop is a vector $x = \langle x_p, x_{v_1}, \ldots, x_{v_n}, x_c\rangle$ where x_p is the state of the plant, $\langle x_{v_1}, \ldots, x_{v_n}\rangle$ is a vector representing the values of the memory slots (in some fixed order), and x_c is the state of the controller. The evolutions of the different parts of the state are as follows:

- Using the matrices A_i, B_i from the definition of the plant as LTI system, we write $x_p(t+1) = A_i x_p(t) + B_i u(t)$ and $u(t) = \langle x_{\Omega_{\mathrm{Plant}}(u_1)}(t), \ldots, x_{\Omega_{\mathrm{Plant}}(u_m)}(t)\rangle$ where $u_1, \ldots, u_m \in \mathbb{I}$ are the input signals of the plant and Ω_{Plant} is the function that maps signals of the plant to sensor/actuator memory slots, i.e. the inputs to the plant are the values stored in the actuators memory slots. Similarly, $\langle x_{\Omega_{\mathrm{Plant}}(y_1)}(t), \ldots, x_{\Omega_{\mathrm{Plant}}(y_l)}(t)\rangle = C_i x_p(t)$ where $y_1, \ldots, y_l$ are the output signals of the plant, i.e., the outputs from the plant are stored in the memory slots of the sensors.
- The rest of the memory slots are updated according to the communication schedule. Specifically, $x_v(n+1) = \sum_{\langle v', v\rangle \in \mu(t)} x_{v'}(t)$ i.e., the data in each memory slot is updated to be the sum of the values of all the sources of the incoming edges to the node. In this paper, we insist that each node has at most one incoming edge which means that each destination of an edge is assigned with the value stored in the source of the edge.
- For the controller, we write $x_c(t+1) = \tilde{A}_i(\eta(t))x_c(t) + \tilde{B}_i(\eta(t))\tilde{y}(t)$ and $\tilde{y}(t) = \langle x_{\Omega_{\mathrm{Con}}(y_1)}(t), \ldots, x_{\Omega_{\mathrm{Con}}(y_l)}(t)\rangle$ where $y_1, \ldots, y_l \in \mathbb{O}$ are the output signals and Ω_{Con} is the function that maps signals of the controller to memory slots. Similarly, $\langle x_{\Omega_{\mathrm{Con}}(u_1)}(t), \ldots, x_{\Omega_{\mathrm{Con}}(u_m)}(t)\rangle = \tilde{C}_i(\eta(t))x_c(t)$ where $u_1, \ldots, u_l \in \mathbb{I}$ are the input signals.

The following two definitions formalize these dynamics as a linear switched system. The dynamics of the memory slots are modeled using the adjacency matrix of the graph induced by the communication schedule. The state of the system is $x = \langle x_p, x_{v_1}, \ldots, x_{v_n}, x_c\rangle$.

Definition 4: Given a multi-hop control network $\mathcal{N}$, consider a plant $\langle \tilde{A}_i, B_i, C_i\rangle$, and the corresponding switched linear controller $\langle \tilde{A}_i(\cdot), \tilde{B}_i(\cdot), \tilde{C}_i(\cdot)\rangle$. For any subset $e \subseteq E_{\mathfrak{R}}$ representing a sub-graph of the memory slots graph, and for any controller mode $m \in M$, we define

$$\hat{A}(e,m) := \begin{pmatrix} A_i & B_i \cdot O_{\mathrm{Plant}} & 0 \\ I^T_{\mathrm{Plant}} \cdot C_i & \mathrm{Adj}(\langle V_{\mathfrak{R}}, e\rangle)^{\mathrm{T}} & O^T_{\mathrm{Con}} \cdot \tilde{C}_i(m) \\ 0 & \tilde{B}_i(m) \cdot I_{\mathrm{Con}} & \tilde{A}_i(m) \end{pmatrix}$$

where $V_{\mathfrak{R}} = \{v_1, \ldots, v_{|V_{\mathfrak{R}}|}\}$, $\mathbb{I} = \{i_1, \ldots, i_{|\mathbb{I}|}\}$ and $\mathbb{O} = \{o_1, \ldots, o_{|\mathbb{O}|}\}$ are respectively enumerations of memory slots, inputs and outputs, $\mathrm{Adj}(\langle V_{\mathfrak{R}}, e\rangle)^{\mathrm{T}}$ is the transposed adjacency matrix of the sub-graph induced by e on $\langle V_{\mathfrak{R}}, E_{\mathfrak{R}}\rangle$, and I_x (resp. O_x) is a $\{0,1\}$ matrix of matching size with the entry $I_x(r,c)$ (resp. $O_x(r,c)$) being one if and only if $\Omega_x(v_r) = i_c$ (resp. $\Omega_x(v_r) = o_c$), for $x \in \{\mathrm{Plant}, \mathrm{Con}\}$.

Definition 5: The dynamics of the control loop are modeled by the switched system

$$x(t+1) = \hat{A}(s(t))x(t),$$

where the communication and computation schedule $s(t) = \langle \eta(t), \mu(t)\rangle$ is the switching signal.

Note the structure of the matrix $\hat{A}(\cdot,\cdot)$ that explicitly expresses the interplay between the components of a multi-hop control network. Specifically, the dynamics of the plant are at the top left, the dynamics of the controller are at the bottom right and the adjacency matrix of the sub-graph of the memory slots graph induced by the communication schedule is at the center. This model allows to use techniques from the theory of switched systems to analyze multi-hop control networks.

By combining the models of the individual loops, one can obtain a model of the whole multi-hop control network. For example, in Section VI we show how the methods presented in [1], [2], [10] are applied in the context of multi-hop control networks. Specifically, the theory of formal languages is applied for answering competing resource requirements of the loops to achieve stability of the whole system. The ability to analyze systems in a compositional manner is enabled by modeling each loop separately.

V. WirelessHART as a Multi-Hop Control Network

In this section, we show that a multi-hop control network implemented according to the WirelessHART specification can be modeled using the mathematical framework described

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

above. Our framework allows modelling the MAC layer (communication scheduling) and the Network layer (routing) of WirelessHART.

MAC layer. WirelessHART access to the channel is time slotted [11], where each slot is 10ms. A series of time slots for a given frequency channel forms a superframe (Figure 6). Slots of a superframe can be either *dedicated* to one

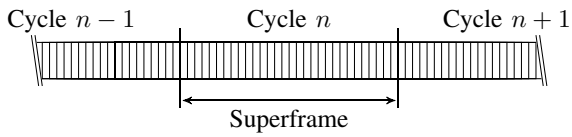


Fig. 6. Superframe structure

node or *shared* by several nodes: dedicated slots use TDMA for medium access, while shared slots use CSMA/CA for medium access. WirelessHART also enables channel hopping to avoid interference and to reduce multi-path fading. Latency requirements are addressed by scheduling the communication in such a way that packets will reach their destination in time, considering multiple hops, possible retransmissions, and alternate routes through the network.

Network layer. Routing can be implemented in two configurations: *graph routing* and *source routing*. Graph routing provides, for each pair of nodes (one source and one destination) a set of paths connecting the two nodes as an acyclic directed graph associated to the destination node. In a properly configured network, and when permitted by the radio connectivity graph, all nodes must have at least two nodes in the graph through which they may send packets (ensuring redundancy and enhancing reliability). A typical routing graph for graph routing is illustrated in Figure 7. In source routing, only one path is associated to each pair of nodes. If an intermediate link fails, the packet is lost. For this reason, source routing is much less reliable then graph routing, and the WirelessHART specification does not recommend to use it for control purposes. The specification contains other guidelines for the use of wireless networks in critical control systems. Another example: The maximum distance from a node to the gateway in such application should not exceed 4 hops. These guidelines are meant to guarantee that networking delays do not harm control performance (e.g. stability). An alternative approach can be to formally verify the composed system using the model proposed in this paper.

A WirelessHART system as a multi-hop control network. We now define the constraints induced by the WirelessHART specification on a multi-hop control network as described in Definition 1. We will consider in our framework superframes that have only dedicated slots, and not shared slots. We will show that our framework can take into account dedicated slots with slight modifications. Given a multi-hop control network $\mathcal{N} = \langle \mathcal{D}, \mathcal{G}, \Omega, \mathcal{R} \rangle$ as in Definition 1, and a schedule $s = \langle \eta, \mu_1, \ldots, \mu_p \rangle$ as in Definition 3, an implementation of such networked system according to the WirelessHART

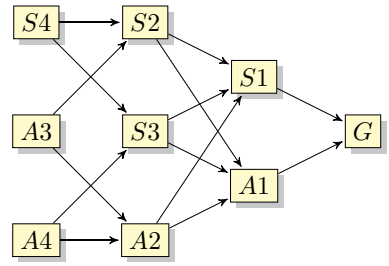


Fig. 7. Graph routing for the destination node G (gateway)

specifications must satisfy the following constraints:

Routing Constraints. Routing $\mathcal{R}$ must be defined so that any node, when needs to route data, has at least two choices for routing in the set of neighbors (if the radio connectivity graph allows this). Formally, given any input or output signal $l \in \mathbb{I} \times \mathbb{O}$, any routing $r = v_1, \cdots, v_m \in \mathcal{R}(l)$, and any $i \in \{1, \cdots, m\}$, there exists a routing $r' = v'_1, \cdots, v'_{m'} \in \mathcal{R}(l), j \in \{1, \cdots, m'\}$ such that $v_1 = v'_1$, $v_m = v'_{m'}$, $v_i = v'_j$, and $v_{i+1} \neq v'_{j+1}$.

Communication Schedule Constraints. Communication Schedule η is required to be periodic, namely a superframe of finite length must be defined for each frequency channel. Let F be the number of available frequency channels: we define $\eta = \langle \eta_1, \cdots, \eta_F \rangle$ the set of communication schedules. Each frequency channel can be characterized by a different number N_i of time slots. For this reason, we can define $\eta_i \colon \{1, \cdots, N_i\} \rightarrow 2^{E_{\mathcal{R}}}$. This implies that the schedule η is periodic, with period given by the least common multiple N of the set $\{N_i : i = 1, \cdots, F\}$. We infer that only one physical node can transmit in one time slot, for each frequency channel. That is, given a communication schedule $\eta_i(k) = \{e_1, \cdots, e_m\}$, then the sources of all scheduled edges are memory slots that correspond to the same physical node. We are thus assuming that no more than one physical node can transmit simultaneously without interfering with the others, namely each node interferes with any other node, and the interference graph is fully connected. However, in order to allow dedicated slots, it is possible to remove this assumption with a slight modification to the above constraint.

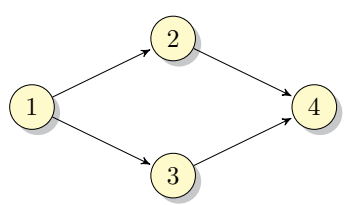
Data Flow and Control Computation Constraint. We require that all measured data is routed to the controller (gateway), that the controller computation is scheduled only when all measured data reach the controller, and that all control data is routed to the actuators, within the time duration of the superframe. Moreover, each possible routing must be scheduled, in order to permit to each node to decide where to route the data. We recall that $\mathcal{R}(i)$ is the set of routing paths from sensor i to the controller, and $\mathcal{R}(o)$ is the set of routing paths from the controller to actuator o. Let N be the length of the superframe. Let us consider without loss of generality a system $\mathcal{N}$ characterized by only one plant: we require that for any pair $\langle i, o \rangle \in \mathbb{I} \times \mathbb{O}$, any routing path $r_i = \{r_i(j)\}_{j=1}^{m_i} \in \mathcal{R}(i)$ and $r_o = \{r_o(j)\}_{j=1}^{m_o} \in \mathcal{R}(o)$, there exist integers $0 < k_i(1) < \cdots < k_i(m_i - 1) < k_c < k_o(1) <$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

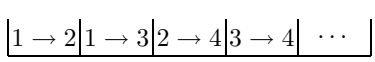
$\cdots < k_o(m_o - 1) < N$ such that:

(i) $\forall j \in \{1, \cdots, m_i - 1\}, \eta(k_i(j)) = (r_i(j), r_i(j+1))$,
(ii) $\mu(k_c) = Active$,
(iii) $\forall j \in \{1, \cdots, m_o - 1\}, \eta(k_o(j)) = (r_o(j), r_o(j+1))$.

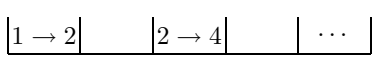
Interaction between scheduling and routing. A fundamental feature that characterizes WirelessHART is the interaction between the scheduling (superframe) and the routing (routing graph), and the associated data flow in the network. According to the specification, for each frequency channel a superframe must be defined. Moreover, for each slot, only one node is allowed to transmit, and only a subset of nodes is allowed to receive. The superframe must be designed so that all sensor data reach the gateway, and all control data reach the actuators within the duration of the superframe. We recall now that, according to the routing graph, each node has at least 2 neighbor choices to route a packet, for any destination node. Moreover, in order to allow each node to choose a routing path according to a local decision algorithm, it is necessary to schedule the nodes' transmissions so that any path can be used, namely so that each node can locally decide the next destination for routing his packet among the choices given by the routing graph. This means that such a definition of the superframe does not deterministically characterize the data flow in the network. The following example aims to clarify this concept, that is crucial for interpreting the semantics of data flow associated to transmission scheduling and graph routing:



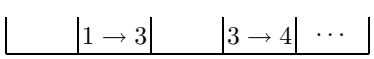
(a) Routing graph of node 4



(b) Superframe schedule



(c) Effective schedule, case 1



(d) Effective schedule, case 2

Fig. 8. Scheduling and routing of Example 1

Example 1: Node 1 needs to route a packet to node 4. Figure 8 illustrates the routing graph associated to the destination node 4. If node 1 tries to transmit data to node 2 and the transmission fails (no acknowledgement packet is received), then in the next superframe node 1 tries to send data to node 3. To allow this, we need to schedule data transmission both for the pair $\langle 1, 2\rangle$ and the pair $\langle 1, 3\rangle$. Moreover, to allow data transmission to node 4 both when node 2 or node 3 has been involved in the routing, we need to schedule data transmission both for the pair $\langle 2, 4\rangle$ and the pair $\langle 3, 4\rangle$, as illustrated in Figure 8. This example clearly shows that a scheduling is not associated to a deterministic data flow in the network, but it is associated to a set of possible data flows that depend on failure of nodes and transmission errors. In Figure 8 we illustrate the superframe schedule and the two possible schedules that can effectively occur in the network, according to the decision of node 1.

It is clear that, given a schedule s of the superframe, the communication schedule that occurs in each superframe is not deterministically identified. We define the set $\mathcal{L}(s)$ of all communication schedules that can non-deterministically occur for any superframe. As a first remark, notice that since a WirelessHART schedule is periodic over the length N of the superframe, then s can be expressed as a word of length N. Moreover, notice that every schedule $s' \in \mathcal{L}(s)$ corresponds to one choice of routing path for any pair sensor-controller and controller-actuator. For this reason, we can characterize the cardinality of $\mathcal{L}(s)$ as follows:

$$|\mathcal{L}(s)| = \prod_{i \in \mathbb{I}} |\mathcal{R}(i)| \cdot \prod_{o \in \mathbb{O}} |\mathcal{R}(o)|,$$

where $|\mathcal{R}(i)|$ is the number of routing paths from sensor i to the controller, and $|\mathcal{R}(o)|$ is the number of routing paths from the controller to actuator o. For this reason, $\mathcal{L}(s)$ is a finite language of finite words of length N.

The translation from s to $\mathcal{L}(s)$ is trivial: for each possible combination of routing paths, the effective schedule s' can be obtained from s by only keeping transmission schedule of edges that correspond to the considered routing paths. All other transmission schedules are removed. Iterating this procedure for all combinations of routing paths, the set $\mathcal{L}(s)$ is defined.

Let be given a multi-hop control network $\mathcal{N}$, and a schedule $s = \langle \eta, \mu \rangle$ that allows each node to locally decide the next destination for routing his packet among the choices given by the routing graph. Let N be the length of the superframe, we define for each $s' \in \mathcal{L}(s)$

$$\hat{A}(s') = \hat{A}(s'(N)) \cdot \hat{A}(s'(N-1)) \cdots \hat{A}(s'(1))$$

the matrix that corresponds to the dynamics of the system over the period of the superframe, when the effective schedule s' occurs. Then the dynamics of the control loop are modeled by the switching system

$$x(N(t+1)) = \hat{A}(s'(Nt))x(Nt), \quad s'(Nt) \in \mathcal{L}(s),$$

where $s'(Nt)$ is a non-deterministic switching signal that takes value in $\mathcal{L}(s)$, for each superframe period N. It is clear that, if $|\mathcal{R}(i)| = |\mathcal{R}(o)| = 1$ for any pair $(i, o) \in \mathbb{I} \cup \mathbb{O}$, then $\mathcal{L}(s) = \{s\}$ and the system is deterministic.

VI. Analysis Tools and Examples

To experiment with the proposed modeling approach, we implemented a Mathematica [12] based tool supporting it. The tool takes multi-hop control network models and transforms

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

```
■ Control Loops

Plant[1] = {Ap1, Bp1, Cp1};
Controller[1] = {Ac1, Bc1, Cc1};

Plant[2] = {Ap2, Bp2, Cp2};
Controller[2] = {Ac2, Bc2, Cc2};

loops = {{Plant[1], Controller[1]}, {Plant[2], Controller[2]}};

■ Wireless Network

topology :=
 expBiDi[{1 ↔ 4, 4 ↔ 5, 4 ↔ C, 2 ↔ 5, 5 ↔ C, 3 ↔ 6, 6 ↔ 7, 7 ↔ C}]

■ Routing

routing[y1,1] = {{1, 4, C}};
routing[y1,2] = {{2, 5, C}};
routing[u1,1] = {{C, 4, 1}};

routing[y2,1] = {{2, 5, C}} ;
routing[u2,1] = {{C, 7, 6, 3}};

■ Obtaining the Switched System

A function that maps EF×{Idle,Active} to matrices that model modes of the switched system

SW = SwitchedSystem1[loops, topology, routing];
```

Fig. 9. A description of the multi-hop control network discussed in Section III and a computation of the corresponding switched system with the Mathematica based tool.

them to switched systems.. In this section we describe the tool, demonstrate analysis techniques and present some experimental data.

A typical usage scenario

A typical use of the tool is by composing a Mathematica notebook such as the one outlined in Figure 9. We use a syntax, similar to the one described in Section III, to define the system. Once the definitions of the loops, network topology and the routing are given, one can automatically compute the switched system, described in Section IV-D, using the functions $\mathrm{SwitchedSystem}_l[\mathrm{loops}, \mathrm{topology}, \mathrm{routing}]$. The switched system, assigned to the variable SW, can then be analyzed, as shown in the following examples.

First example: Fixing a schedule and designing the controller accordingly

As a first example of how a formal model of a multi-hop control network can be used, we show a control design based on it. Consider the network depicted in Figure 3 where the first plant is a double integrator, modeled by the equation

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

with output, $y = x$. When sampling with time-step (sampling interval) h, the discrete-time system is

$$x^+ = \begin{pmatrix} 0 & h \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} h^2/2 \\ h \end{pmatrix}.$$

For the sake of the example, we choose $h = 1/20$.

The approach that we propose in this example is to fix a schedule for the system and design a controller that stabilizes the plant even with the delays induced by the network. To that end, we start with the cyclic schedule whose cycle is the following communication and computation sequences. As a communication schedule (i.e. a sequence of sets of edges of the memory slots graph) we choose:

$$\langle \emptyset, \{\langle 1, y_{1,1}\rangle \to \langle 4, y_{1,1}\rangle\}, \{\langle 2, y_{1,2}\rangle \to \langle 5, y_{1,2}\rangle\},$$
$$\{\langle 4, y_{1,1}\rangle \to \langle C, y_{1,1}\rangle\}, \{\langle 5, y_{1,2}\rangle \to \langle C, y_{1,2}\rangle\}, \emptyset,$$
$$\emptyset, \{\langle C, u_{1,1}\rangle \to \langle 4, u_{1,1}\rangle\}, \{\langle 4, u_{1,1}\rangle \to \langle 1, u_{1,1}\rangle\}, \emptyset\rangle$$

As a computation schedule (i.e. a sequence of modes of the controller) we choose:

$$\langle \mathrm{Idle}, \mathrm{Idle}, \mathrm{Idle}, \mathrm{Idle}, \mathrm{Idle}, \mathrm{Active}, \mathrm{Idle}, \mathrm{Idle}, \mathrm{Idle}, \mathrm{Idle}\rangle.$$

This pair of schedules model sending data from the plant to the controller, computing the control signal, and sending the result of the computation back to the actuator. These schedules are assumed to repeat periodically.

Towards a controller design, we first fix the matrices of the controller and leave the values of some entries as design parameters. Then, we use the Mathematica based tool for assigning values to these parameters. Specifically, the dynamics of the controller are defined by the equations $A_c = (K_3); B_c = (K_1, K_2); C_c = (1)$ where K_1, K_2 and K_3 are scalars, left as design parameters. To assign values to the parameters we compute the matrix CycleM, as shown in Figure 10. This matrix is the product of the matrices M[i] that model the dynamics of each step of the schedule (obtained from the switched system SW computed by the code in Figure 9). The product, assigned to the variable CyclicM, models the transformation of the state of the system through a cycle of the schedule.

As shown is Figure 10, the parameters K_1, K_2 and K_3 are resolved by assigning the poles of the matrix CyclicM. Because this matrix models the dynamics of the system through a cycle of the schedule, assigning its eigenvalues to be contained in the unit ball (of the complex plane) assures stability.

Second example: Verifying stability under non-deterministic schedules

As discussed in Section IV-D, scheduling in wireless control networks may not be deterministic. As an example, we consider a time varying scheduling constraint for the network depicted in Figure 3. Specifically, we assume that some of the times it is possible to send data from both nodes 1 and 2 simultaneously (e.g. because two radio frequencies are available) and some of the times data has to be sent sequentially, from 1 to 4 first and then from 2 to 5. While both the schedule that applies sequential messages and the schedule that applies parallel messages are stable (as can be verified by computing the eigenvalues of matrices similar to CycleM shown in Figure 10) it does not necessarily mean that any switching between them is stable (see e.g. [9]).

To guaranty stability, we apply a sufficient condition for stability of switched systems to verify that switching arbitrarily between the two schedules is safe. Specifically, we verify that

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

■ Computting dynamics of a schedule

```
commSch = {{}, {{1, y1,1} → {4, y1,1}},
   {{2, y1,2} → {5, y1,2}}, {{4, y1,1} → {C, y1,1}},
   {{5, y1,2} → {C, y1,2}}, {}, {}, {{C, u1,1} → {4, u1,1}},
   {{4, u1,1} → {1, u1,1}}, {}};

compSch = {Idle, Idle, Idle, Idle, Idle, Active,
   Idle, Idle, Idle, Idle};
```

```
M[i_] := SW[commSch[[i]], compSch[[i]]]
CycleM = M[10].M[9].M[8].M[7].M[6].M[5].M[4].
   M[3].M[2].M[1];
```

■ Solving the design parameters K_1, K_2 and K_3

```
sol =
 Solve[Eigenvalues[CycleM] ==
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 1/10, 2/10, 3/10},
  {K1, K2, K3}]
```

$$\left\{\left\{K_1 \to -\frac{504}{25}, K_2 \to -\frac{3452}{125}, K_3 \to \frac{3}{500}\right\}\right\}$$

Fig. 10. Computation of matrix representing dynamics of a schedule and using it to assign values to design parameters.

$\|C_{\sigma(7)} \cdots C_{\sigma(1)}\| < 1$ for every $\sigma \in \{1,2\}^7$ where C_1 and C_2 are matrices modeling the transformation of state variables through the first and the second schedule, respectively. This is, of course, a sufficient condition for stability (even exponential stability) under arbitrary switching, because it implies that every seven steps are contracting. The Mathematica code for this example is given in Figure 11.

Third example: Using compositional analysis for schedules design

One advantage of our modeling approach is that, because dynamics are defined for each control loop separately, it allows compositional analysis. As an example, we show how a system comprising of two control loops is analyzed, in a compositional manner, to obtain a joint schedule that renders both loops stable.

Consider the network depicted in Figure 12. Assume that both plants are double integrators with dynamics and controller as described above (in the first example of this section). Assume also that at most one node can send data at any time slot.

The design approach that we demonstrate in this example is as follows. First, we analyze each control loop separately to obtain scheduling constraints in the form of regular languages. Then, we use formal-languages based algorithms to compute the intersection of the constraints and obtain a joint schedule that is safe for both control loops.

Figure 13 shows the code for applying the compositional approach to schedule design. We use the Automata [13] package for Mathematica for formal languages manipulation. The first part of the code instantiates an automaton for each control loop, as follows. A set of schedules is obtained by all interleavings of idle steps into a base schedule, and an automaton is constructed whose language is the interleaved schedules that are stable.

■ Definition of two schedules and verification that both are stable

```
commSche[1] = {{}, {{1, y1,1} → {4, y1,1}},
   {{2, y1,2} → {5, y1,2}}, {{4, y1,1} → {C, y1,1}},
   {{5, y1,2} → {C, y1,2}}, {}, {}, {{C, u1,1} → {4, u1,1}},
   {{4, u1,1} → {1, u1,1}}, {}};

compSche[1] = {Idle, Idle, Idle, Idle, Idle, Active,
   Idle, Idle, Idle, Idle};

commSche[2] =
  {{}, {{1, y1,1} → {4, y1,1}, {2, y1,2} → {5, y1,2}},
   {{4, y1,1} → {C, y1,1}}, {{5, y1,2} → {C, y1,2}}, {},
   {}, {{C, u1,1} → {4, u1,1}}, {{4, u1,1} → {1, u1,1}}, {}};

compSche[2] = {Idle, Idle, Idle, Idle, Active, Idle,
   Idle, Idle, Idle};

Mn_[i_] := SW[commSche[n][[i]], compSche[n][[i]]] /. sol[[1]]

CM[1] = M1[10].M1[9].M1[8].M1[7].M1[6].M1[5].M1[4].
   M1[3].M1[2].M1[1];
CM[2] = M2[9].M2[8].M2[7].M2[6].M2[5].M2[4].M2[3].
   M2[2].M2[1];

If[ isStableMatrix[CM[2]],
 Print[Style["The first schedule is stable", Green]],
 Print[Style["The first schedule is not stable", Red]]]

If[ isStableMatrix[CM[2]],
 Print[Style["The second schedule is stable", Green]],
 Print[Style["The second schedule is not stable", Red]]]
```

The first schedule is stable

The second schedule is stable

■ Verification of stability under arbitrary switching

```
prod[seq_] := Dot @@ Reverse[CM /@ seq]
cond[sws_] := Norm[prod[sws]] < 1

test[H_] := If[ And @@ (cond /@ Sequences[2, H]),
  Print[Style["All products of length " <>
     ToString [H] <> " are contracting", Green]],
  Print[Style["Some product of length " <>
     ToString [H] <> " is not contracting", Red]]]

test[6]
test[7]
```

Some product of length 6 is not contracting

All products of length 7 are contracting

Fig. 11. Applying a sufficient condition for stability of switched systems to verify stability under non-deterministic network schedules.

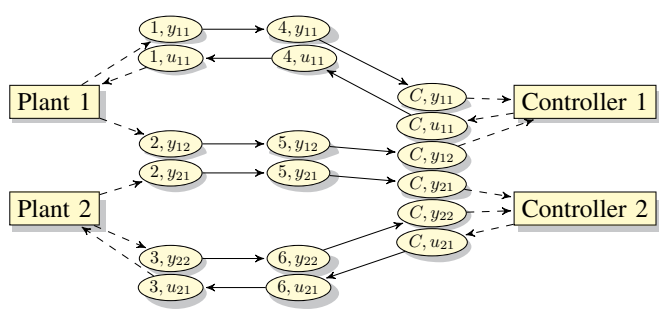


Fig. 12. Memory slots graph of a multi-hop control network with two symmetric double integrators.

The next step, in the code, is intersecting the constraints of both control loops. Note that we need to lift the automata to a common alphabet (pairs $\langle \sigma_1, \sigma_2 \rangle$ where σ_1 is a letter from the alphabet of the first automaton and σ_2 is a letter form the alphabet of the second automaton). This lifting is obtained by

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.

■ Compute automata of stable schedules for both subsystems

In[231]:=
```
Needs["Automata`automata`"]

Σ1 = ( {}                  Idle
       {{1, y1,1} → {4, y1,1}}  Idle
       {{2, y1,2} → {5, y1,2}}  Idle
       {{4, y1,1} → {C, y1,1}}  Idle
       {{5, y1,2} → {C, y1,2}}  Idle
       {}                 Active
       {{C, u1,1} → {4, u1,1}}  Idle
       {{4, u1,1} → {1, u1,1}}  Idle )  ;

Σ2 = ( {}                  Idle
       {{3, y2,1} → {6, y2,1}}  Idle
       {{2, y2,2} → {5, y2,2}}  Idle
       {{6, y2,1} → {C, y2,1}}  Idle
       {{5, y2,2} → {C, y2,2}}  Idle
       {}                 Active
       {{C, u2,1} → {6, u2,1}}  Idle
       {{6, u2,1} → {3, u2,1}}  Idle )  ;

baseSchedule = {2, 3, 4, 5, 6, 7, 8, 1};

aut1 = StableSchedulesDFA_{SW1,Σ1}[
   AddIdlesToWord[baseSchedule, 1]];
aut2 = StableSchedulesDFA_{SW2,Σ1}[
   AddIdlesToWord[baseSchedule, 1]];
```

■ Lift the automata to a common alphabet and compute intersection

In[237]:=
```
TR[v_] := Transpose[v];

compositionAlphabet =
  ArrayFlatten[
   ( TR[(1 2 3 4 5 6 7 8)]        1
     TR[(1 2 3 4 5 6 7 8)]        6
     1                  TR[(2 3 4 5 7 8)]
     6                  TR[(2 3 4 5 7 8)]
     3                  3
     5                  5 )  ];

extAut1 = ExtendAlphabetDFA[aut1, compositionAlphabet,
   1];
extAut2 = ExtendAlphabetDFA[aut2, compositionAlphabet,
   2];

inter = MinimizeFA[IntersectionFA[extAut1, extAut2]];
```

■ Select a schedule in the intersection and print it

In[242]:=
```
s = ToIndex[LanguageFA[inter, 13][[1]]];

explain[{i_, j_}] :=
 {Union[Σ1[[i]][[1]], Σ2[[j]][[1]], {Σ1[[i]][[2]], Σ2[[j]][[2]]}}
explain /@ compositionAlphabet[[s]]
```

Out[244]=

$\{\{1, y_{1,1}\} \to \{4, y_{1,1}\}\}$	{Idle, Idle}
$\{\{3, y_{2,1}\} \to \{6, y_{2,1}\}\}$	{Idle, Idle}
$\{\{2, y_{1,2}\} \to \{5, y_{1,2}\}, \{2, y_{2,2}\} \to \{5, y_{2,2}\}\}$	{Idle, Idle}
$\{\{4, y_{1,1}\} \to \{C, y_{1,1}\}\}$	{Idle, Idle}
$\{\{6, y_{2,1}\} \to \{C, y_{2,1}\}\}$	{Idle, Idle}
$\{\{5, y_{1,2}\} \to \{C, y_{1,2}\}\}$	{Idle, Idle}
$\{\{5, y_{2,2}\} \to \{C, y_{2,2}\}\}$	{Active, Idle}
$\{\}$	{Idle, Active}
$\{\{C, u_{1,1}\} \to \{4, u_{1,1}\}\}$	{Idle, Idle}
$\{\{C, u_{2,1}\} \to \{6, u_{2,1}\}\}$	{Idle, Idle}
$\{\{4, u_{1,1}\} \to \{1, u_{1,1}\}\}$	{Idle, Idle}
$\{\{6, u_{2,1}\} \to \{3, u_{2,1}\}\}$	{Idle, Idle}
$\{\}$	{Idle, Idle}

Fig. 13. Applying compositional analysis to design a schedule for a system with two control loops.

applying the function ExtendAlphabetDFA which implements the lifting in the standard way. Once the lifting is done, we take the intersection of the languages to obtain a joint schedule. The first word of the automaton (in length-lex order) is extracted and displayed.

We remark that compositional analysis allows synchronizing node transmissions to send data of different plants simultaneously. In fact, the third element of the composition scheduling illustrated in Figure 13 triggers a simultaneous transmission of data $y_{1,2}$ and $y_{2,2}$. This is allowed, since they are transmitted from the same physical node 2 to the physical node 5.

VII. Conclusions

We proposed a compositional mathematical framework for modeling and analysing multi-hop communication networks. We separated control, topology, routing, and scheduling and proposed formal syntax and semantics for the dynamics of the composed system. Our model allows separate analysis of control loops towards a compositional design of schedules that cope with competing needs of communication and computation resources. We showed that the WirelessHART specification fits our model, and we illustrated an experimental tool that can be used both for verification and design. The tool, along with the code for the examples, is available as a Mathematica notebook at www.seas.upenn.edu/~gera.

References

[1] G. Weiss and R. Alur, "Automata based interfaces for control and scheduling," in *Hybrid Systems: Computation and Control, HSCC*, 2007, pp. 601–613.

[2] R. Alur and G. Weiss, "Regular specifications of resource requirements for embedded control software," in *Real-Time and Embedded Technology and Applications Symposium, RTAS*, 2008, pp. 159–168.

[3] G. Walsh, H. Ye, and L. Bushnell, "Stability analysis of networked control systems," *Control Systems Technology, IEEE Transactions on*, vol. 10, no. 3, pp. 438–446, 2002.

[4] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *Control Systems Magazine, IEEE*, vol. 21, no. 1, pp. 84–99, 2001.

[5] J. K. Yook, D. M. Tilbury, N. R. Soparkar, E. Syst, and E. S. Raytheon, "Trading computation for bandwidth: Reducing communication indistributed control systems using state estimators," *Control Systems Technology, IEEE Transactions on*, vol. 10, no. 4, pp. 503–518, 2002.

[6] K. Aström and B. Wittenmark, *Computer-controlled systems: Theory and Design*. Prentice Hall, 1997.

[7] M. Andersson, D. Henriksson, A. Cervin, and K. Arzen, "Simulation of wireless networked control systems," in *Conference on Decision and Control and European Control Conference. CDC-ECC*, 2005, pp. 476–481.

[8] A. J. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, 1st ed. Springer, Dec. 1999.

[9] D. Liberzon, *Switching in Systems and Control*. Boston, MA: Birkhäuser, 2003.

[10] R. Alur and G. Weiss, "RTComposer: a framework for real-time components with scheduling interfaces," in *International conference on EMbedded SOFTtware, EMSOFT*, 2008, pp. 159–168.

[11] "TDMA data-link layer specification," HART communication foundation, HCF SPEC 075 Revision 1.0, 2007.

[12] S. Wolfram, *The Mathematica Book*. Wolfram Media, August 2003.

[13] K. Sutner, "Automata, a hybrid system for computational automata theory," in *International Conference on Implementation and Application of Automata, CIAA*, 2002, pp. 217–222.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on July 6, 2009 at 14:25 from IEEE Xplore. Restrictions apply.