Distributed Control for Cyber-Physical Systems

Rahul Mangharam, *Member, IEEE* and Miroslav Pajic, *Member, IEEE*.

***Abstract*—Networked Cyber-Physical Systems (CPS) are fundamentally constrained by the tight coupling and closed-loop control and actuation of physical processes. To address actuation in such closed-loop wireless control systems there is a strong need to re-think the communication architectures and protocols for maintaining stability and performance in the presence of disturbances to the network, environment and overall system objectives. We review the current state of network control efforts for CPS and present two complementary approaches for robust, optimal and composable control over networks. We first introduce a computer systems approach with Embedded Virtual Machines (EVM), a programming abstraction where controller tasks, with their control and timing properties, are maintained across physical node boundaries. Controller functionality is *decoupled* from the physical substrate and is capable of runtime migration to the most competent set of physical controllers to maintain stability in the presence of changes to nodes, links and network topology.**

We then view the problem from a control theoretic perspective to deliver fully distributed control over networks with Wireless Control Networks (WCN). As opposed to traditional networked control schemes where the nodes simply route information to and from a dedicated controller, our approach treats the network itself as the controller. In other words, the computation of the control law is done in a fully distributed way *inside the network*. In this approach, at each time-step, each node updates its internal state to be a linear combination of the states of the nodes in its neighborhood. This causes the entire network to behave as a linear dynamical system, with sparsity constraints imposed by the network topology. This eliminates the need for routing between "sensor $\rightarrow$ channel $\rightarrow$ dedicated controller/estimator $\rightarrow$ channel $\rightarrow$ actuator", allows for simple transmission scheduling, is operational on resource constrained low-power nodes and allows for composition of additional control loops and plants. We demonstrate the potential of such distributed controllers to be robust to a high degree of link failures and to maintain stability even in cases of node failures.

***Index Terms*—Networked control systems, decentralized control, wireless sensor networks, structured systems, in-network control, network coding, cooperative control**

I. Introduction

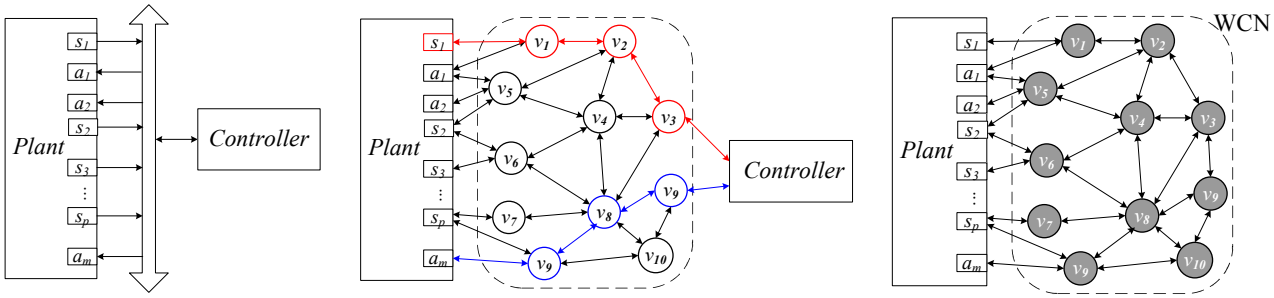
Time-critical and safety-critical automation systems are at the heart of essential infrastructures such as oil refineries, automated factories, logistics and power generation systems. To meet the reliability requirements, automation systems are traditionally severely constrained along three dimensions, namely, operating resources, scalability of interconnected systems and flexibility to mode changes. Oil refineries, for example, are built to operate without interruption for over 25 years and can never be shutdown for preventive maintenance or upgrades. They are built with rigid ranges of operating throughput and require a significant re-haul to adapt to changes in crude oil quality and market conditions. This rigidity has resulted in systems with limited scope for re-appropriation of resources during faults and retooling to match design changes on-demand. For example, automotive assembly lines lose an average of $22,000 per minute of downtime during system faults [1]. This has created a culture where the operating engineer is forced to patch a faulty unit in an ad hoc manner which often necessitates masking certain sensor inputs to let the operation proceed. This process of unsystematic alteration to the system exacerbates the problem and makes the assembly line difficult and expensive to operate, maintain and modify.

Embedded Wireless Sensor-Actuator-Controller (WSAC) networks are emerging as a practical means to monitor and operate automation systems with lower setup/maintenance costs. While the physical benefits of wireless, in terms of cable replacement, are apparent, plant owners have increasing interest in the logical benefits. With multi-hop WSAC networks, it is possible to build *Wireless Plug-n-Play Automation Systems* which can be swapped in and efficiently reconnect hundreds of I/O lines. Such modular systems can be dynamically assigned to be primary or backup on the basis of available resources or availability of the desired calibration. Modularity allows for incremental expansion of the plant and is a major consideration in emerging economies. WSAC networks allow for runtime configuration where resources can be re-appropriated on-demand, for example when throughput targets change due to lower electricity price during off-peak hours or due to seasonal changes in end-to-end demand.

The current generation of embedded wireless systems has largely focused on open-loop sensing and monitoring applications. To address actuation in closed-loop wireless control systems there is a strong need to re-think the communication architectures and protocols for reliability, coordination and control [2]. Wireless networked control systems, or Networked Cyber-Physical Systems (Networked-CPS), fundamentally differ from standard distributed systems in that the dynamics of the *network* (variable channel capacity, probabilistic connectivity, topological changes, node and link failures) can change the operating points and physical dynamics of the *closed-loop system* [3], [4]. The most important objective of control in Networked-CPS is to provide stability of the closed-loop system. It is therefore necessary for the network (along with its interfaces to sensors and actuators) to be able to provide some form of guarantee of the control system's stability in the face of the non-idealities of the wireless links and the communication constraints of the wireless swarm network. A secondary goal in Networked-CPS is to allow for composition of additional controllers and plants within the same network without requiring reconfiguration of the entire network operation.

R. Mangharam and M. Pajic are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA 19014. Email: {rahulm, pajic}@seas.upenn.edu.

This work has been partially supported by the NSF-CNS 0931239 and NSF-MRI 0923518 grants.



(a) Wired Network Control (b) Wireless Network Controlled System (c) Wireless Control Network

Fig. 1. Standard architectures for Networked Control Systems; (a) Wired system with a shared bus and dedicated controller; (b) Red links/nodes - routing data from the plant's sensors to the controller; Blue links/nodes - routing data from the controller to the plant's actuators; (c) A multi-hop wireless control network used as a distributed controller.

The most common approach to incorporating Networked-CPS into the feedback loop is to use it primarily as a communication medium: the nodes in the network simply route information to and from one or more *dedicated controllers*, which are usually specialized CPUs capable of performing computationally expensive procedures (see Fig. 1(b)). The use of dedicated controllers imposes a routing requirement along one or more fixed paths through the network, which must meet the stability constraints, encapsulated by end-to-end delay requirements [5], [6]. However, this assignment of routes is a static setup, which commonly requires global reorganization for changes in the underlying topology, node population and wireless link capacities.

Routing couples the communication, computation and control problems [7], [8], [9]. Therefore, when a new route is required due to topological changes, the computation and control configurations must also be recalculated. Merely inserting a WNCS into the standard network architecture "sensor $\rightarrow$ channel $\rightarrow$ controller/estimator $\rightarrow$ channel $\rightarrow$ actuator" requires the addition of significant software support [10], [11], as the overhead of completely recomputing the computation and control configurations, due to topological changes or packet drops, is too expensive and does not scale.

A. Wireless Control Design Challenge

Providing closed-loop stability and performance guarantees for Networked CPS is a challenging problem. On one hand, the control systems community typically abstracts away the systems details and solves the problem for semi-idealized networks with approximated noise distributions and link perturbations [3]. While this approach provides mathematical certainty of the properties of the network, it fails to provide a systematic path to real-world network design. On the other hand, the network systems community uses hardware and software approaches to address open-loop issues, but these fail to provide any guarantees to maintaining stability and performance of closed-loop control. We propose a control scheme over wireless networks that provides closed-loop stability and optimality, with respect to standard metrics, while maintaining ease of implementation in real-world networks.

While there has been considerable research in the general area of wireless sensor networks, a majority of the work has been on open-loop and non-real time applications. As we extend the existing programming paradigm to closed-loop control applications with tight timeliness and safety requirements, we identify four primary challenges with the design, analysis and deployment of WSAC networks:

1. The current approaches of programming motes in the event-triggered paradigm [12] are tedious for control networks. Time-triggered architectures are required as they naturally integrate communication, computation, and physical aspects of control networks [13], [14].

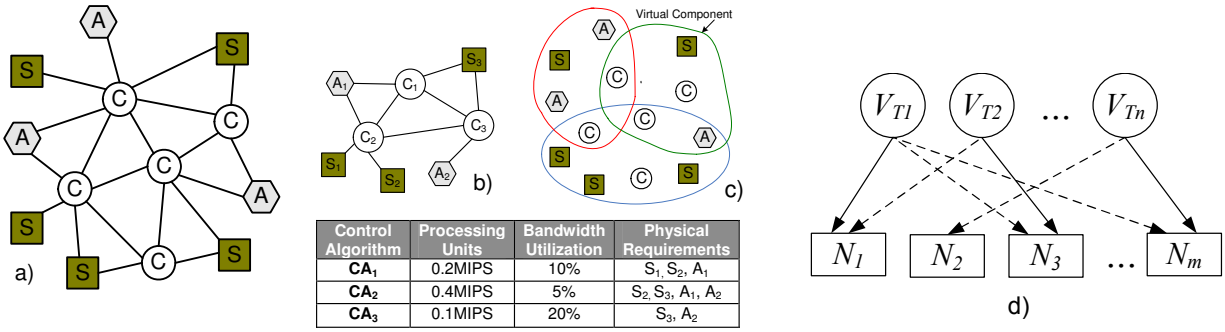
2. Programming of sensor networks is currently at the physical node-level [15] and is the key reason responsible for the lack of robustness for higher-level control applications.

3. Design of networked control systems with flexible topologies is hard with physical node-level programming, as the set of tasks (or responsibility) is associated with the physical node [16].

4. Fault diagnostics, repair and recovery are manual and template-driven for a majority of networked control systems [17], [18]. Runtime adaptation is necessary to maintain the stability and performance of the higher-level control system.

5. Furthermore, the networks might be shared among control loops (i.e., a node may be involved in several feedback loops), and new feedback loops may be added at run-time. Adding new communication loops in a standard wireless network control system could affect the performance of the existing loops, and the system must be analyzed as a whole. Although techniques have been developed for compositional analysis of such networks (e.g., [7]), their complexity limits their use. Therefore, it is necessary to derive a *composable* control scheme, where control loops can be easily added and a simple compositional analysis can be performed at run-time, to ensure that one loop does not affect the performance of other loops.

The applications of interest in this work are industrial process control systems (such as natural gas refineries and paper pulp manufacturing plants) and building automation systems. In general, the plant time-constants are on the order of several seconds to a few minutes and the control network is expected to operate at rates of hundreds of milliseconds. While such plants may have as many as 80,000 to 110,000 control loops, they are organized in a hierarchal manner such that networks span 10-20 wireless nodes (per gateway) for low-level control. Therefore, in this work we focus on the networks with up to a few tens of nodes.



Control Algorithm	Processing Units	Bandwidth Utilization	Physical Requirements
CA_1	0.2MIPS	10%	S_1, S_2, A_1
CA_2	0.4MIPS	5%	S_2, S_3, A_1, A_2
CA_3	0.1MIPS	20%	S_3, A_2

Fig. 2. (a) A wireless sensor, actuator and controller network. (b) Algorithm assignment to a set of controllers, each mapped to the respective nodes. (c) Three Virtual Components, each composed of several network elements. (d) Decoupled virtual tasks and physical nodes with runtime task mapping.

B. Contributions

While providing a review of classical and recent approaches for control over wireless networks, we present two complementary approaches on maintaining stability in the presence of environment and network disturbances. The first approach adopts a "computer systems" perspective on the design of robust architectures for embedded wireless control and actuation. We call this scheme Embedded Virtual Machines (see Fig. 2) which provides software mechanisms to decouple controller functionality from the physical node - thus providing resilience to node, link and topology changes. The second approach adopts a "control theoretic" perspective on distributed control within the network (see Fig. 1(c)). This provides control mechanisms to remove controller functionality from a dedicated node to all nodes in the network - thus eliminating the need for routing and guaranteeing stability and optimal control in the presence of link, node and topology changes.

1) ***Embedded Virtual Machines****:* Current approaches for robust networked control [4] require the underlying network to satisfy a minimal set of requirements (e.g. guaranteed packet deliver rate, upper bound on network induced delay) and reduce the network model to that of a single channel with random delays. In addition, they do not address the spatial aspects of the network, i.e., how changes in the network topology affect the closed-loop system performance.

As the links, nodes and topology of wireless systems are inherently unreliable, such time-critical and safety-critical applications require programming abstractions where the tasks are assigned to the sensors, actuators and controllers as a *single component*, rather than statically mapping a set of tasks to a specific physical node at design time (as shown in Fig. 2). Such wireless controller grids are composed of many nodes that share a common sense of the control application but without regard to physical node boundaries. Our approach, is to *decouple* the functionality (i.e., tasks) from the inherently unreliable physical substrate (i.e., nodes) and allow tasks to migrate/adapt (Fig. 3) to changes in the topology.

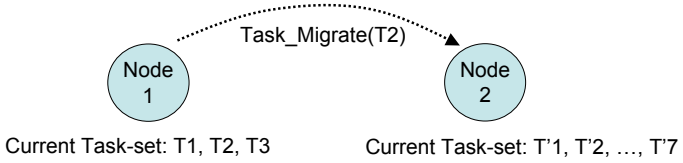


Fig. 3. Task migration for real-time operation (instructions, stack, data & timing/fault tolerance meta-data) on one physical node to another.

To this end, we introduced the Embedded Virtual Machine (EVM), a powerful and flexible programming abstraction where a Virtual Component (VC) and its properties are maintained across node boundaries [6], [19], as shown in Fig. 2(c). EVMs differ from classical system virtual machines. In the enterprise or on PCs, one (powerful) physical machine may be partitioned to host multiple virtual machines for higher resource utilization. On the other hand, in the embedded domain, an EVM is composed across multiple physical nodes with the goal to maintain correct and high-fidelity operation even under changes in the physical composition of the network. The goal of the EVM is to maintain a set of *functional invariants*, such as a control law and *para-functional invariants* such as timeliness constraints, fault tolerance and safety standards across a set of controllers given the spatio-temporal changes in the physical network. Thus, the EVM introduces new degrees of freedom, task migration and routing which facilitates, at runtime, the network configuration (operating point, conditions) to meet the requirements of the networked control algorithms. However, the EVM does not provide explicit guarantees but only finds the optimal operation configuration in terms of routing and task assignment.

2) ***Distributed Control over Wireless Networks****:* We consider the problem of stabilizing a plant with a multi-hop network of resource constrained wireless nodes. We introduce the concept of a *Wireless Control Network* (WCN) [20], which is a paradigm change for distributed control over a wireless network. In a WCN the entire network *itself* acts as a controller, as the computation is spread over the whole network, instead of assigning a particular node with the execution of the control procedure. We devise a numerical design procedure that produces the coefficients of the linear combinations for each node and actuator to apply in order to stabilize the plant. The radio connectivity between nodes in the network induces topological constraints to the control algorithm, and this topology determines whether it is even possible to stabilize the system with the use of linear iterative strategies. In addition, we describe the method that can be used to synthesize an optimal WCN, with respect to the standard cost functions.

Given the fundamental unreliability of wireless communication, the WCN method handles topological constraints while maintaining mean square stability for packet drop rates ***up to 20%*** for a specific network topology and plant. This bridges

the gap between the basic WCN and the theoretical upper bound of robustness to packet drops [21]. We also show a method to synthesize a WCN robust to a certain level of node failures, and then extend the synthesis procedures to allow for the use of the WCN for control of continuous-time plants. Finally, we illustrate the use of the WCN on a real-world industrial case study, for control of a distillation column.

While in the past efforts, we consider scenarios where the network topology is already set, in recent efforts [22], [23] we have investigated a dual problem, "how to synthesize the network so that a stable WCN configuration exists?" The topological conditions from [22], along with the results from [20] provide the essential building blocks for an integrated decentralized wireless control network design framework. Early experiments in an industrial process control case study of a distillation column in a process-in-the-loop testbed demonstrate optimal control of continuous-time physical processes which maintain system stability under the presence of node and link failures.

Finally, in [24] we addressed security challenges for the WCN and presented a method to design an Intrusion Detection System (IDS) within the Wireless Control Network (WCN) architecture. The IDS is responsible for observing the transmissions of certain nodes in the network in order to (a) recover the outputs of the plant (e.g., for fault-diagnosis purposes), and (b) detect and identify data modification attacks by nodes in the network. We showed that the WCN scheme allows malicious behavior to be identified by examining the transmissions of only a *subset* of the network nodes, provided that the network topology satisfies certain conditions (more details can be found in [24]).

Organization: The remainder of the paper is organized in two parts covering EVM (Section **??**) and WCN (starting in Section IX), respectively. It is worth noting here that these two approaches for control over wireless networks are complementary, and thus they could be read in any order. Section II presents an overview of the EVM and its automated design flow from a control problem specification to binding controller tasks to a group of nodes within a VC. Sections III - V present the architecture of the EVM, task assignment during network changes and runtime procedures to migrate controller functionality while maintaining stability during topological changes. We describe the implementation on real hardware in Section VI and a case study in Section VII.

Sections IX describes the concept of the WCN followed by Sec. X - XIII covering optimal control over WCN, robust control over WCN and a case study to show how the WCN can be used in an industrial, process control application.

II. Part I: Embedded Virtual Machines

While wireless system engineers optimize the physical, link and network layers to provide an expected packet error rate, this does not translate accurately to stability of the control problem at the application layer. For example, planned and unplanned changes in the network topology with node/link failures are currently not easily captured or specifiable in the metrics and requirements for control engineers. For a given plant connected to its set of controllers via wireless links (see Figure 1(a-b)) it is necessary that the controller process the sensor inputs and perform actuation within a bounded sampling interval. While one approach is to design specialized wireless control algorithms that are robust to a specified range of packet errors [4], [3], it is non-trivial to design the same for frequent topological changes. Furthermore, it is difficult to extend the current network infrastructure to add/remove nodes and to redistribute control algorithms to suit environmental changes such as battery drain for battery-operated nodes, increased production during off-peak electricity pricing, seasonal production throughput targets and operation mode changes.

The EVM approach is to allow control engineers to use the same network control algorithms on the wireless network without knowledge of the underlying network protocols, node-specific operating systems or hardware platforms. The virtual machine executing on each node (within the VC) instruments the VC to adapt and reconfigure to changes while ensuring the control algorithm is within its stability constraints. This approach is complementary to the body of network control algorithms as it provides a logical abstraction of the underlying physical node topology.

A. Network CPS Related Work

There have been several variants of virtual machines, such as Maté [25], Scylla [26] and SwissQM [27], and flexible operating systems, such as TinyOS [12], SOS [28], Contiki [29], Mantis [30], Pixie [31] and LiteOS [32], for wireless sensor networks. The primary differences that set EVM apart from prior work is that it is centered on real-time operation of controllers and actuators. Within the design of the EVM's operating system, link protocol, programming abstractions and operation, timeliness is a first-class citizen and all operations are synchronized. The EVM does not have a single node-perspective of mapping operations to one virtualized processor on a particular node but rather maintains coordinated operation across a set of controllers within a virtual component. The Virtual Node Layer [33] provides a programming abstraction where each virtual node is identified with a particular region and it is emulated by one of the physical nodes in its region. On the other hand, EVM uses several physical nodes and allows the user to consider the virtual component as a single logical entity.

In the last few years, several different systems for macro-programming in WSN have been developed. [15] have defined a set of abstractions representing local communication between nodes in order to expose control over resource consumption along with providing feedback on its performance. An extension of these ideas is used to develop Regiment [34], a high-level language based on the functional reactive programming. Kairos [35] allows a programmer to describe code execution for each of the nodes in a network using a centralized approach where details about code generation, remote data access and management along with node interactions are hidden from the programmer. EVM is not a generic macroprogramming system as it focuses on closed-loop control with native runtime support for task assignment and migration.

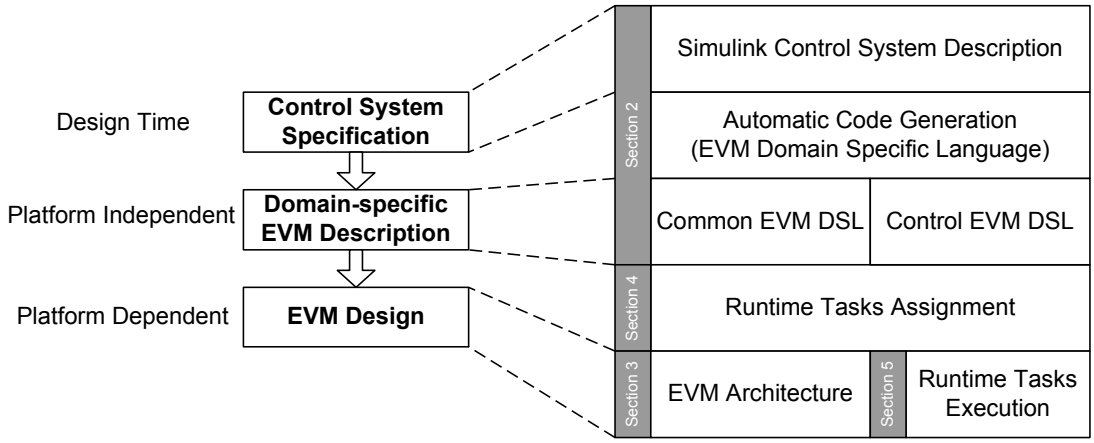


Fig. 4. Embedded Virtual Machines (EVM) design flow

The development of control algorithms able to deal with the unreliability of the wireless channel for Networked Control Systems (NCSs) is an active area of research in the control systems community [4], [3], [36]. Few efforts consider networked control over arbitrary topologies ([16], [37], [38]). In these articles, the authors assume the existence of a single actuation point and a single sensing point on the plant. They show that the optimal position of the controller is at the actuation point, while ignoring the wireless channel in the estimation of the plant's state. In general case, the problem of assigning the best location of the controller node is very complex. Finally, Etherware [11] presents challenges in software development for NCSs along with abstractions and architectures used to implement control algorithms for NCSs. The authors describe a middleware for control systems but do not provide algorithms which might be used to guarantee that designed middleware satisfies requirements for the control algorithms.

B. EVM Design Flow

Our focus is on the design and implementation of wireless controllers and in providing such controllers with runtime mechanisms for robust operation in the face of spatio-temporal topological changes. We focus exclusively on controllers and not on sensors or actuators, as the latter are largely physical devices with node-bound functionality. A three-layered design process is presented to allow control engineers to design wireless control systems in a manner that is both largely platform/protocol/hardware/architecture independent and extensible to different domains of control systems (in process, discrete, aviation, medical, etc.). This section describes the design flow from a control problem formulation in Simulink, automatic translation of control models from Simulink to the platform-independent EVM interpreter-based code and finally to platform-dependent binaries (see Fig. 4). These binaries are assigned to physical nodes within a VC using assignment and scheduling algorithms presented in Section IV. The binaries are executed as Virtual Tasks within the platform dependent architecture described in Section III.

At design time, control systems are usually designed using software tools, such as Matlab/Simulink, that incorporate both modeling and simulating capabilities. Therefore, to automatize the design flow the EVM is able to automatically generate functional models from the Simulink control system description. These functional models define the processes by which input sampled data is manipulated into output data for feedback and actuation. The models are represented by generated code and meta data for *platform and node independent system description*. This allows a system designer to exclusively focus on the control problem design. Beside the functional description in the platform-independent and domain specific language (DSL), from the Simulink model the EVM design flow automatically extracts additional para-functional properties like *timing* and *inter task dependencies*. These properties, along with the functional description are used to define a *platform optimized binary* for each Virtual Task (VT).

C. Platform Independent Domain Specific Language

To generate functional description of the designed system, the EVM programming language is based on FORTH, a structured, stack-based, extensible, interpreter-based programming language [39]. Since the goal of the EVM design is to allow flexibility and designing utilities independent of chosen programming language, the intermediate programming language is not constrained to the EVM programming language. The interpreter used to execute modules described in the EVM programming language can also execute precompiled binaries. The EVM implementation, presented in Section VI, executes binaries derived from embedded C code. This enables execution of code binaries developed in other languages used to describe control system implementation.

The use of the EVM intermediate programming language enables *domain-specific constructs*, where basic programming libraries are related to the type of application that is being developed. For example, for use in embedded wireless networks for industrial control we developed two predefined libraries, Common EVM and Control EVM (a full list of API's is provided in [40]). Common EVM (Fig. 5(b)) is based on the standard FORTH library [39]. Beside the : word, used to define new words, all other words can be separated into the following categories: 1) arithmetic operations, 2) logical operations, 3) memory manipulation, 4) sensor and actuator handling, and 5) networking. Control EVM (Fig. 5(a)) contains functionalities widely used to develop control applications. First three words

- Dictionary consists of predefined blocks/functions that are usually used for control systems description
- PID (K_P K_I K_D *addr_in addr_out* –)
- PI (K_P K_I *addr_in addr_out* –)
- TF (*m n addr_alpha addr_beta addr_in addr_out* –)

$$G(z)=\frac{N(z)}{D(z)} \quad N(z)=\alpha_0+\alpha_1 z^{-1}+\alpha_2 z^{-2}+...+\alpha_m z^{-m}, \quad D(z)=\beta_0+\beta_1 z^{-1}+\beta_2 z^{-2}+...+\beta_n z^{-n}.$$

- LTI (*p m n addr_A addr_B addr_C a_in a_x a_out* -)

$$x[k+1]=Ax[k]+Bu[k],$$
$$y[k]=Cx[k], A\in R^{n\times n}, B\in R^{n\times m}, C\in R^{p\times n}$$

(a) Control-EVM

- Arithmetic operations (on 16bit)
- Logical operations
- Comparison and testing
- Controlling programming flow
- Memory manipulation
- Sensor/Actuator handling
  - RDSensG (sensID – n1)
  - RDSensL (sensID – n1)
  - WRActG (value actID –)
  - WRActL (value actID –)
- Networking
  - PktSendG (addr n nodeID -)
- Task handling
  - TaskActivate (addrTCB actID –)

(b) Common-EVM

Fig. 5. EVM platform-independent and domain-specific language for expressing functional and timing description of Simulink models.

specified in Fig. 5(a) are used for Singe-Input-Single-Output (SISO) systems. Although these words can be described using the LTI word (describing Linear Time Invariant systems), their wide use in control systems recommended their specific use.

The extensibility of the EVM allows definition of additional domain-specific libraries such as Automotive EVM, Aviation EVM or Medical EVM libraries, which will contain functionalities specific to each of these application fields. Using EVM libraries, the code generator creates a system description from a predefined components, thus creating a task description file for each of the Virtual Tasks.

D. Control Problem Synthesis: From Simulink to Platform Independent Specification

We now describe the procedure to automatically extract the functional description of a VT from a Simulink design. Within Simulink, each block (and, thus, the model itself) is represented as a hierarchical composition of other Simulink blocks, either *subsystems* or *library-defined* blocks. This organization of Simulink models allows for a natural extraction of a structured functional description using predefined words from the platform-independent EVM DSL dictionary. When a new Simulink block is defined as a composition of previously defined blocks, a new word is defined for the EVM functional description using previously defined words. The process is repeated until a level is reached where all words belong to the EVM dictionary.

A VT description is obtained by parsing the Simulink model file. This is done by searching for new block definitions along with the interconnections between blocks. In a Simulink model file (i.e., mdl file) blocks are presented as shown in Fig. 6(c) and Fig. 6(d) where BlockType parameter describes whether the block is a part of the Simulink library or a subsystem, consisting other Simulink blocks. To extract the VT description we require that the task is implemented in a singular, discrete-time Simulink subsystem, such as the example shown in Fig. 7. The synthesis of the platform-independent specification from the model is carried out in three steps:

(1) **Definition of intermediate words and variables:** Each block i is associated with a word W_i from the EVM DSL, where the output of the block is assigned to a variable var_i. To illustrate this consider the extended PID controller from Fig. 7. The outputs of all intermediate blocks are assigned to variables as shown in Fig. 7. For example, the EVM description of block "Sum1" is described with word W_8 and its output with variable var_8. As the EVM DSL is stack-based with reversed Polish notation the block is described as:

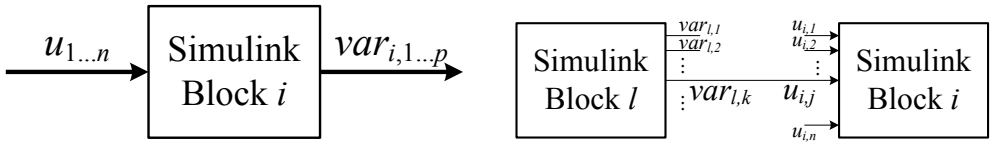
$$: \ W_8 \ R2 \ out3 \ ? \ NEG \ sum \ var_8 \ @ \ ;$$

where ? and @ are read and write operators respectively. In general case, for a block presented in Fig. 6(a) the parser defines the following word:

$$: \ W_i \ u_1 \ ? \ u_2 \ ? \ ...u_n \ ? \ coeffs \ BlockWord \ var_{i_1}\} \ @ \ var_{i_2} \ @ \ ... \ var_{i_p} \ @ \ ;$$

where **BlockWord**, depending on BlockType, corresponds to either a predefined word (if a library block is used) or a new word that needs to be defined using the same parser algorithm (if the block is a subsystem). Variables presented as **coeffs** are extracted from the 'Block Specific' data in cases when they are contained in the block description (from Figure 5(c),(d)), along with initial values for variables $\mathbf{var_i}$. For example, consider definition of word $\mathbf{W_4}$. Since block *PID_controller1* contains coefficients for K_p, K_i and K_d their values are included in the definition. Finally, in the previous formulation variables $u_{[1..n]}$ are replaced with appropriate system variables with respect to connections between blocks. To illustrate this consider a connection (i.e., line) between blocks from Fig. 6(b). Simulink defines the *Line* as in Fig. 6(e). Thus, for *Simulink Block i*, in the definition of word $\mathbf{W_i}$ each variable $u_{i,j}$ is replaced with appropriate variable $var_{l,k}$.

(2) **Composing extracted words:** The intermediate words are composed to create functional description of the system (e.g., $\mathbf{VT_{ctrl}}$). The parser is recursively executed for all subsystems till all words are part of the library. The description for the example from Fig. 7 is presented in Step 2, Fig. 8. It is worth noting that the intermediate words are executed in the blocks' execution order for the Simulink model. The order is



(a) General model of a block

(b) Interconnection between blocks

Block{
BlockType *Gain*
Name "*Integral_Gain*"
Position $[120, 93, 160, 117]$
Block specific data $\cdots$
}

(c) Simulink description of a library block

Block{
BlockType *SubSystem*
Name "*PID_controller*"
Ports $[n, p]$
Block specific data $\cdots$
}

(d) Simulink description of a subsystem

Line{
SrcBlock "*Simulink Block l*"
SrcPort k
DstBlock "*Simulink Block i*"
DstPort j
}

(e) Simulink description of a link between two blocks

Fig. 6. General relations between Simulink blocks

either specified explicitly in the model or determined implicitly based on block connectivity and sample time propagation [41].

(3) **DSL code optimization:** Intermediate blocks with elementary functions can be pruned in a single word. For the example from Fig. 7 the optimized description is shown in Step 3, Fig. 8. Words $\mathbf{W3}, \mathbf{W4}, \mathbf{W5}$ and $\mathbf{W8}, \mathbf{W9}, \mathbf{W10}$ are combined into a single word ($\mathbf{W3}$ and $\mathbf{W8}$, respectively). Also, instead of word $\mathbf{W6}$ and variable $\mathbf{var6}$, $\mathbf{W1}$ and $\mathbf{var1}$ are used. The code optimization reduces the number of defined words and used variables. Currently, the optimization is restricted to a small set of control system configurations. A more general approach is an avenue for future work.

As our intention is to map the control problem to a scheduling problem, timing parameters (i.e., period and worst-case execution time) are also extracted from the model. We consider only discrete-time controllers as potential VTs. For these, Simulink design rules force the designer to define a sampling rate for each (discrete-time) block. Currently we cover cases where the controller is designed in a single clock domain (i.e., all blocks use the same sampling period). In general case, when a controller contains several clock domains, each sub-domain is represented with its respective virtual tasks. Also, a set of dependencies between the tasks is extracted. Finally, to extract the worst-case execution time, a simple static analysis is performed using the execution time measurements for library defined words with respect to the specific platform.

III. EVM Architecture

We now describe the node-specific architecture which implements the mechanisms for the virtual machine on each node. The Common-EVM and Control-EVM description are scoped within Virtual Tasks (VTs) that are mapped at runtime by the Task Assignment procedure presented in the next section. This description is interpreted by the Virtual Component Interpreter running on each node. The EVM runtime system is built as a *supertask* on top of the nano-RK real-time operating system [42], allowing node-specific tasks to execute native and virtual tasks (i.e., those that are dynamically coupled with a node) to run within the EVM. The EVM block-level reference architecture is presented in Fig. 9(a). This allows the EVM to maintain node specific functionalities and be extensible to runtime task evocation of existing or new virtual tasks.

The interface between nano-RK and all VTs is realized using the Virtual Component Manager (VCM). The VCM maintains local resource reservations (CPU, network slots, memory, etc.) within nano-RK, the local state of the VTs and global mapping of VTs within the VC. The VCM is responsible for memory and network management for all VTs-to-physical nodes and presents a mapping between local and remote ports which is transparent to all local VTs. It includes a FORTH-like interpreter for generic and domain-specific runtime operations and a Fault/Failure Manager (FFM) for runtime fault-tolerant operation. The VCM is implemented in a modular form so the interpreter, FFM and other specialized modules may be swapped with extensions over time and for domain-specific applications.

A. EVM Extensions to the nano-RK RTOS

nano-RK is a fully preemptive RTOS with multi-hop networking support that runs on a variety of sensor network platforms (8-bit Atmel-AVR, 16-bit TI-MSP430, Crossbow motes, FireFly) [42]. nano-RK uses the RT-Link [43], a real-time link protocol. It supports fixed-priority preemptive scheduling to ensure that task deadlines are met, along with support for enforcement of CPU and network bandwidth reservations. nano-RK had been design as a fully static OS, configured at design time. Thus, to allow parametric and programmatic runtime code changes nano-RK was redesigned and extended with several new features:

• *Runtime Parametric Control:* Support for dynamic change of the sampling rates, runtime task and peripheral activation/deactivation and runtime modification of the task utilization was added. These facilities are exposed and executed via the Common-EVM programmer interface.

• *Runtime Programmatic Control:* As a part of the EVM design a procedure for dynamic task migration was implemented. This requires runtime schedulability analysis, capability checks to migrate a subset of the task data, instructions, required libraries and task control block. Based on the procedure presented in Sections IV and V, tasks may be activated or migrated between primary and backup nodes. Such facilities

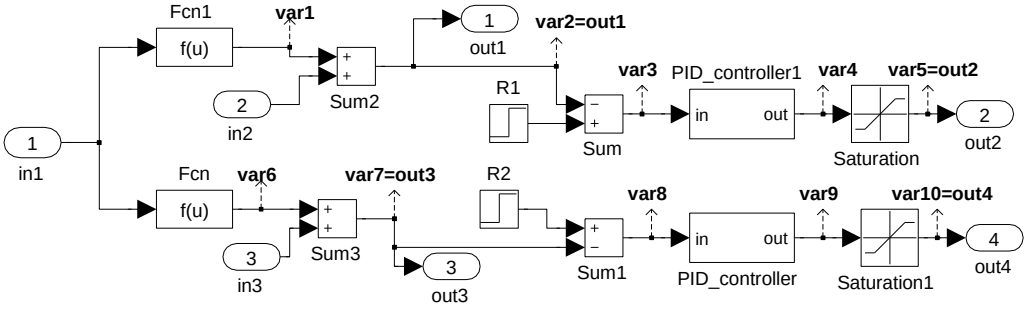


Fig. 7. Simulink model of an extended PID controller.

Step 1: Intermediate words/variables

: W1 in1 ? f var1 @ ;
: W2 var1 ? in2 sum out1 @ ;
: W3 out1 ? NEG R1 sum var3 @ ;
: W4 var3 ? Kp1 Ki1 Kd1 PID var4 @ ;
: W5 var4 ? thr SAT out2 @ ;
: W6 in1 ? f var6 @ ;
: W7 var6 ? in3 ? sum out3 @ ;
: W8 R2 out3 ? NEG sum var8 @ ;
: W9 var8 ? Kp2 Ki2 Kd2 PID var8 @ ;
: W10 var9 ? thr SAT out4 @ ;

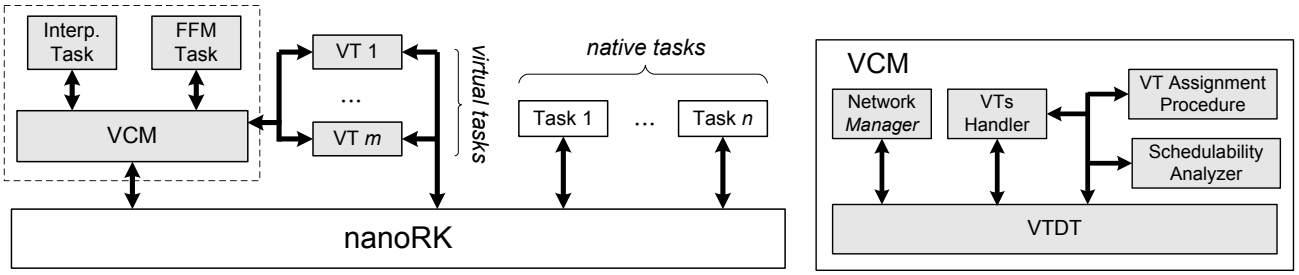
Step 2: Composition

: Vctrl W1 W2 W3 W4 W5 W6 W7 W8 W9 W10 out1 ? out2 ? out3 ? out4 ?

Step 3: Optimization

: W1 in1 ? f var1 @ ;
: W2 var1 ? in2 sum out1 @ ;
: W3 out1 ? NEG R1 sum Kp1 Ki1 Kd1 PID thr SAT out2 @ ;
: W7 var1 ? in3 ? sum out3 @ ;
: W8 R2 out3 ? NEG sum Kp2 Ki2 Kd2 PID thr SAT out4 @ ;
: Vctrl W1 W2 W3 W7 W8 out1 ? out2 ? out3 ? out4 ? ;

Fig. 8. EVM functional description extracted from Simulink model shown in Fig. 7



(a) EVM block-level reference architecture (b) Structure of the VCM

Fig. 9. EVM architecture with the Virtual Component Manager running as a supertask alongside native nano-RK tasks.

are triggered by the primary-backup policy implemented on top of the EVM architecture.

• *Dynamic Memory Management:* Both *Best-fit* and *First-fit* memory allocation methods are supported. In addition, a *Garbage Collector* (GC) has been designed to reclaim all memory segments owned by tasks that had been terminated. The GC is scheduled only when its execution does not influence execution of other tasks.

B. Virtual Component Interpreter

The Virtual Component Interpreter provides an interface to define and execute all VTs. Every VT is defined as a word within the VCM library. When a new VT description is received over the network, the VCM calls the interpreter that defines a new word using the description file of the task and existing VC libraries. After a VT is activated, each execution of the VT is realized as a scheduled activation of the interpreter with the VT's word provided as an input. To allow preemptivity of the tasks, each call of the interpreter uses a VT-specific stack and dedicated memory segments. In addition, during its execution, each VT is capable of dynamically allocating new memory blocks of fixed size (currently 128B) using the EVM's memory manager. Therefore, the interpreter is designed to use logical addresses in the form $(block\_index, address\_in\_block)$.

Each node maintains a local copy of standard Common-EVM and Control-EVM dictionaries. If a new word needs to be included in the existing library, the interpreter first checks the global word identifier and revision number to discard obsolete versions.

C. Virtual Tasks

Each VT is described using the Virtual Task's Description Table (VTDT), comprised of global and local descriptions of a VT. Copies of the table are stored on all members of the VC. While this requirement for consistency currently results in an issue of scalability, a large fraction of the higher-speed control in SCADA systems require networks with less than 20 nodes and is hence within the practical limits of the current approach. Each VT's global description has information about memory requirements, stack size and number of used fixed size memory blocks (128B). In addition to the above meta data, network requirements in terms of number of RT-Link transmit and receive slots are specified at design time.

The above descriptors are specified within the VCM's Task Control Block (TCB) for each task, which is an extension to

the native nano-RK TCB (for details see [40]).

D. Virtual Component Manager

The fundamental difference between the native nano-RK and the VCM is that the scope of nano-RK's activities is local, node-specific and defined completely at design time, while the scope of the VCM is the VC that may span multiple physical nodes. The VCM subcomponents are presented in Fig. 9(b). The current set of supported runtime functionalities is:

4.4.1. Virtual Task handling (controlled by the VT Handler):
4.4.1.1 VC state includes the mapping of VTs to physical nodes and quality of links between physical nodes. The VCM in each controller node within the VC maintains the VC state and periodically broadcasts it to keep consistency between all members of the VC. Currently, a centralized consensus protocol is used, while a distributed consensus protocol is needed to scale operations.
4.4.1.2 VT migration and activation that can be triggered as a result of a fault/failure procedure or by a request from either the VT or the VCM. As a part of a task migration, the task's VTDT is sent along with all memory blocks utilized by the task. If the VT is already defined on a Backup node (checked by exchange of hash values), only task parameters are exchanged. In addition, before migrating a VT to a particular node the Schedulability Analyzer performs network and CPU schedulability analysis for nodes that are potential candidates (details are provided in the next section). If the analysis shows that no node can execute the task correctly, an error message is returned. Finally, after a VT is defined, to activate the task the host node performs a local CPU and network schedulability analysis to ensure that the task will not adversely affect correct execution of previously defined VTs.
4.4.1.3 Control of tasks executed on other nodes: For all VTs in the *Backup* mode, the VT Handler shadows execution of the VT in the *Primary* mode. If a departure from the desired operation is observed (e.g., low battery level, decreased received packet signal strength), *Backup* nodes may be assigned to the *Primary* mode based on the policy.
4.4.1.4 VT Assignment: VT Assignment procedure is activated to assign execution of the VTs to specific nodes, when incremental and local re-assignment (described in Section V) fails. The procedure determines the best set of physical controller nodes to execute VTs given a snapshot of the current network conditions along with the initial communication and computation schedules for the nodes.

4.4.2. Network Management (performed by the Network Manager):
4.4.2.1 Transparent radio interface: Using the message header which contains information about message type, the VCM determines tasks that should be informed about the message arrival. Messages containing tasks and their parameter definitions are first processed by the VCM, before the VCM activates the interpreter.
4.4.2.2 Logical-to-physical address mapping: Communication between VTs is done via the VCM. Since a VT does not have information on which nodes other VTs are deployed, the VCM performs logical-to-physical address mapping. In cases when both tasks are on the same node, the VCM directly passes a message to the receiving task's buffer.

IV. Virtual Task Assignment

With the knowledge of the underlying EVM architecture, we now discuss the algorithm used for the VT Assignment procedure. The procedure determines the initial assignment of the VT's executions along with the communication and computation schedules. The criteria for triggering re-assignment calculation is described in Section V. We derived a general case problem formulation for the VT's assignment as a binary integer linear optimization problem which is then solved efficiently using well-known techniques (branch and bound) [44]. In addition, since standard link protocols for wireless factory automation, such as WirelessHART [45], recommend that only one physical node may transmit in each time slot, we were able to obtain an efficient reformulation of the relaxed assignment problem. In this case, each control loop (operating across the same physical set of controllers) can be considered separately, which considerably simplifies tasks assignments, as it allows a *compositional* system design.

A. General Formulation

To develop an assignment algorithm we considered a multi-hop control network that corresponds to our model of a VC. The network consists of $p \geq 1$ processes ($\mathbb{J} = \{1, ..., p\}$ denotes set of all processes) and a set of nodes (sensors, actuators and controllers), where all nodes have a radio transceiver along with memory and computing capabilities (see Fig. 10(a)). The nodes communicate using a TDMA based protocol (i.e., in a time-triggered manner) with frame size F_S. The network is described with a directed graph $G = (V, E)$ that represents radio connectivity in the network. Set $V = \{v_1, v_2, ..., v_m\}$ denotes a set of physical nodes in the network,[1] while $E = \{(v_i, v_j)|\ v_i \text{ and } v_j \text{ are connected}\}$ is a set of all links. In addition, each link e is described with its link quality $LQ(e)$. To extract a problem formulation it is necessary to enumerate all paths in the network which should be used for communication between a node and a sensor (or an actuator).[2,3] Thus, the l^{th} path between node v_i and sensor/actuator k is denoted as $\psi_{i,k}^{l}$.

The goal of the assignment procedure is to determine: (1) An assignment of the Virtual Tasks (i.e., Control Algorithms - CAs) to the set of nodes V, where each VT is assigned to one node in the *Primary* mode and to R nodes in the *Backup* mode. (2) A communication schedule that determines active links at each time slot. (3) A computational schedule that determines in which time slot each VT is executed. In addition, to define the problem as an optimization problem, the following assumptions were made:

[1] In the remainder of the paper, V will also denote the set that contains nodes' indexes $\{1, 2, ..., m\}$.

[2] A path is represented as a directed path connecting the sender with exactly one receiver.

[3] Including all paths could significantly increase complexity of the optimization problem. Therefore, the user might opt to enumerate only selected paths with best characteristics (e.g., a small number of hops, high packet delivery ratio).

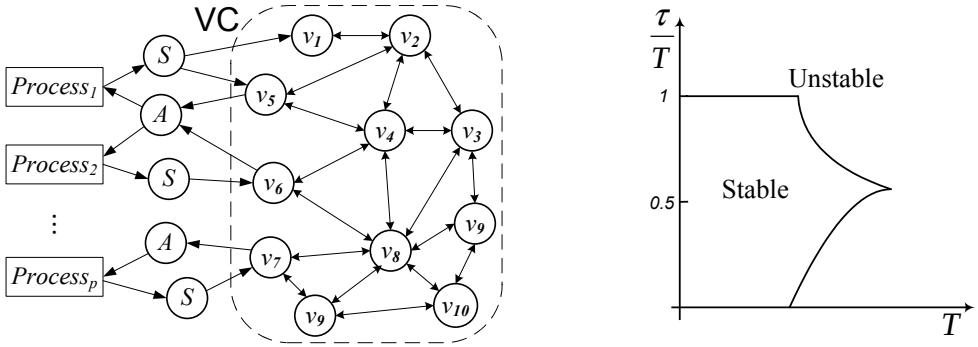


Fig. 10. (a) Reference model of a multi-hop wireless network used for control p physical plants (i.e., processes). The network consists of multiple sensors (S), actuators (A) and controllers (v_i's). The VC includes multiple physical controller nodes; (b) An example stability region for such a network. T is a controller sampling period, while τ is the network induced delay.

A.1 For each process j, the *Primary* and all *Backup* nodes assigned with the j^{th} virtual task are scheduled in the same time slot(s).
A.2 Virtual Tasks are mutually independent.
A.3 A process i (for all i) will remain stable if its sampling period is less than some predefined value T_i. Therefore, we require $F_S \leq min(T_1, T_2, ..., T_p)$.

The first assumption simplifies the problem formulation and allows for an easier schedulability analysis scheme. The second assumption is reasonable since a significant class of process controllers execute a large number of simple and independent control loops. As an avenue of future work, this assumption will be relaxed to consider dependencies between tasks. To justify the last assumption we use the approach described in [3]. For example, consider a closed-loop control of a plant modeled with continuous-time Linear-Time Invariant (LTI) dynamics:

$$\dot{x}(t) = Ax(t) + Bu(t),\ y(t) = Cx(t).$$

The controller employs a discrete-time state feedback control with $u(kT) = -Kx(kT)$, where T denotes the plant's sampling period. If network induced delay τ_k is less than one sampling period,[4] the control feedback has the following form:

$$u(t^+) = -Kx(kT),\ t \in [kT + \tau_k, (k+1)T + \tau_{k+1}).$$

Thus, $u(t)$ is a piecewise continuous function that changes values only at time instances $kT + \tau_k$. The EVM utilizes fully synchronous networks, which allows scheduling the actuators to apply new input values at the same time, after the messages were delivered to all of them. This guarantees the same delay for all plant's inputs at each sampling period ($\tau_k = \tau, \forall k$). Using the methods based on simulation, as in [3], the stability region can be determined with respect to sampling period T and the induced delay τ. The region is used to establish the maximal sampling period for which the system maintains stability if a network delay is less than the period ($\frac{\tau}{T} \leq 1$, an example is shown in Fig. 10(b)).

To formulate the problem, the following decision variables are used:

[4]A similar approach can be used even if the delay is longer than the sampling period.

- $2mp$ binary assignment variables, $x^{st}_{i,j} \in \{0,1\}$, where $i \in V, j \in \mathbb{J}, st \in \{a,b\}$ and

$$x^a_{i,j} = \begin{cases} 1, v_i \text{ is the } \textit{Primary} \text{ for } j^{th} \text{ VT} \\ 0, \text{ otherwise} \end{cases},$$
$$x^b_{i,j} = \begin{cases} 1, v_i \text{ is a } \textit{Backup} \text{ for } j^{th} \text{ VT} \\ 0, \text{ otherwise} \end{cases}$$

- Routing binary variables $y^l_{i,k} \in \{0,1\}$, where:
$$y^l_{i,k} = \begin{cases} 1, l^{th} \text{ path between node } v_i \\ \text{ and sensor/actuator } k^{th} \text{ is used} \\ 0, \text{ otherwise} \end{cases}$$
- Communication schedule binary variables $\eta^{l,n}_{i,k} \in \{0,1\}$, where $n \in \{1, ...F_s\}$ and:
$$\eta^{l,n}_{i,k} = \begin{cases} 1, l^{th} \text{ path between node } v_i \\ \text{ and sensor/actuator } k \text{ is active in } n^{th} \text{ slot} \\ 0, \text{ otherwise} \end{cases}$$
- Computation schedule binary variables $\mu^n_i \in \{0,1\}$, where $n \in \{1, ...Fs\}$ and: $\mu^n_j = \begin{cases} 1, j^{th} \text{ VT is scheduled for execution in } n^{th} \text{ time slot} \\ 0, \text{ otherwise} \end{cases}$

Our goal is to describe the assignment problem in the form:

$$\min f(\mathbf{x}, \mathbf{y}, \eta, \mu),\ \text{subject to} \quad \mathbf{x}, \mathbf{y}, \eta, \mu \in \mathbb{S}_{\mathbb{C}}$$

where vectors $\mathbf{x}$, $\mathbf{y}$, η, μ contain the aforementioned decision variables and $\mathbb{S}_{\mathbb{C}}$ describes a set that satisfies all constraints, ensuring desired system's behavior. The constraints take into account the requirements for control problem along with dependencies between communication and computation schedules. In the remaining of this section the imposed set of constraints is described.

*1) **Assignment of the Control Algorithms**:* Each VT has to be assigned to exactly one node in the *Primary* mode and R additional *Backup* nodes (different from the *Primary* node for the CA). These constraints are described as:

$$\sum_{i=1}^{m} x^a_{i,j} = 1,$$
$$\sum_{i=1}^{m} x^b_{i,j} = R, \quad \text{and}$$
$$x^a_{i,j} + x^b_{i,j} \leq 1, \qquad \forall j \in \mathbb{J}, \forall i \in V.$$

*2) **Requirements for robust design**:* Additional sets of constraints are introduced to improve performance of the closed-loop system. *Link reliability* constraints require that only links with quality above a given threshold are considered, which reduces complexity of the problem formulation. Logical pruning of graph G results in a graph $G_T = (V, E_T)$, where $E_T = \{(v_i, v_j) \in E | LQ(v_i, v_j) \geq THR)\}$.

The *Routing constraints* describe a means to increase system robustness to the link failures with the use of different paths for data routing. For example, WirelessHART recommends that each node can use at least two separate paths to route data [14]. Thus, we require that the *Primary* node for each VT uses two different paths to deliver information to all actuators related to the process' control. In addition, the *Primary* and all *Backup* nodes have to be connected with all sensors related to the VT.[5] Denoting as A_j and S_j the sets of actuators and sensors respectively, related to the j^{th} process, these constraints are described as:

$$\sum_{\forall l} y^l_{i,k_a} = 2x^a_{i,j}, \ \sum_{\forall l} y^l_{i,k_s} = x^a_{i,j} + x^b_{i,j}, \quad \forall j \in \mathbb{J}, k_a \in A_j, k_s \in S_j, \forall i \in V.$$

Finally, a set of *Monitoring constraints* is imposed, where all *Backup* nodes monitor the execution of a VT on the *Primary* node. Thus, to alleviate the system design and VT migration when the *Primary* node fails, constraints are enforced that all R *Backup* nodes have to be 1-hop neighbors of the *Primary* node. Denoting as N_i set of all neighbors of node v_i, these constraints are described as: $\sum_{k \in N_i} x^b_{k,j} \geq R \cdot x^a_{i,j}, \ \forall j \in \mathbb{J}, \forall i \in V$.

*3) **Computation schedule constraints**:* From assumptions A.3 and A.1, we require that computations of each VT on the *Primary* and *Backup* nodes have to be scheduled exactly once in a frame. This implies that all VTs have the same sampling rate and could result in a more frequent computation of a VT. In most automation systems the increase of the sampling rate can not endanger the closed-loop system stability. On the contrary, it can increase the performance of the implemented controller if the optimal discrete-time controller is used [46].[6] Thus, the constraints are expressed as: $\sum_{n=1}^{F_S} \mu^n_j = 1, \ \forall j \in \mathbb{J}$.[7]

*4) **Communication schedule constraints**:* From assumption A.3, closed-loop system stability is guaranteed if the end-to-end communication delay (i.e., delay from the sensors to the assigned controller and from the controller to the actuators) along with the time needed for the controllers' computation is less than F_S. Thus, the first requirements for the communication schedule is that only used paths are scheduled and that the number of slots assigned to the used path is exactly equal to the path's length (i.e., number of hops on the path):

$$\eta^{l,n}_{i,k} \leq y^l_{i,k}, \ \forall n, 1 \leq n \leq F_S,$$

$$\sum_{n=1}^{F_S} \eta^{l,n}_{i,k} = y^l_{i,k} \cdot d(\psi^l_{i,k}), \ \forall i, k \in V, \forall l. \tag{1}$$

Additionally, the schedule has to be collision free (i.e., two interfering nodes cannot transmit in the same time slot). To express these constraints, for each path $\psi^l_{i,k}$ where k is a sensor, all links are enumerated in increasing order starting from the link with origin at sensor k and ending with the link with the destination at node i. Similarly, for each path $\psi^l_{i,k}$ where k is an actuator, enumeration starts at node i and ends at actuator k. This is used to create the interference links table for each pair of paths $(\psi^{l_1}_{i_1,k_1}, \psi^{l_2}_{i_2,k_2})$. An element (n_1, n_2) is a member of the $(\psi^{l_1}_{i_1,k_1}, \psi^{l_2}_{i_2,k_2})$ interference table (IT) if transmissions over the $n_1{}^{st}$ link of the path $\psi^{l_1}_{i_1,k_1}$ interferes with transmissions over the $n_2{}^{nd}$ link of the path $\psi^{l_2}_{i_2,k_2}$. Constraints for interference-free schedule can be described as: For all $n, \ 1 \leq n \leq F_s$, $\forall i_1, i_2 \in V$,

$$|\sum_{n_0=1}^{n} \eta^{l_1,n_0}_{i_1,k_1} - n_1| +$$

$$|\sum_{n_0=1}^{n} \eta^{l_2,n_0}_{i_2,k_2} - n_2| \geq 1, \ \forall k_1, k_2 \in S \cup A, \ (n_1, n_2) \in IT(\psi^{l_1}_{i_1,k_1}, \psi^{l_2}_{i_2,k_2}) \tag{2}$$

*5) **Dependencies between the schedules**:* Communication and computation schedules must be aligned, meaning that measured data (i.e., data from sensors) is routed to the controller prior to the VT's activation. Also, data designated to the actuators are forwarded after the computation of the VT: For all $n, \ 1 \leq n \leq F_s$

$$\eta^{l,n}_{i,k_s} \leq (1 - \sum_{n_0=1}^{n} \mu^{n_0}_j),$$

$$\eta^{l,n}_{i,k_a} \leq (\sum_{n_0=1}^{n-1} \mu^{n_0}_j), \forall j \in \mathbb{J}, \forall i \in V, \ \forall l, k_s \in S_j, k_a \in A_j, \tag{3}$$

*6) **Objective function**:* The goal of the assignment procedure is to minimize the aggregate number of used links while maximizing the aggregate link quality. In addition, we want to maximize the use of disjoint routing. Thus, a cost for sharing links is introduced, both in paths from sensors to controllers and from the *Primary* controller to the actuators. As can be seen, the objective function (i.e., cost) does not depend on utilized scheduling. Therefore, it is defined as a weighted sum $f(\mathbf{x}, \mathbf{y}) = w_1 f_{LN} + w_2 f_{LQ} + w_3 f_{SL}$, where weights w_1, w_2 and w_3 are used to emphasize impacts of the following cost functions:

1) Aggregate number of used links: $f_{LN}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{p} (\sum_{l,k,i} y^l_{i,k} \cdot d(\psi^l_{i,k}))_j$, where $d(\psi^l_{i,k})$ is a distance (i.e., length, number of hops) of path $\psi^l_{i,k}$.
2) Negative aggregate link quality: $f_{LQ}(\mathbf{x}, \mathbf{y}) = -\sum_{j=1}^{p} (\sum_{l,k,i} y^l_{i,k} \cdot LQ(\psi^l_{i,k}))_j$
3) Cost of the shared links:

[5]It is worth noting here that a different routing policy could be used. However, even if that is the case these constraints could be expressed in a similar way.

[6]Future extensions of this work will allow CAs to have different sampling periods.

[7]In the constraint formulation we assume that each VT can be executed in one time slot. In general this might not be the case. However, it would just require a formulation change where instead of 1, execution time necessary for execution of the j^{th} VT (i.e., e_j) is placed. Even more general, if the network contains nodes with different computational power, the previous term should be expressed as $\max_{i=1}^{m}(x^a_{i,j} \cdot e^a_j + x^b_{i,j} \cdot e^b_j)$. To simplify the notation, we decided to use the aforementioned assumption.

$$f_{SL}(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{p} \sum_{\substack{1\le i\le t\le m \\ l_i,l_t,\ k\in S_j\cup A_j}} y_{i,k}^{l_i}\cdot y_{t,k}^{l_t}\cdot SH(\psi_{i,k}^{l_i},\psi_{t,k}^{l_t}),$$

where $SH(\psi_{i,k}^{l_i},\psi_{t,k}^{l_t})$ is a number of links shared between paths $\psi_{i,k}^{l_i}$ and $\psi_{t,k}^{l_t}$.

Therefore, the assignment problem can be formulated as a binary integer programming optimization problem and solved using some of the well-known techniques (branch and bound) [44]. One caveat is in order. Since the problem formulation has a large number of decision variables, even for a small network it can be computationally expensive to solve the problem. Thus, we translated the problem into the satisfiability problem, by transforming each constraint into conjunctive normal form (CNF) (for details see [40]). The satisfiability problem is then solved using zChaff [47], a very efficient satisfiability solver. This allows us to solve the previous problem in real-time even for large scale networks.

B. Problem Relaxation

When only one node in the VC can transmit in each time slot, the number of slots needed to send a message from node v_1 to node v_2 is equal to the distance between the nodes. This is used for the relaxed problem formulation, as it eliminates the need to include communication and computation decision variables used in the general formulation and, therefore, significantly reduces complexity of the optimization problem. In addition, the collision-free communication requirement, which is the most complex set of constraints from the general formulation, becomes redundant. The requirement is inherently fulfilled with the policy that allows a single transmission per time slot for the whole VC.

As the first step for the problem formulation, two maximum node-disjoint paths r_{i,a_c}^1 r_{i,a_c}^2 are determined for each node v_i and each actuator a_c. The existence of two node disjoint paths from a node to all sensors and actuators can be checked using Menger's theorem [48] (for details see [40]). When two node-disjoint paths exist for the node, using a polynomial time algorithm (MIN-SUM 2-paths [49]) paths r_{i,a_c}^1 r_{i,a_c}^2 with the minimal total length can be determined. Otherwise, path r_{i,a_c}^1 is computed in polynomial time as the shortest path to the actuator. Path r_{i,a_c}^2 is calculated as the shortest path to the actuator after removing nodes from path r_{i,a_c}^1, while preserving connectivity. Using a similar approach, for each node v_i and all its neighbors $v_{i_1},...,v_{i_{n_i}}$ (n_i is a degree of node v_i), a set of n_i+1 paths is created between each sensor s and the nodes. We denote these distances as $(d_{i,s}, d_{i_1,s},...,d_{i_{n_i},s})$.

To extract the relaxed problem's formulation we used only $2mp$ binary assignment variables $x_{i,j}^a$ and $x_{i,j}^b$ defined as in the general problem formulation. This allows us to formulate the problem as follows:

$$\min w_1\cdot f_{LN}(\mathbf{x}) + w_2\cdot f_{LQ}(\mathbf{x}),$$

with the respect to $\mathbf{x}\in\{0,1\}^{2mp}$, which contains the aforementioned decision variables. The feasible set is described with the following set of constrains:

$$\sum_{i=1}^{m} x_{i,j}^a = 1, \qquad \sum_{i=1}^{m} x_{i,j}^b = R, \quad x_{i,j}^a + x_{i,j}^b \le 1,$$

$$\sum_{k\in N_i} x_{k,j}^b \ge R\cdot x_{i,j}^a,\ \forall j\in\mathbb{J},\ \forall i\in V,$$

$$\sum_{\substack{i\in V \\ j\in\{1,...,p\}}} \{\sum_{s\in S_j}(x_{i,j}^a\cdot d_{i,s} + \sum_{k\in N_i} x_{k,j}^b\cdot x_{i,j}^a d_{i_k,s}) +$$

$$\sum_{a\in A_j} x_{i,j}^a\cdot\big(d(r_{i,a}^1)+d(r_{i,a}^2)\big)\} + 1 \le F_s$$

The last constraint requires that all communication is done within one frame and therefore, meets the timing requirements necessary for the system's stability. This constraint is the only one that depends on the number of VTs and utilized data routing. Thus, a suboptimal, yet feasible solution can be obtained (if and only if a feasible solution exists) using *compositional analysis*. In this case each control loop, operating across the same physical set of controllers is considered separately. Optimizing only for the cost function f_{LN} and for each loop separately provides an optimal assignment for each loop that uses the minimal number of communication slots (details see in [40]). Note that if $w_1/w_2 >> 1$, the approach provides the optimal solution for the relaxed assignment problem in general. Also, for a sufficiently high link quality threshold (while deriving graph G_T) the impact of function f_{LQ} is reduced. This enables use of the compositional design, which significantly simplifies the system analysis and schedule extraction. Since the EVM is focused on networks with less than 20 nodes, we are able to run the optimization algorithm on all nodes in a VC, as the *VT Assignment Procedure*.

V. EVM Runtime Operation: Virtual Task Execution

Given the task migration mechanisms and the algorithms to (re)assign tasks, we now describe the relationship between primary and backup nodes for planned and unplanned scenarios. More specifically, we consider the criterion for triggering task migration and the node and network schedulability analysis that must be conducted prior to migration. To completely address the issues in wireless networked control systems, we must consider (a) the mechanisms for runtime adaptation, (b) the algorithms for runtime task (re)assignment to physical nodes and (c) the fault tolerance policy. In this paper we focus on the first two aspects and apply them to simple network models with non-Byzantine single node and link failures. As the fault tolerance policy is dependent on the control application and fault/failure model is a function of the specific environment, we do not consider specific policies here. We aim to address Byzantine errors such as software errors in future work.

A. Adaptation to Planned and Unplanned Network Changes

Planned adjustments occur in situations when a *Primary* node is informed of changes in VC state (e.g., when a node detects that its battery level is below some threshold). To determine a *Backup* node to migrate its task, the *Primary* node has to execute computation and communication schedulability analyses in $k = 1$-hop neighborhood and select a *Backup* node that maximizes the communication slack value while

maintaining computation schedulability.
For unplanned changes caused by potential failures we consider the following cases:

- The *Primary* nodes dies: Computation and communication schedulability analysis in $k = 1$-hop neighborhood is initiated. Since state data of the *Primary* node is maintained at *Backup* nodes, a new *Primary* node continues VT execution.
- A *Backup* node dies: The *Primary* node detects the *Backup* has died and selects a new *Backup* from one of its neighbors.
- A forwarding node dies or a link's quality goes below some criterion: The detection of a forwarding node failure is performed by its predecessor/successor on the routing path. Again, a communication schedulability analysis is performed (only for the affected sensor and actuator) to determine a new routing scheme.

To decrease response time for the schedulability analyses, each node uses its idle computation time to calculate in advance the optimal reaction to a set of potential failures. Besides decreasing the response time, this approach enables triggering the execution of the *Assignment Procedure* if it is determined that for some failures there is no adjustments that can meet all of the constraints. Also, if the procedure can not derive a feasible assignment, an alarm is raised notifying system operators to add more nodes in the network to prevent a potential failure.

B. Communication Schedulability Analysis

The goal of communication schedulability is to determine whether we can *incrementally* reassign the available communication slots due to the change in the task assignment, without executing a global reassignment of communication slots. To accomplish this we determine the current communication slack and evaluate if it is sufficient for the incremental slot reassignment. When a VT is to be migrated from a node v_i to a node v_j, we define sets S_{VT} and A_{VT} of all sensors and actuators respectively, related to the VT. Also, for each $s \in S_{VT}$ we denote as $v^k_{i,s}$ a node that is k-hops away from node v_i on the route from sensor s to node v_i. Similarly, for each $a \in A_{VT}$, $v^k_{i,a}$ denotes a node that is k-hops away from node v_i on the route to the actuator a. In addition, we denote as N^i_u the number of unused time slots in the time interval between the first slot in which *all* nodes $v^k_{i,s}$ were suppose to receive values from sensors in S_{VT} and a first slot in the frame in which *at least one* node $v^k_{i,a}$ was scheduled to receive information from the node v_i. The parameter k determines the set of candidate backup nodes to which the task may be reassigned.

More specifically, the goal of communication schedulability is to determine whether we can reassign (with the respect to the current communication schedule) the available communication slots and slots used to send data in the k-hop neighborhood of a node v_i. The re-assignment should re-route all sensor and actuator data from these nodes to node v_j. A new feasible communication schedule can be generated if $\Delta \geq 0$, where Δ denotes communication slack value defined as:

$$\Delta = \sum_{s \in S_{VT}} d(v_i, v^k_{i,s}) + \sum_{a \in A_{VT}} d(v_i, v^k_{i,a}) + N^i_u - \sum_{s \in S_{VT}} d(v_j, v^k_{j,s}) - \sum_{a \in A_{VT}} d(v_j, v^k_{j,a}),$$

where $d(v_p, v_q)$ is the distance between nodes v_p and v_q. If more than one task is migrated from a node, similar analysis is performed with the previous equation adjusted to contain sums of all sensors and actuators related to the tasks. In addition, if tasks should be migrated from node v_i to separate nodes, the schedulability test is performed on a pairwise basis.

C. Computation Schedulability Analysis

For the computation schedulability analysis we use standard real-time response analysis [50] and the mode-change protocol, presented in [51] and [52], adapted for the EVM. Consider a node v_i that executes a task set $\mathbb{T} = \{T_{i_1}, ..., T_{i_m}, VT_{i_1}, ..., VT_{i_n}\}$, where tasks T_{i_j} are local, node specific tasks, while tasks VT_{i_j} are VTs assigned to the node (in descending order of priority). We define a set $HP\_VT(T)$ as a set of all VTs with higher priority than local task T and, similarly, a set $HP\_T(VT)$ as a set of all node-specific tasks, with higher priority than task VT. To allow an assignment of a new VT, a schedulability analysis is performed where both active and inactive tasks are considered as active. Although this approach is conservative, it eliminates the need for repeated schedulability analysis prior to tasks activation. Each node-specific task is denoted as $T_j = (p_{T_j}, e_{T_j})$ and each VT as $VT_j = (p_{VT_j}, e_{VT_j}, \phi_{VT_j}, d_{VT_j})$ (period, execution time, offset and deadline respectively). Schedulability of a new task set is performed by checking only the schedulability of each task with a lower priority than the new virtual task VT_k, using its time-demand function $w(t)$ [50].

As mentioned in Section IV, we currently consider the case where all VTs have the same execution period. Since execution of a VT is triggered by the reception of sensed signals and must be finished before its scheduled communication to actuators, its deadline is significantly lower than its period. Thus, from a VT's activation till its deadline, all other VTs can be active at most once, so for a task $VT_i,\ i \geq k$:

$$w_{VT_i}(t) = e_{VT_i} + \sum_{j \in HP\_T(VT_i)} \left\lceil \frac{t}{t_{T_j}} \right\rceil \cdot e_{T_j} + \sum_{j=1}^{i-1} e_{VT_j}$$

The equation is too conservative as it assumes that all VTs can be activated at the same time. However, VTs are activated when a last radio message containing necessary data is received. In addition, since all VT's periods are multiples of TDMA slot duration, when a communication schedule is known, all possible offset combinations of a task activation can be easily calculated. Therefore, for a task VT_i, released at time t_i, for all possible combinations of release times t_j of VTs with higher priority, the time-demand function for $t \geq t_i$ is defined as:

$$w^{(t_0,t_1,...,t_{i-1})}_{VT_i}(t) = e_{VT_i} + \sum_{k \in HP\_T(VT_i)} \left\lceil \frac{t}{t_{T_k}} \right\rceil \cdot e_{T_k}$$

$$+ \sum_{\substack{j=1,\\ t_j \le t \le t_i+d_i}}^{i-1} \min(e_{VT_j}, t-t_j) + \sum_{\substack{j=1,\\ t_i \in [t_j, t_j+d_j]}}^{i-1} \min(e_{VT_j}, t_j+d_j-t_i)$$

Here the second term corresponds to the execution of all higher-priority native tasks; the third term corresponds to the demand from higher-priority VTs which are activated after the i^{th} task's activation, but before its deadline. Finally, the last term describes the demand of the higher priority VTs when the i^{th} task is activated between the higher priority tasks' activation and deadline. For schedulability we are interested in time instances where $w_{VT_i}^{(t_0,t_1,...,t_{i-1})}(t) = t$. These points can be obtained using efficient recurrence procedure described in [50]. The task is schedulable, if for all combinations of activation times, the solution of recurrence procedure is less than the task's deadline (d_{VT_i}).

Although the previous equation seems complicated, in the case when all VTs are executed once per frame there is only one combination of release times $(t_0, t_1, ..., t_{i-1})$ (i.e., only one set of task offsets as the TDMA schedule is fixed). Even in general case there is no need to cover a large number of possible combinations since for most control systems, all loops usually have the same sampling period or all sampling periods are integer multiples of one of the periods.

A similar approach is used for schedulability analysis of a node-specific task T_i.

VI. EVM Implementation

To evaluate the EVM's performance in a real setting with multiple coordinated controller operations, we used a factory simulation module shown in Fig. 11(a). The FischerTechnik model factory consists of 22 sensors and actuators (Fig. 11(b)) that are to be controlled in a coordinated and timely manner. A block of wood is passed through a conveyor, pushed by a rammer onto a turn table and operated upon by up to three milling/cutting/pneumatic machines. The factory module was initially controlled by wired programmable logic controllers (PLCs). We converted it to use wireless control with FireFly embedded wireless nodes [53] controlling all sensors and actuators via a set of electrical relays. FireFly is a low-power platform based on Atmel ATmega1281 8-bit microcontroller with 8KB of RAM and 128KB of ROM along with a Chipcon CC2420 IEEE 802.15.4 standard-compliant radio transceiver. FireFly nodes support tight global hardware-based time synchronization for real-time TDMA-based communication with the RT-Link protocol [43]. The EVM also works on TI MSP430 architectures.

In our experiments we demonstrate:

1. On-line capacity expansion when a node joins the VC.
2. Redistribution of VTs when adding/removing nodes.
3. Planned VT migration triggered by the user.
4. Unplanned VT migration due to a node or a communication link failure.
5. Multiple coordinated work-flows.

We tested the setup with a batch of 10 input blocks consisting of 3 different types which require different processing procedure. This is an example of the logical benefits of the EVM as it enables a more agile form of manufacturing. Details about the experiments, along with the videos can be seen in [54].

VII. EVM Case Study

As this is an early effort to describe the main functionalities of the EVM, we limit our case study to a simple simulated control network. We simulated the performance of the EVM for the case when a wireless networks is used for control in the Shell Problem, a well-known problem from process control theory concerning control of a heavy oil fractionator [55], [56]. The controlled variables (outputs) are differences of the top product end point ($Y1$) and the bottom reflux temperature ($Y2$) from predefined (reference) values. Fig. 12(a) presents a Simulink framework used for the simulation, where *Controller* (shown in Fig. 7) and *Plant* are similar to models from [55]. The major difference is that *Plant*'s dynamics was sped up to be able to test system's performance.

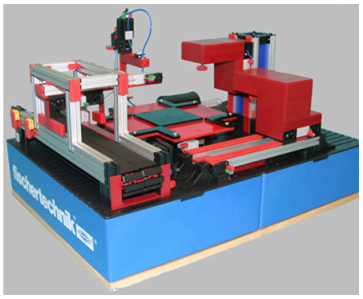
The functional description of the VT, shown in Fig. 8, is derived as described in Section II. Since all continuous outputs of the *Plant* have to be sampled before processed with a discrete-time controller, the sampling period defined in *SampleAndHold* blocks in the Simulink model is used to extract the period of each VT.

Fig. 12(b) presents the initial topology of the VC along with the *Primary* and the *Backup* node. To be able to address the effects of message drops, we assigned each link in the network a Packet Delivery Ratio (PDR) that is less than 1 (i.e., 100%). A TDMA protocol with 32 slots per frame is used for communication between nodes, where 24 slots were used for transfer of data related to the control problem, while 8 remaining slots per frame were used to exchange messages about VC's status. The system response to a series of different step inputs (a new one was set to arrive every 60s) for the initial topology is presented in Fig. 12(d). Also, a scenario was simulated where the initial topology changes after some of the links fail (as shown in Fig. 12(c)). Fig. 12(e) presents the response of the system without the EVM, where only re-routing algorithms are used without changing positions of the *Primary* and *Backup* nodes. This results in a system response that rapidly deteriorates. The system becomes unstable, due to increase in end-to-end communication time from all sensors to the *Primary* node to all actuators.

Fig. 12(f) shows how the EVM's adaptation to unplanned changes in link quality keeps the system's response similar to that in the initial topology. For the case presented in Fig. 12(f), we simulated the system when at time $t = 60s$ the network topology changes to that presented in Fig. 12(c). Due to the task re-assignment, one execution of the control algorithm is omitted, but as it can be seen, without significant influence to the overall system performance. This was expected since, from the perspective of the *Plant*, this case is equivalent to packet drops, which already occurs due to the fact that PDR is less than 100%.

VIII. Limitations of the EVM Approach

• **Complexity of Consensus:** The complexity of reaching consensus forces our current implementation to maintain a



(a) Work-cell module

Component	Sensors			Actuators	
	Proximity Home	Proximity Out	IR Position	Non-reversing Motor	Reversing Motor
Conveyor belt			X		
Ram	X	X		X	X
Horizontal slotting mill • Horizontal direction • Vertical direction	 X X	 X X		X	X
Vertical Gluing Machine	X	X	X	X	
Table Rotation				X	X
Vertical boring machine	X	X			X

(b) Module components

Fig. 11. FischerTechnik factory module with 22 sensors and actuators

substantial amount of state information with a relatively high update frequency. This limits the scalability of the current EVM approach to small networks with $\leq$20 nodes. While this is 'good enough' for a large number of small embedded wireless control applications such as natural gas processing with slowly varying operating parameters, it is essential to explore distributed algorithms to maintain state across the virtual component.

• **Centralized Approach:** The centralized algorithm has been used to solve the assignment problem. This limitation motivated us to explore a distributed solution for incremental strategies for control-loop implementation. Using the entire node population within a virtual component as a distributed controller would remove the need for the virtual task's assignment procedure.

So far, we presented an initial stab at a problem that unravels series of difficulties at the heart of networked Cyber-Physical Systems. We have investigated several fundamental challenges with the use of wireless networks for time-critical closed-loop control problems. Our approach was to build the networking infrastructure to maintain state across physical node boundaries, allowing tasks to be decoupled from the underlying unreliable physical substrate. We present a modular architecture used for control applications in wireless sensor/actuator/controller networks that allows component integration and system reconfiguration at runtime, without any negative effects on the execution of already assigned functionalities. The EVM enables a simple transition from the controller design in widely used simulation tools to the actual, physical 'plug-and-play' deployment for wireless networks.

To overcome the shortcomings of EVM, we now present the Wireless Control Network (WCN) approach for distributed in-network control.

IX. Part II: Wireless Control Networks

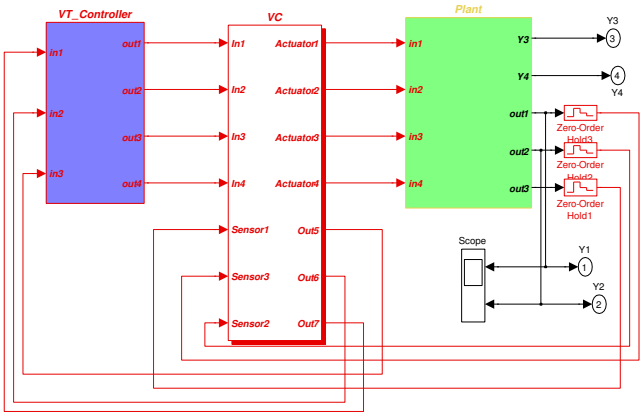
We consider the problem of stabilizing a plant with a multi-hop network of resource constrained wireless nodes. We present a distributed scheme used for control over a network of wireless nodes. As opposed to traditional networked control schemes where the nodes simply route information to and from a dedicated controller (perhaps performing some encoding along the way), our approach, Wireless Control Network (WCN), treats the network itself as the controller. In other words, the computation of the control law is done in a fully distributed way inside the network. In the WCN approach, at each time-step, each node updates its internal state to be a linear combination of the states of the nodes in its neighborhood. This causes the entire network to behave as a linear dynamical system, with sparsity constraints imposed by the network topology. We demonstrate that with observer style updates, the WCN's robustness to link failures is substantially improved. Furthermore, we show how to design a WCN that can maintain stability even in cases of node failures. We also address the problem of WCN synthesis with guaranteed optimal performance of the plant, with respect to standard cost functions. We extend the synthesis procedure to deal with continuous-time plants and demonstrate how the WCN can be used on a practical, industrial application, using a process-in-the-loop setup with real hardware.

Given the fundamental unreliability of wireless communication, the WCN method handles topological constraints while maintaining mean square stability for packet drop rates ***up to 20%*** for a specific network topology and plant. This bridges the gap between the basic WCN and the theoretical upper bound of robustness to packet drops [21]. We also present a method to synthesize a WCN robust to a certain level of node failures, before we extended the synthesis procedures to allow for the use of the WCN for control of continuous-time plants. Finally, we illustrate the use of the WCN on a real-world industrial case study, for control of a distillation column.

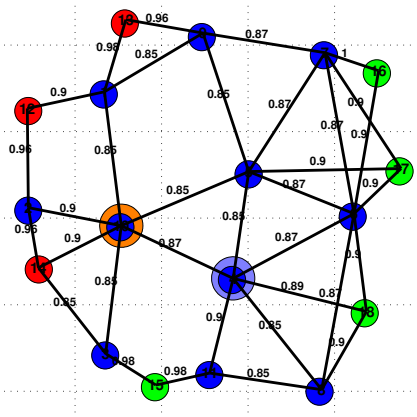
While in the past efforts, we consider scenarios where the network topology is already set, in recent efforts [23] we have investigated a dual problem, "how to synthesize the network so that a stable WCN configuration exists?" The topological conditions from [23], along with the results from [20] provide the essential building blocks for an integrated decentralized wireless control network design framework. Early experiments in an industrial process control case study of a distillation column in a process-in-the-loop test-bed to demonstrate optimal control of continuous-time physical processes which maintain system stability under the presence of node and link failures.

A. An Intuitive Overview of the WCN

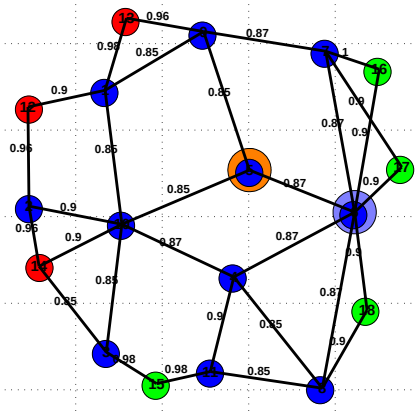
The role of feedback control is to apply inputs to the plant (based on observed outputs) in order to elicit the desired behavior. The exact mapping between observed behavior and applied inputs depends on a mathematical model of the plant, describing how inputs affect the system (over time). Here, we start with a common discrete-time, linear time-invariant model



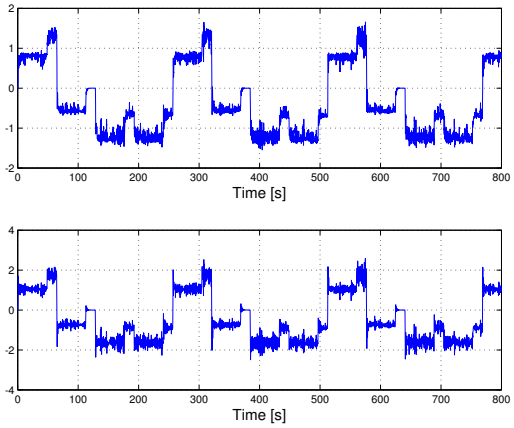
(a) Simulink model for Shell problem



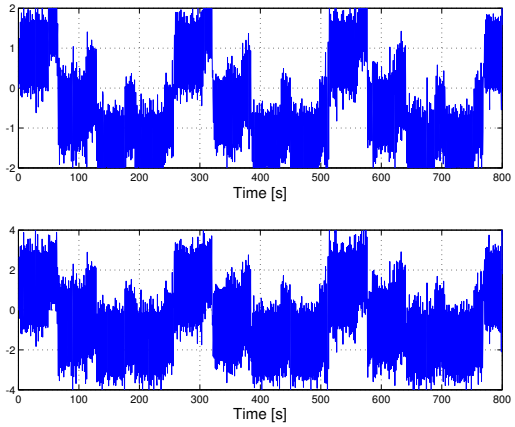
(b) Initial network topology



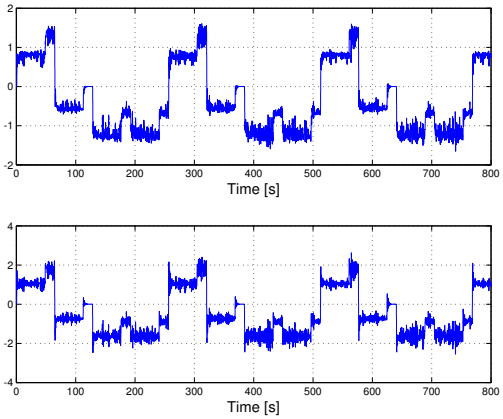
(c) Topology after link failures



(d) System response for initial configuration, showing outputs Y1 (top) and Y2 (bottom)



(e) System response when EVM is not used (when only re-routing is used), Y1 (top) and Y2 (bottom)



(f) System response when EVM adapts to changes in network conditions, Y1 (top) and Y2 (bottom)

Fig. 12. Simulation of EVM behavior when used for 'Shell problem' control; Nodes: green - actuators, red - sensors, blue circle - the *Primary* node, orange circle - the *Backup* node.

of the form:[8]

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{B_w}\mathbf{u}_w[k] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k], \end{aligned} \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^p$ denote the plant's state and output, $\mathbf{u} \in \mathbb{R}^m$ is the plant's (controllable) input, and $\mathbf{u}_w \in \mathbb{R}^{m_w}$ is the disturbance input.[9] Accordingly, the matrices $\mathbf{A}, \mathbf{B}, \mathbf{B}_w, \mathbf{C}$ have suitable dimensions.

Standard dynamical feedback controllers collect the observed plant outputs $\mathbf{y}[k]$ and generate the control input $\mathbf{u}[k]$ as the output of a linear system of the form:

$$\begin{aligned} \mathbf{x_c}[k+1] &= \mathbf{A_c}\mathbf{x}_c[k] + \mathbf{B_c}\mathbf{y}[k] \\ \mathbf{u}[k] &= \mathbf{C_c}\mathbf{x}_c[k] + \mathbf{D_c}\mathbf{y}[k]. \end{aligned} \tag{5}$$

The vector $\mathbf{x}_c[k]$ denotes the state of the controller, and the matrices $\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c$ and $\mathbf{D}_c$ are designed using standard tools

[8] In Section XIII we will show how continuous-time plants can be cast in this framework using discretization.

[9] We do not have any control over the disturbances.

from control theory, to ensure that the control inputs are stabilizing. Depending on the control method used, the state of the controller can often be as large as the state of the system itself.

In the above traditional approach to controller design, a wireless network would simply be placed between the controller and the plant to carry information back and forth. The goal of our work is to derive a truly networked and fully distributed control scheme, where the collective computation and communication capabilities of the wireless nodes are fully leveraged to compute the control inputs *in-network*. Intuitively, we propose a simple scheme for each node in the network to follow (using only information from its nearest neighbors at each time-step) that results in the desired network behavior. Essentially, we would like each wireless node to act as a small dynamical controller, with two main differences: (i) the state of the controller at each node will be constrained to be rather small (in order to account for resource and computational constraints), and (ii) in its updates, each node only uses the states of its nearest neighbors (which could include the plant's outputs, if the node is within transmission range of the outputs). Note that the latter condition precludes the need to route information from the plant to each controller in order for it to perform its update. In the rest of this section, we will make these conditions more mathematically precise.

B. Model of the Wireless Control Network

To model the WCN we consider the basic WCN setup from Fig. 1(c), where the plant is to be controlled using a multi-hop, fully synchronized wireless network with N nodes. In this paper, we extend the proposed scheme to allow for the design of a WCN that applies inputs in an 'optimal' manner (according to a cost function that we will define later). The plant model is given by (4), where the output vector $\mathbf{y}[k]$ contains the plant's output measurements provided by the sensors $s_1, \ldots, s_p$, while the input vector $\mathbf{u}[k]$ corresponds to the signals applied to the plant by actuators $a_1, \ldots, a_m$. The wireless network is described by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the radio connectivity (communication topology) in the network (i.e., edge $(v_j, v_i) \in \mathcal{E}$, if node v_i can receive information directly from node v_j).

As mentioned earlier, our scheme views each node v_i as a (small) linear dynamical controller, with (possibly vector) state z_i. Each node updates the state of its controller as a linear combination of the states of its neighbors and its own state. The state update for node v_i can also include a linear combination of the plant outputs from all plant sensors in v_i's neighborhood.

For example, consider the network presented in Fig. 13, where at the beginning of a time frame each node has an initial state value denoted by z_i (Fig. 13(a)). If each node maintains a scalar state, the size of the state is just **2 bytes**.[10] In the first time slot of a frame (Fig. 13(b)) node v_4 transmits its state, and in the second slot node v_5 transmits the state, etc. Finally, in the 6^{th} slot node v_3 is the last node in the frame to transmit its state (Fig. 13(g)). This results in a communication schedule as depicted in Fig. 13(h). After slot 6, node v_4 is informed about all its neighbors' states, which enables it to update its state by activating the WCN task. The task has to compute the updated state value before the node is scheduled for transmission in the next frame.

In the general case, if $z_i[k]$ denotes the i^{th} node's state at time step (i.e., communication frame) k, the **runtime update procedure** is:

$$z_i[k+1] = w_{ii} z_i[k] + \sum_{v_j \in \mathcal{N}_{v_i}} w_{ij} z_j[k] + \sum_{s_j \in \mathcal{N}_{v_i}} h_{ij} y_j[k], \quad (6)$$

where the neighborhood of a vertex v is represented as $\mathcal{N}_v$ and $y_j[k]$ is the measurement provided by sensor s_j. We will model the resource constraints of each node in the network by limiting the size of the state vector that can be maintained by each node.[11] Note the similarity of the update (6) to the state update equation for traditional dynamical controllers of the form (5); the state $z_i[k]$ plays the role of $x_c[k]$, the weights w_{ii} and w_{ij} play the role of $\mathbf{A}_c$ and the columns of $\mathbf{B}_c$, respectively.

To enable interaction between the network and the plant, each actuator a_i applies input $u_i[k]$, which is computed as a linear combination of states from the nodes in the neighborhood of the actuator:

$$u_i[k] = \sum_{j \in \mathcal{N}_{a_i}} g_{ij} z_j[k]. \quad (7)$$

Once again, note the resemblance of this applied input to the input applied by a standard controller of the form (5). Therefore, the behavior of each node in the network is determined by values w_{ij}, h_{ij} and g_{ij}. Aggregating the state values of all nodes at time step k into the value vector $\mathbf{z}[k]$, we see that the above individual controllers at each node collectively cause the entire network to act as a dynamical controller of the form:

$$\begin{aligned}
\mathbf{z}[k+1] &= \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}}_{\mathbf{W}} \mathbf{z}[k] + \\
&\quad + \underbrace{\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1p} \\ h_{21} & h_{22} & \cdots & h_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{Np} \end{bmatrix}}_{\mathbf{H}} \mathbf{y}[k] \\
&= \mathbf{W}\mathbf{z}[k] + \mathbf{H}\mathbf{y}[k] \ , \\
\mathbf{u}[k] &= \underbrace{\begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{mN} \end{bmatrix}}_{\mathbf{G}} \mathbf{z}[k] = \mathbf{G}\mathbf{z}[k]
\end{aligned}$$

for all $k \in \mathbb{N}$. Since for all $i \in \{1, \ldots, N\}$, $w_{ij} = 0$ if $v_j \notin \mathcal{N}_{v_i}$, $h_{ij} = 0$ if $s_j \notin \mathcal{N}_{v_i}$, and $g_{ij} = 0$ if $v_j \notin \mathcal{N}_{a_i}$

[10] Given that standard analog-to-digital converters have a precision of 12-16 bits, two bytes suffice for scalar values.

[11] To present our results, we will focus on the case where each node's state is a scalar. The general case, where each heterogeneous node can maintain a vector state with possibly different dimensions, can be treated with a natural extension of our approach (e.g., see [20]).

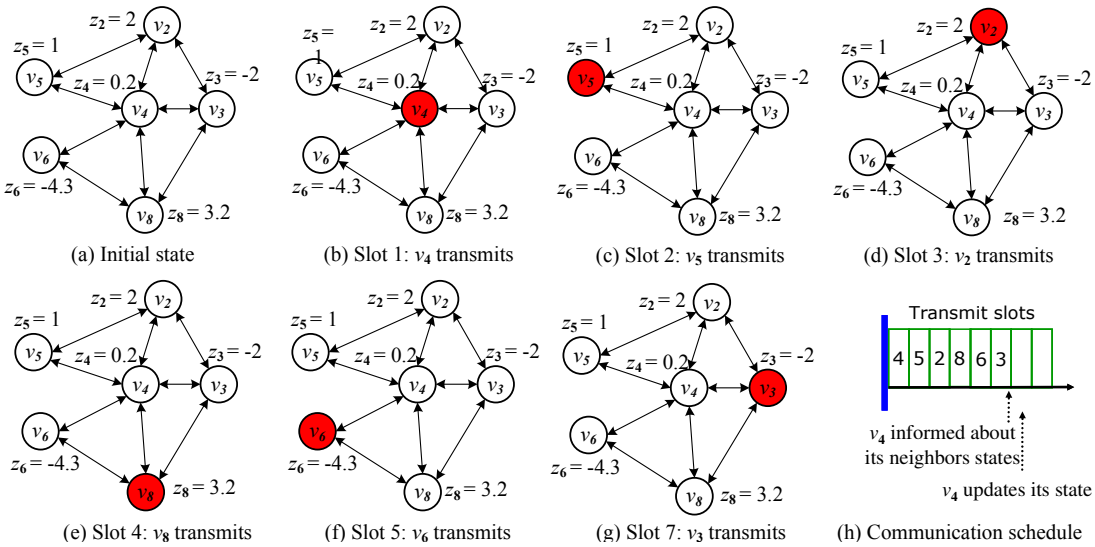


Fig. 13. An illustration of the WCN scheme for a simple network.

the matrices $\mathbf{W}, \mathbf{H}$ and $\mathbf{G}$ are *structured*, with sparsity constraints determined by the network topology at design time. Throughout the rest of the paper, we will define Ψ to be the set of all tuples $(\mathbf{W}, \mathbf{H}, \mathbf{G}) \in \mathbb{R}^{N\times N} \times \mathbb{R}^{N\times p} \times \mathbb{R}^{m\times N}$ satisfying the aforementioned sparsity constraints. Denoting the overall system state (plant's state and states of all nodes in the network) by $\hat{\mathbf{x}}[k] = \begin{bmatrix}\mathbf{x}[k]^T & \mathbf{z}[k]^T\end{bmatrix}^T$, the closed-loop system evolves as:

$$\begin{aligned}\hat{\mathbf{x}}[k+1] &= \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{BG}\\ \mathbf{HC} & \mathbf{W}\end{bmatrix}}_{\hat{\mathbf{A}}}\underbrace{\begin{bmatrix}\mathbf{x}[k]\\ \mathbf{z}[k]\end{bmatrix}}_{\hat{\mathbf{x}}[k]} + \underbrace{\begin{bmatrix}\mathbf{B}_w\\ \mathbf{0}\end{bmatrix}}_{\hat{\mathbf{B}}}\mathbf{u}_w \\ &= \hat{\mathbf{A}}\hat{\mathbf{x}}[k] + \hat{\mathbf{B}}\mathbf{u}_w[k].\end{aligned} \tag{8}$$

To use the WCN runtime scheme it is essential to determine an appropriate set of link weights (w_{ij}, h_{ij} and g_{ij}) at **design-time**, so that the closed loop system is asymptotically stable.[12] When there are no disturbances (i.e., $\mathbf{u}_w[k] \equiv \mathbf{0}$), an initial procedure was proposed for the basic WCN that guarantees that the closed-loop system is stable, or Mean Square Stable (MSS) if the communication links are unreliable.[13]

1) Advantages of the WCN: The WCN introduces very low communication and computation overhead. The linear iterative runtime procedure (6) is computationally very inexpensive as each node only computes a linear combination of its value and values of its neighbors. This makes it suitable for resource constrained, low-power wireless nodes (e.g., Tmote). Furthermore, the communication overhead is also very small, as each node needs to transmit only its own state once per frame. In the case when a node maintains a scalar state it transmits only *2 bytes* in each message, making it suitable to combine this scheme with periodic message transmissions in existing wireless systems.

Another key benefit is that the WCN can easily handle plants with multiple geographically distributed sensors and actuators, a case that is not easily handled by the "sensor $\rightarrow$ channel $\rightarrow$ controller/estimator $\rightarrow$ channel $\rightarrow$ actuator" setup commonly adopted in networked control design. The existence of a centralized controller might impose a requirement that the sampling time of the plant is greater than or equal to the sum of communication delays, from sensors to the controller and from the controller to the actuator, along with the time required for the computation of the control algorithm. The WCN does not rely on the existence of centralized controllers, and inherently captures the case of nodes exchanging values with the plant at various points in the network. Therefore, when the WCN is used, the network diameter does not affect the sampling period of the plant.

Finally, the WCN utilizes a simple transmission schedule where each node is active only once during a TDMA cycle and the control-loop does not impose end-to-end delay requirements. This allows the network operator to decouple the computation schedule from the communication schedule, which significantly simplifies closed-loop system design and enables compositional design and analysis. As long as each node can send additional states in a single transmission packet, and schedule computation of additional linear procedures, adding a new control loop will not affect the performance of the existing control loops. For example, consider IEEE 802.14.5 networks that have the maximal packet size of 128 bytes. If each plant is controlled using the WCN scheme where all nodes maintain a scalar 16 bit state value, then up to 64 plants can be controlled in parallel.

In this paper, we provide an enhanced WCN scheme that maintains all of these desirable properties, and further incorporates optimality and robustness metrics into the basic scheme.

2) Synchronization Requirements: For the network sizes considered here, it is necessary to use either hardware-

[12] A linear system $\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k]$ is asymptotically stable if for any $\mathbf{x}[0]$, $\lim_{k\to\infty}\mathbf{x}[k] = \mathbf{0}$. This is equivalent to saying that all eigenvalues of $\mathbf{A}$ have magnitude less than 1.

[13] A switched system described as $\mathbf{x}[k+1] = \mathbf{A}_{\theta(\mathbf{k})}\mathbf{x}[k]$, where subscript $\theta(k)$ describes time-variations caused by (probabilistic) drops of communication packets, is mean-square stable if for any initial state $(\mathbf{x}[0], \theta(0))$, $\lim_{k\to\infty}\mathbb{E}\left[\|\mathbf{x}[k]\|^2\right] = 0$, where the expectation is with respect to the probability distribution of the packet drop sequence $\theta(k)$ [57], [58].

based out-of-band synchronization or some of the built-in synchronization protocols that guarantee low synchronization error between neighboring nodes (e.g., the approach described in [59] guarantees that the maximal synchronization error between neighboring nodes is less than 1 μs). Even for 10 μs synchronization error between neighboring nodes, for large scale networks with the network diameter less than 100 nodes, maximal synchronization error between nodes is less than 1ms, which is significantly smaller than standard sampling rates of the plant when WCN is used. For example, if communication frames that consist of 16 slots are used, where each slot is 10 ms wide, the sampling period of the plant equals to 160 ms. In this case, synchronization errors would take less than 1% of the sampling period. We employ a synchronized network and use the RT-Link [43] time synchronized protocol in our evaluation. Time synchronized network protocols are the norm in the control automation industry, and two recent standards, WirelessHART[60] and ISA 100.11a [61] utilize a time division multiplexing link protocol.

X. Synthesis of an Optimal WCN

In this section we present a design-time method to determine a WCN configuration (i.e., link weights for a network with predefined topology) that minimizes effects of the disturbances acting on the system. More specifically, consider the model of the closed-loop system from (8), and assume that we want to minimize the influence of the disturbance input $\mathbf{u}_w$ on the vector $\hat{\mathbf{y}} = \hat{\mathbf{C}}\hat{\mathbf{x}}[k]$, for some matrix $\hat{\mathbf{C}}$. For example, if we would like to focus on minimizing the effects on the plant's state $\mathbf{x}$, we would define $\hat{\mathbf{C}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$. Thus, we can consider the vector $\hat{\mathbf{y}}$ as the 'output' of the system:

$$\begin{aligned} \hat{\mathbf{x}}[k+1] &= \hat{\mathbf{A}}\hat{\mathbf{x}}[k] + \hat{\mathbf{B}}\mathbf{u}_w[k] \\ \hat{\mathbf{y}} &= \hat{\mathbf{C}}\hat{\mathbf{x}}[k]. \end{aligned} \tag{9}$$

To determine the effect of the disturbance on the system's outputs, it is necessary to define a unit of measure to capture the 'size' of discrete-time signals. We will use the norms: $\|\mathbf{v}\|_{\ell_2} \triangleq \left(\sum_{k=0}^{\infty} \|\mathbf{v}[k]\|^2\right)^{1/2}$ and $\|\mathbf{v}\|_{\ell_\infty} \triangleq \sup_{k\geq 0} \|\mathbf{v}[k]\|$. Furthermore, the notion of a *system gain* is introduced to classify the worst-case system response to limited energy input disturbances.

Definition 1 ([62]): System gains for the discrete-time system (9) are defined as:

- **Energy-to-Peak Gain**: $\gamma_{ep} = \sup_{\|\mathbf{u}_w\|_{\ell_2}\leq 1} \|\hat{\mathbf{y}}\|_{\ell_\infty}$
- **Energy-to-Energy Gain**: $\gamma_{ee} = \sup_{\|\mathbf{u}_w\|_{\ell_2}\leq 1} \|\hat{\mathbf{y}}\|_{\ell_2}$

□

We will require the following result from [63].

Theorem 1: Suppose that the system (9) is asymptotically stable and consider any nonnegative $\gamma \in \mathbb{R}$.
(a) $\gamma_{ep} < \gamma$ if and only if there exist matrices $\mathcal{X} \succ 0, \Upsilon \succeq 0$ and $\mathcal{Z}$ such that $\Upsilon \prec \gamma \mathbf{I}$ and

$$\mathcal{R}(\mathcal{X}, \mathcal{Z}, \Upsilon, \mathcal{X}^{-1}) = \begin{bmatrix} \mathcal{X} & \mathcal{Z} & \hat{\mathbf{A}} & \hat{\mathbf{B}} \\ \mathcal{Z}^T & \Upsilon & \hat{\mathbf{C}} & \mathbf{0} \\ \hat{\mathbf{A}}^T & \hat{\mathbf{C}}^T & \mathcal{X}^{-1} & \mathbf{0} \\ \hat{\mathbf{B}}^T & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \succ 0 \tag{10}$$

(b) $\gamma_{ee} < \gamma$ if and only if there exist matrices $\mathcal{X} \succ 0, \Upsilon \succeq 0$ such that $\Upsilon \prec \gamma^2 \mathbf{I}$ and (10) holds for $\mathcal{Z} = \mathbf{0}$. □

Only the matrix $\hat{\mathbf{A}}$ contains the WCN parameters, aggregated in the structured matrices $\mathbf{W}, \mathbf{G}, \mathbf{H}$ (from (8)). Our goal is to determine matrices $\mathbf{W}, \mathbf{G}, \mathbf{H}$ that satisfy the imposed structural constraints, along with matrices $\mathcal{X}, \mathcal{Z}, \Upsilon$, for which the value γ is minimized.

The constraint (10) is linear with respect to all variables, except the matrix $\mathcal{X}$ (due to the presence of the term $\mathcal{X}^{-1}$). This term causes the problem of solving the matrix inequality to be non-convex. To ameliorate this issue and efficiently solve the optimization problem, we linearize the $\mathcal{X}^{-1}$ term. As shown in [63], the Taylor series expansion of $\mathcal{X}^{-1}$ 'around' any matrix $\mathcal{X}_k$ is

$$LIN(\mathcal{X}^{-1}, \mathcal{X}_k) = \mathcal{X}_k^{-1} - \mathcal{X}_k^{-1}(\mathcal{X} - \mathcal{X}_k)\mathcal{X}_k^{-1}. \tag{11}$$

With the above linearization we obtain a linear matrix inequality (LMI) for the constraint 10. As in [63], [64], we can now define an iterative algorithm to minimize γ, while ensuring that the constraint from (10) is satisfied. This is achieved by replacing the term $\mathcal{X}^{-1}$ with $LIN(\mathcal{X}^{-1}, \mathcal{X}_k)$ in each iteration, which results in Algorithm 1. Note that $\hat{\mathbf{A}}(\mathbf{W}, \mathbf{H}, \mathbf{G})$ denotes the matrix $\hat{\mathbf{A}}$ obtained from matrices $\mathbf{W}$, $\mathbf{H}$, $\mathbf{G}$ as defined in (8). Finally, for γ obtained from Algorithm 1, $\sqrt{\gamma}$ should be used if we had optimized for γ_{ee}.

Consider the sequence $\{\gamma_k\}_{k\geq 0}$ obtained from Algorithm 1. As shown in [63], the linearization from (11) guarantees that for each $k \geq 0$, in step $k+1$ there exists a feasible matrix in an open neighborhood of the point $\mathcal{X}_k$ for which there exists γ, such that $\gamma \leq \gamma_k$. Since γ_{k+1} is the minimum in that iteration, it follows that $\gamma_{k+1} \leq \gamma$. Thus, the sequence $\{\gamma_k\}_{k\geq 0}$ is non-increasing and bounded ($\gamma_k \geq 0$), meaning that it will always converge. Since we are optimizing a convex function over a non-convex set, by linearizing the constraints we might obtain a sub-optimal WCN configuration. The final result and the convergence rate depend on the initial point (from Step 1. of the algorithm). Finally, the smallest ϵ for which we can find an optimal controller can be obtained using bisection on the parameter ϵ.

XI. WCN: Robustness to Link Failures

We now describe the main limitation of the basic WCN, and extend the WCN scheme to improve its robustness to link failures.

The unreliability of wireless communication links is one of the main drawbacks when wireless networks are used for control. When communication links in the feedback loop fail according to a given probability distribution, the notion of asymptotic stability is typically relaxed to settle for *mean square stability* (MSS), where the expected value of the norm of the state stays bounded. For the basic WCN, we proposed a design-time procedure that can be used to extract a stabilizing configuration that guarantees MSS despite unreliable communication links [20]. For example, consider the system from Fig. 14 with a scalar plant, where $\alpha = 2$ (the plant is unstable), and assume that the link between node v_2 and the actuator is reliable (i.e., never drops packets). The basic WCN

Algorithm 1 ***Design-time*** procedure used to extract optimal WCN configuration

1. Set $\epsilon > 0, k = 0$. Find a feasible point $\mathcal{X}_0, \mathcal{Y}_0, \Upsilon_0 \succ 0$, $\hat{\mathbf{A}}(\mathbf{W_0}, \mathbf{H_0}, \mathbf{G_0})$, such that $\mathcal{R}(\mathcal{X}_0, \mathcal{Z}, \Upsilon_0, \mathcal{Y}_0) \succ 0$, $\mathcal{X}_0 \succeq \mathcal{Y}_0^{-1}$ and $(\mathbf{W}_0, \mathbf{H}_0, \mathbf{G}_0) \in \Psi$. If a feasible point does not exist, it is not possible to stabilize the system with this network topology.

2. At iteration k ($k \geq 0$), from $\mathcal{X}_k$ obtain the matrix $\mathcal{X}_{k+1}$ and scalar γ_{k+1} by solving the LMI problem

$$\mathcal{X}_{k+1} = \arg \min_{\mathcal{X},\mathcal{Z},\Upsilon,\mathbf{W},\mathbf{H},\mathbf{G},\gamma_{k+1}} \gamma_{k+1} \tag{12}$$

$$\mathcal{R}(\mathcal{X}, \mathcal{Z}, \Upsilon, LIN(\mathcal{X}^{-1}, \mathcal{X}_k)) \succ 0, \tag{13}$$

$$\Upsilon \prec \gamma_{k+1}\mathbf{I}, \tag{14}$$

$$(\mathbf{W}, \mathbf{H}, \mathbf{G}) \in \Psi, \;\; \mathcal{X} \succ 0, \Upsilon \succeq 0 \tag{15}$$

if γ_{ee} is being optimized, add the constraint $\mathcal{Z} = \mathbf{0}$.

3. If $\gamma_{k+1} < \epsilon$ stop the algorithm. Otherwise, set $k = k+1$ and go to the step **2**.

scheme, where each node maintains a scalar state, guarantees that the closed-loop system is MSS for probabilities of packet drops $\leq 1.18\%$.

To place this result in context, it is worth comparing it with the theoretical limit of robustness in lossy networks from [21]. The work in [21] considers a system with a plant controlled by a centralized controller, which is connected to the plant using a single wireless link between a sensor and the controller. In addition, the controller is connected to the actuators with a set of wired connections. It was shown that for this setup, the system can not be stabilized with a linear controller for probability of message drops p greater than $\frac{1}{|\lambda_{max}|^2}$, where $|\lambda_{max}|$ denotes the maximal norm of the plant's eigenvalues (i.e., eigenvalues of $\mathbf{A}$ from (4)). For the plant from Fig. 14, this would mean that a centralized controller in the aforementioned setup cannot provide MSS of the plant if the probability of message drops is higher than 25% (since $\alpha = 2$). This value is significantly larger than the 1.18% value obtained when the basic WCN scheme is used. We now show how the basic WCN formulation presented in (6), (7) can be modified to significantly improve tolerance to packet drops.

A. WCN with Observer Style Updates

To improve WCN robustness to independent link failures, we now allow each node in the network to use different weights in each time step, depending on which neighbors' transmissions were successfully received. Thus, we define the update procedure as:

$$z_j[k+1] = \tilde{w}_{jj} z_j[k] + \sum_{i \in \mathcal{N}_{v_j}} \tilde{w}_{ji} z_i[k],^{14} \tag{16}$$

where $\tilde{w}_{ji} = 0$ if the message from the node v_i was not received, or w_{ji} otherwise.[15] More importantly, $\tilde{w}_{jj}$ depends on a newly introduced set of link weights (q_{ji}): $\tilde{w}_{jj} = w_{jj} - \sum_{i \in \mathcal{N}_{v_j}} \tilde{q}_{ji}$. Here, $\tilde{q}_{ji} = 0$ if the message from the node v_i was not received, and q_{ji} (a free parameter that will be carefully designed) otherwise.

To model the WCN that employs the above scheme, we need to model the links in the network. We utilize the approach proposed in [58], where each unreliable link $\xi_{ji} = (v_i, v_j)$ (i.e., $v_i \rightarrow v_j$) can be modeled as a memoryless, discrete, independent and identically distributed (IID) random process ξ_{ji}. Here, IID implies that the random variables $\{\xi_{ji}[k]\}_{k \geq 0}$ are IID.[16] For each link, these random processes map each transmitted value t_{ji} into a received value $\xi_{ji}[k]t_{ji}$ (see Fig. 15).

With this link model, (16) can be described as:

$$z_j[k+1] = (w_{jj} - \sum_{i \in \mathcal{N}_{v_j}} \xi_{ji} q_{ji}) z_j[k] + \sum_{i \in \mathcal{N}_{v_j}} \xi_{ji} w_{ji} z_i[k],$$

Remark 1: If we consider the case with reliable communication links, the update procedure for each node v_j in the network can be described as:

$$z_j[k+1] = w_{jj} z_j[k] + \sum_{i \in \mathcal{N}_{v_j}} (w_{ji} z_i[k] - q_{ji} z_j[k]), \tag{17}$$

Since the above equation has the standard observer structure [65], we refer to this scheme as the WCN with observer style updates (as in [37]). □

Following the approach from [58], each link described with a random process ξ_{ji} can be specified with a fixed gain, corresponding to the mean value of the random variable, and the zero-mean random part: $\xi_{ji} = \mu_{ji} + \Delta_{ji}$. For example, if each link (i.e., random process ξ_{ji}) is described as a Bernoulli process with probability $p_{ji} \leq 1$ (i.e., the link delivers the transmitted message with probability p_{ji}), then $\mu_{ji} = p_{ji}$ and Δ_{ji} can have values $-p_{ji}$ and $1-p_{ji}$, with probabilities $1-p_{ji}$ and p_{ji}, respectively. Therefore, the above procedure becomes:

$$z_j[k+1] = (w_{jj} - \sum_{i \in \mathcal{N}_{v_j}} \mu_{ji} q_{ji}) z_j[k] + \sum_{i \in \mathcal{N}_{v_j}} \mu_{ji} w_{ji} z_i[k] + \sum_{i \in \mathcal{N}_{v_j}} \Delta_{ji} (w_{ji} z_i[k] - q_{ji} z_j[k]).$$

We define $r_t[k] := (w_{ji} z_i[k] - q_{ji} z_j[k])$, for each link $t = (v_i, v_j)$. Also, for each link $t = (s_i, v_j)$ we denote $r_t[k] := (h_{ji} y_i[k] - q_{ji} z_j[k])$. After aggregating all of the $r_t[k]$'s in a vector $\mathbf{r}[k]$ of length N_l (where N_l is the number of links), we obtain:

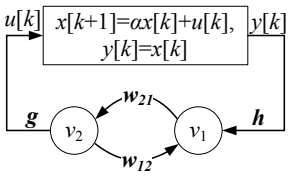


Fig. 14. An example of the WCN: A plant with a scalar state controlled by a WCN.

[14] A similar update is introduced for nodes that receive sensor values. This part has been omitted for ease of exposition.

[15] Although these weights are technically time varying (i.e., they depend on k), we use this notation for simplicity.

[16] We will address these assumptions later in this section.

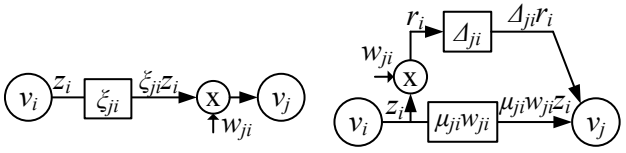


Fig. 15. Communication over a non-deterministic channel; (a) A link between nodes v_i and v_j; (b) Link transformation into a robust control form.

$$\mathbf{r}[k] = \mathbf{J}^{or}\begin{bmatrix}\mathbf{y}[k]\\ \mathbf{z}[k]\end{bmatrix} = \underbrace{\mathbf{J}^{or}\begin{bmatrix}\mathbf{C} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_N\end{bmatrix}}_{\hat{\mathbf{J}}^{or}}\hat{\mathbf{x}}[k]. \tag{18}$$

Each row of the matrix $\mathbf{J}^{or} \in \mathbb{R}^{N_l\times(N+p)}$ contains up to two nonzero elements, equal to a gain w_t, h_t, g_t or $-q_t$.

This allows us to model the behavior of the closed-loop system with unreliable communication. Specifically, the update equation for each node v_j is:

$$z_j[k+1] = (w_{jj} - \sum_{i\in\mathcal{N}_{v_j}}\mu_{ji}q_{ji})z_j[k] + \sum_{t=(v_i,v_j)}\mu_t w_t z_i[k] + \sum_{t=(s_i,v_j)}\mu_t h_t y_i[k] + \sum_{t=(v_i,v_j)}\Delta_t[k]r_t[k] + \sum_{t=(s_i,v_j)}\Delta_t[k]r_t[k]$$

Similarly, the input value applied by each actuator at time k is:

$$u_j[k] = \sum_{t=(v_i,a_j)}\mu_t g_t z_i[k] + \sum_{t=(v_i,a_j)}\Delta_t[k]r_t[k].$$

Finally, denoting $\Delta[k] = \text{diag}(\{\Delta_t[k]\}_{t=1}^{N_l})$, the above expressions can be written in vector form as:

$$\mathbf{z}[k+1] = \mathbf{W}_\mu\mathbf{z}[k] + \mathbf{H}_\mu\mathbf{y}[k] + \mathbf{J}_v^{dst}\Delta[k]\mathbf{r}[k], \tag{19}$$

$$\mathbf{u}[k] = \mathbf{G}_\mu\mathbf{z}[k] + \mathbf{J}_u^{dst}\Delta[k]\mathbf{r}[k], \tag{20}$$

where all elements of matrices $\mathbf{W}_\mu, \mathbf{H}_\mu$ and $\mathbf{G}_\mu$ (except the diagonal entries of $\mathbf{W}_\mu$) are of the form $\mu_{ji}w_{ji}, \mu_{ji}h_{ji}$ and $\mu_{ji}g_{ji}$, respectively. The diagonal entries of $\mathbf{W}_\mu$ are of the form $w_{jj} - \sum_{i\in\mathcal{N}_{v_j}}\mu_{ji}q_{ji}$. The binary matrices $\mathbf{J}_v^{dst}$ and $\mathbf{J}_u^{dst}$ are designed in a way that each row of the matrices selects elements of the vector $\Delta[k]\mathbf{r}[k]$ that are added to the linear combinations calculated by the nodes and the actuators. If we denote $\mathbf{J}^{dst} = \begin{bmatrix}\mathbf{J}_u^{dst}\\ \mathbf{J}_v^{dst}\end{bmatrix}$ the overall system with unreliable links can be modeled as:

$$\hat{\mathbf{x}}[k+1] = \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{B}\mathbf{G}_\mu\\ \mathbf{H}_\mu\mathbf{C} & \mathbf{W}_\mu\end{bmatrix}}_{\hat{\mathbf{A}}_\mu}\hat{\mathbf{x}}[k] + \underbrace{\begin{bmatrix}\mathbf{B} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_N\end{bmatrix}\mathbf{J}^{dst}}_{\hat{\mathbf{J}}^{dst}}\mathbf{\Delta}[k]\mathbf{r}[k], \tag{21}$$

with $\mathbf{r}[k]$ given by (18). Now, using the same approach as in [58], [20], the following theorem can be proven.

Theorem 2: The system from (21) is MSS if and only if there exist matrices $\mathcal{X}, \mathcal{Y} \succ 0$ and scalars $\alpha_1, ..., \alpha_{N_l}$ such that

$$\begin{bmatrix}\mathcal{X} - \hat{\mathbf{J}}^{dst}\text{diag}\{\alpha\}(\hat{\mathbf{J}}^{dst})^T & \hat{\mathbf{A}}_\mu\\ \hat{\mathbf{A}}_\mu^T & \mathcal{Y}\end{bmatrix} \succ 0 \tag{22}$$

$$\mathcal{Y} = \mathcal{X}^{-1} \tag{23}$$

$$\alpha_i \geq \sigma_i^2(\hat{\mathbf{J}}^{or})_i\mathcal{Y}^{-1}(\hat{\mathbf{J}}^{or})_i^T,\ \forall i \in \{1,\dots,N_l\} \tag{24}$$

where $(\hat{\mathbf{J}}^{or})_i$ denotes the i^{th} row of the matrix $\hat{\mathbf{J}}^{or}$. □

A procedure based on LMIs, with the same structure as Algorithm 1, can be used in this case to compute a WCN configuration that guarantees MSS of the closed-loop system with error-prone links. The difference from Algorithm 1 is that in Step 2, the following problem should be solved:

$$\mathcal{X}_{k+1} = \arg\min_{\mathcal{X},\mathcal{Y},\Upsilon,\mathbf{W},\mathbf{H},\mathbf{G}} tr(\Upsilon)$$
$$\mathcal{Y} - LIN(\mathcal{X}^{-1}, \mathcal{X}_k) \prec \Upsilon, \quad \mathcal{X} \succeq \mathcal{Y}^{-1}$$
such that the constraints from (22),(24),(15) are valid,

where $tr(\mathbf{A})$ denotes the trace of the matrix $\mathbf{A}$. Note that the above algorithm adds only one additional LMI constraint for each link in the network.

1) Validity of the Assumptions: While developing the model of the WCN from (19), we have assumed that all links in the network are memoryless and independent. Memoryless channels can be obtained if channel hopping is used at the network layer [66]. However, the physical placement of the nodes might introduce correlation between some of the network links.

If these IID assumptions are not valid (or too simplistic), we must model correlation between links along with more complex link failures (such as those induced by a Markov process). In these cases, an approach similar to [57] can be used, which would result in an exponential number of additional constraints introduced to deal with link failures (compared to the linear number of additional constraints introduced under the IID assumption of independent and memoryless channels). Except for very large scale systems, the observer style update procedure is practical as the computation of WCN configurations $(\mathbf{W}, \mathbf{H}, \mathbf{G})$ is only required at design time.

2) Evaluation: We evaluated the performance of the proposed scheme by modeling all links as independent Bernoulli processes. To analyze robustness of the WCN with observer style updates, we first analyzed the performance of WCNs with $N \geq 2$ nodes that create a complete graph. The WCN is used for control of a single-state plant shown in Fig. 14 (with $\alpha > 1$). Node v_1 receives the plant output $y[k] = x[k]$ at each time-step k, and the input to the plant is derived as a scaled version of the transmission of the node v_2 (i.e., $u[k] = gz_2[k]$ for a scalar g). Using the bisection method from [57], we extracted the maximal probabilities of message drops (p_m) for which there exists a stabilizing configuration that ensures MSS.

We considered two scenarios: In the first scenario, we have compared the performance of the basic WCN with that of the WCN with observer style updates (denoted oWCN). We analyzed networks where all the links are unreliable, described with the same probability of packet drops p (including the links between the plant and the network nodes). The results are presented in Fig. 16(a). In addition, we have investigated the case where the link between node v_2 and the plant's actuator is reliable (without any packet drops). The results are shown in Fig. 16(b). As can be observed, the proposed scheme **significantly** improves system robustness to link failures. *For*

	WCN (scalar state)	WCN ($\mathbb{R}^2$ state)	$oWCN$ (scalar state)	$oWCN$ ($\mathbb{R}^2$ state)
$N=2$	$p_m = 0.69\%$	$p_m = 0.72\%$	$p_m = 1.64\%$	$p_m = 1.82\%$
$N=3$	$p_m = 0.74\%$	$p_m = 0.77\%$	$p_m = 1.66\%$	$p_m = 1.88\%$
$N=4$	$p_m = 0.77\%$	$p_m = 0.79\%$	$p_m = 1.66\%$	$p_m = 1.88\%$

(a) With all links being unreliable

	WCN (scalar state)	WCN ($\mathbb{R}^2$ state)	$oWCN$ (scalar state)	$oWCN$ ($\mathbb{R}^2$ state)
$N=2$	$p_m = 1.18\%$	$p_m = 1.30\%$	$p_m = 10.46\%$	$p_m = 17.82\%$
$N=3$	$p_m = 1.32\%$	$p_m = 1.46\%$	$p_m = 11.24\%$	$p_m = 17.88\%$
$N=4$	$p_m = 1.41\%$	$p_m = 1.54\%$	$p_m = 11.46\%$	$p_m = 17.88\%$
		$oWCN$ ($\mathbb{R}^3$ state)	$oWCN$ ($\mathbb{R}^4$ state)	$oWCN$ ($\mathbb{R}^5$ state)
$N=2$		$p_m = 20.40\%$	$p_m = 20.48\%$	$p_m = 20.64\%$

(b) With a reliable link between the node v_2 and actuator

Fig. 16. Maximal probabilities of link failures for which the closed-loop system from Fig. 14 ($\alpha = 2$) is MSS, when controlled without (WCN) and with observer style updates (oWCN).

example, the WCN with observer style updates guarantees MSS for the system from Fig. 14 even when the probability of link failures is more than 20% (compared to 1.5% for the basic WCN). Similarly, going back to the discussion from the beginning of the section, we have shown in this simple example that the WCN performance is much closer to that of the optimal centralized controllers used for control over wireless links (guaranteeing MSS with up to 25% packet drops).

Using the observer style updates, similar significantly improved results were obtained for the more complex examples from [20], including larger plants with multiple inputs and outputs, controlled by a mesh network with 9 nodes.

XII. WCN: Robustness to Node Failures

The stability of the closed-loop system, described by (8), can be affected by node crash failures (i.e., nodes that stop working and drop out of the network). Currently, we have considered two approaches to deal with the node failures. One obvious method to deal with up to k node failures is to precompute at the design-time a set of $N_k = \sum_{j=0}^{k} \binom{N}{j}$ different stabilizing configurations $(\mathbf{W}, \mathbf{H}, \mathbf{G})$ that correspond to all possible choices of k or fewer failed nodes. In this case, each node would need to maintain N_k different sets of link weights for all its incoming links. For example, if each node in the WCN maintains a scalar state, a node with d neighbors would have to maintain on the order of $d \cdot N_k$ different scalar weights. The switching between the precomputed stabilizing configurations could be done either by implementing the detection algorithm from [24], or by having the neighbors of failed nodes broadcast the news of the failures throughout the network, which will prompt all nodes to switch to the appropriate choice of $(\mathbf{W}, \mathbf{H}, \mathbf{G})$.

A more sophisticated method for dealing with the node failures would be to design the WCN in a way that even if some of the nodes fail, the closed-loop system remains stable. For simplicity, consider a WCN that can deal with a single node failure. Let us denote with $\hat{\mathbf{A}}_i$ the matrix $\hat{\mathbf{A}}$ from (8) in the case when node i dies. This is equivalent to setting to zero the i^{th} row of matrices $\mathbf{W}$ and $\mathbf{H}$, along with the i^{th} column of $\mathbf{W}$ and $\mathbf{G}$:

$$\hat{\mathbf{A}}_i \triangleq \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{G}\mathbf{I}_N^i \\ \mathbf{I}_N^i\mathbf{H}\mathbf{C} & \mathbf{I}_N^i\mathbf{W}\mathbf{I}_N^i \end{bmatrix}, \ i = 1, \ldots, N, \tag{25}$$

Here, $\mathbf{I}_N^i$ denotes $N \times N$ diagonal matrix, with all ones on the diagonal except at the i^{th} position. A sufficient condition for system stability in this case is that there exists a positive definite matrix $\mathcal{X}$ (and, thus, a common Lyapunov function $V(\hat{\mathbf{x}}) = \hat{\mathbf{x}}^T \mathcal{X} \hat{\mathbf{x}}$) such that $\mathcal{X} - \hat{\mathbf{A}}^T \mathcal{X} \hat{\mathbf{A}} \succ \mathbf{0}$ and

$$\mathcal{X} - \hat{\mathbf{A}}_i^T \mathcal{X} \hat{\mathbf{A}}_i \succ \mathbf{0}, \ i = 1, 2, \ldots N. \tag{26}$$

Therefore, the procedure from the previous section with additional N LMI constraints, can be used to extract a stabilizing configuration that can deal with a single node failure. However, in this case it is necessary to design the network in a way that guarantees that such a stabilizing configuration exists. Initial results on these topological conditions have been presented in [23].

XIII. WCN: Control of Continuous-Time Plants

Optimal and stabilizing WCN configurations can be obtained using algorithms developed from the closed-loop system model (8) that contains a discrete-time model of the plant (4). However, a similar framework can be used for control of continuous-time plants by discretizing the controlled plant, while taking into account a subtle delay introduced by the communication schedule. To illustrate this, consider a standard continuous-time plant model:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}_c \mathbf{x}(t) + \mathbf{B}_c \mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}_c \mathbf{x}(t), \end{aligned} \tag{27}$$

with input $\mathbf{x}(t) \in \mathbb{R}^n$, output $\mathbf{y}(t) \in \mathbb{R}^p$, $\mathbf{u}(t) \in \mathbb{R}^m$ and matrices $\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c$ of the appropriate dimensions.[17] We denote the sampling period of the plant by T, and we assume that all sensors sample the plant outputs at the beginning of the zero-th slot (as shown in Fig. 17(a)). We also assume that all actuators are scheduled to apply their newly calculated inputs at the beginning of the h^{th} time slot. Note that $h > 0$, because from (7) each actuator has to first receive state values from all of its neighbors, before calculating its next plant input. Similarly, from (7) $h \geq \max(d_{a_i})$, where d_{a_i} denotes the number of neighbors of the actuator a_i.

Therefore, the new inputs will be applied to the plant with the delay $\tau = hT_{sl}$, where T_{sl} is the size of communication slots. This results in the input signal with the form shown in Fig. 17(b). Denoting the number of slots in a communication frame by F, we can write $T = FT_{sl}$. Using the approach from [4], [3], we describe the system:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}_c \mathbf{x}(t) + \mathbf{B}_c \mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{C}_c \mathbf{x}(t), \quad t \in [kT + \tau, (k+1)T + \tau), \\ \mathbf{u}(t^+) &= \mathbf{G}\mathbf{z}[k], \quad t \in \{kT + \tau, k = 0, 1, 2, \ldots\} \end{aligned} \tag{28}$$

where $\mathbf{u}(t^+)$ is a piecewise continuous function and only changes values at time instances $kT + \tau, \ k = 0, 1, \ldots$. From

[17] For simplicity we do not model disturbance inputs to the plant. However, the approach presented in this section can readily handle that scenario.

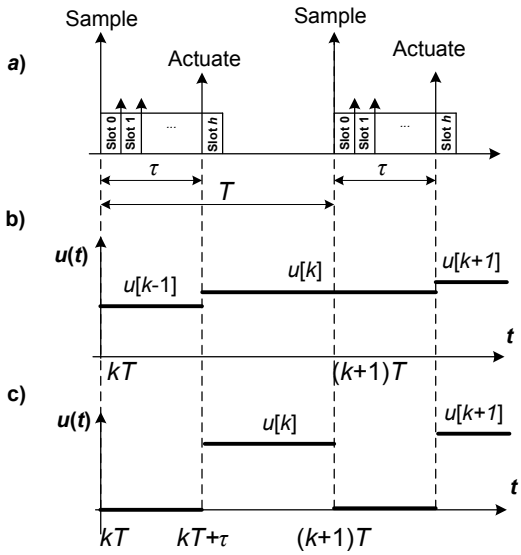


Fig. 17. (a) Scheduling sampling/actuation at the start of the slots; (b) Timing diagram for the first type of plant inputs; (c) Plant inputs when actuators reset the inputs at the beginning of the frames.

the above equation, the discretized model of the system with the sampling period T can be represented as [65]:

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{G}\mathbf{z}[k] + \mathbf{B}^{-}\mathbf{G}\mathbf{z}[k-1] \\ \mathbf{y}[k] &= \mathbf{C}\mathbf{x}[k],\end{aligned} \tag{29}$$

where $\mathbf{x}[k] = \mathbf{x}(kT),\ k \geq 0$ and

$$\mathbf{A} = e^{\mathbf{A}_c T},\ \ \mathbf{B} = \int_0^{T-\tau} e^{\mathbf{A}_c \delta}\mathbf{B}_c d\delta,\ \ \mathbf{B}^{-} = \int_{T-\tau}^{T} e^{\mathbf{A}_c \delta}\mathbf{B}_c d\delta. \tag{30}$$

When the communication schedule is extracted and the network is configured, the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{B}^{-}$ obtain fixed-values that depend on the continuous-time plant dynamics, communication frame size T (i.e., the sampling period of the plant) and the utilized communication schedule (as it determines the value for h).

If each actuator applies its current input only until the end of the corresponding frame and then forces its input to zero until the next actuation slot (i.e., h^{th} slot), the input signals would have the form shown in Fig. 17(c) (instead of the form from Fig. 17(b)). In this case, the discretized system could be specified as in (29), (30), with the difference that $\mathbf{B}^{-} = \mathbf{0}$. Therefore, the discrete-time system takes the form from (4), and stabilizing and optimal configurations can be obtained using the procedures described in the previous sections. However, due to the delay τ, the resulting discrete-time system could be uncontrollable, which in the general case would mean that there is no stabilizing configuration for the closed-loop system.

In situations where $(\mathbf{A}, \mathbf{B})$ is not controllable it is necessary for all actuators to apply their 'old' inputs until new inputs are available (as shown in Fig. 17(b)). This results in a discrete-time plant that does not have the form from (4), and the previous algorithms cannot be directly employed. However, by defining a new vector $\tilde{\mathbf{x}}[k] \triangleq \begin{bmatrix}\mathbf{x}[k]^T & \mathbf{u}[k-1]^T\end{bmatrix}^T$ the discrete-time system can be described as:

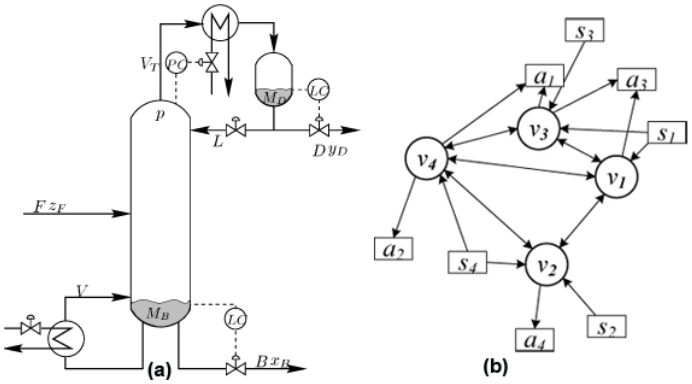


Fig. 18. (a) Structure of the distillation column [67]; (b) The network topology of the WCN corresponding to the sensor and actuator positions.

$$\tilde{\mathbf{x}}[k+1] = \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{B}^{-} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\tilde{\mathbf{A}}}\tilde{\mathbf{x}}[k] + \underbrace{\begin{bmatrix}\mathbf{B} \\ \mathbf{I}\end{bmatrix}}_{\tilde{\mathbf{B}}}\mathbf{u}[k] = \tilde{\mathbf{A}}\tilde{\mathbf{x}}[k] + \tilde{\mathbf{B}}\mathbf{u}[k],$$
$$\mathbf{y}[k] = \underbrace{\begin{bmatrix}\mathbf{C} & \mathbf{0}\end{bmatrix}}_{\tilde{\mathbf{C}}}\tilde{\mathbf{x}}[k] = \tilde{\mathbf{C}}\tilde{\mathbf{x}}[k]$$

The above system has the same form as (4) and, therefore, we can use the aforementioned algorithms to obtain a stabilizing or optimal configurations of the WCN.

XIV. WCN: Process Control Application

The WCN has been deployed on a process-in-the-loop test-bed with a plant running in Simulink and the plant's sensors and actuators connected to analog interfaces (see Fig. 19(a)). We first describe the plant's model, then the closed-loop wireless control test-bed and finally demonstrate the WCN use for control of the plant.

A. Case Study Description

To illustrate the use of the WCN, we consider the distillation column control (Fig. 18(a)), a well-known process control problem described in [67]. Four input flows (in $[mols/s]$) are available for the column control: reflux (L), boilup (V), distillate (D) and bottom flow (B). The goal is to control four outputs: x_D - top composition, x_B - bottom composition, M_D - liquid levels in condenser, and M_B - liquid levels in the reboiler (in $[mol]$). Finally, the column has two disturbances, feed flow-rate F and feed composition z_F. The columns are described using the continuous-time Linear Time Invariant (LTI) model from [67], where the state-space contains 8 states.

B. WCN Experimental Platform

We have implemented the WCN scheme on FireFly embedded wireless nodes [53] and TI's MSP430F5438 Experimenter Boards, both equipped with IEEE 802.15.4 standard-compliant radio transceivers. FireFly is a low-cost, low-power platform based on Atmel ATmega1281 8-bit microcontroller, while the experimenters board uses a 16-bit MSP430 microcontroller.

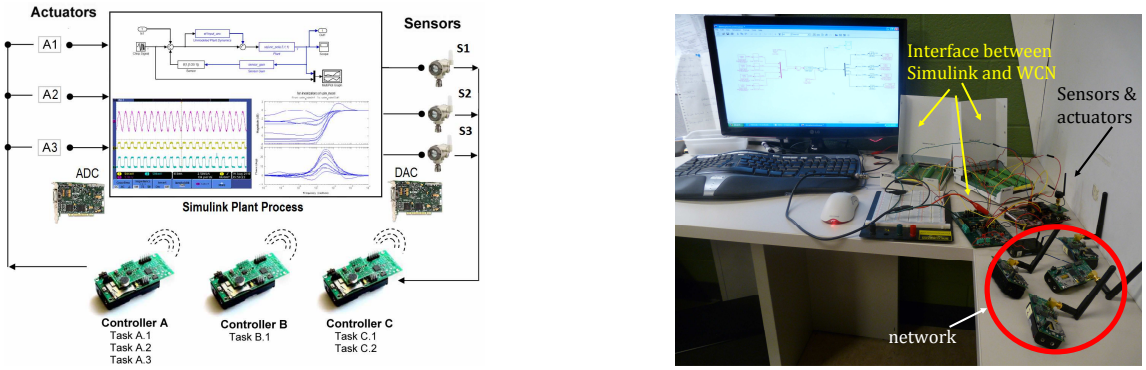


Fig. 19. Process-in-the-loop simulation of the distillation column control; (a) The plant model is simulated in Simulink, while the WCN is implemented on FireFly nodes; (b) Experimental setup used for the WCN validation.

Both platforms can be used for TDMA-based communication with the RT-Link protocol [43], and support in-band synchronization provided as a part of the protocol.

The WCN procedure on each wireless node was implemented as a simple task executed on top of the nano-RK, a Real-Time Operating System (RTOS) [42]. The WCN task had a 140.64ms period, equal to the RT-Link frame size (RT-Link was configured to use 16 slots of size 8.79 ms). Since the WCN requires a TTA, nano-RK has been modified to enable scheduling of sensing and actuation at the start of the desired slots. This guarantees synchronized actions at all sensors and all actuators.

The column, modeled as a continuous-time LTI system along with disturbances and measurement noise was run in Simulink in real-time using Real-Time Windows Target [68]. The interface between the model and the real hardware were two National Instruments PCI-6229 boards which provided analog outputs that correspond to the Simulink model's outputs (see Fig. 19(a)). The output signals were saturated between -4V and 4V, due to NI boards limitations. Also, to provide inputs to the Simulink model, the boards sampled the analog input signals within range [-4V, 4V], at a 1 kHz rate. Finally, Simulink's input and output signals were monitored and controlled with 4 sensors and 4 actuators positioned according to the distillation column structure (Fig. 18(a)). In addition, 4 real wireless controller nodes ($v_1 - v_4$) were added, resulting in the topology shown in Fig. 18(b).

C. WCN Results

From the communication and computation schedules, we obtained the discrete-time plant model using the discretization procedure from Section XIII (Eqs. (29),(30)), with sampling rate $T = 140.64\ ms$ (RT-Link frame size).

We first investigated the problem of providing MSS of the closed-loop system with uncorrelated random link failures and single node failures. Assigning each node to maintain a scalar state, using the procedures from Sections XI and XII we derived a stabilizing WCN configuration for the topology presented in Fig. 18(b) and the discretized LTI plant model. To solve the convex optimization problems we used the CVX, a package for specifying and solving convex programs [69].

We were able to obtain only WCN configurations that maintain stability if one of the nodes v_1-v_3 fails, meaning that the constraint from (26) for the node v_4 was violated (without v_4 the topology violates the conditions from [23], for existence of a stabilizing configuration). Fig. 20 shows obtained measurements where the disturbance inputs F, z_F were set to zero, while we provided periodical pulses to the input L. Although the output of the plant degrades when the node v_1 is turned off, the WCN maintains system stability. However, if the node v_4 is turned off, the system becomes

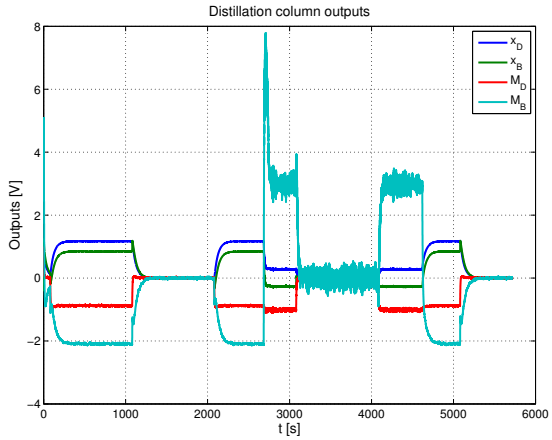


Fig. 20. Plant outputs for a stabilizing WCN configuration. Node v_1 has been turned off at time $t = 1680\ s$ and turned back on at $t = 4560\ s$.

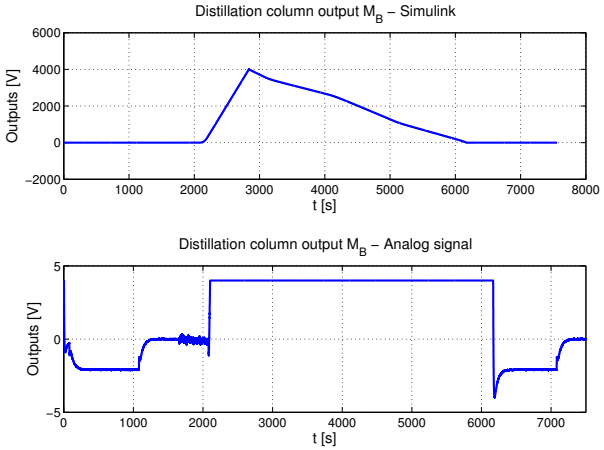


Fig. 21. Distillation column output M_B. Node v_4 has been turned off at $t = 2140\ s$ and back on at $t = 2860\ s$. Top - Simulink signal; bottom - analog signal, saturated at 4V.

unstable (shown in Fig. 21 - after the node is turned back on, the system slowly, due to the output saturation, returns to stability). Finally, we showed that if a node was added, connected to actuator a_2, sensor s_4 and nodes v_2, v_4, we could maintain stability if one of the node fails.

We also considered optimal WCN design that minimizes effects of disturbance inputs F, z_F. Using Algorithm 1 we computed an optimal WCN configuration for energy to peak minimization. The obtained measurements for a setup with periodical F impulses are shown in Fig. 22. Fig. 22(b) and Fig. 22(a) present the plant outputs for the optimal and stable WCN configurations. As shown in Fig. 22(c), the norm of the output controlled with the optimal configuration is almost 5 times smaller than the norm with the stabilizing WCN.

XV. Conclusion

This paper presents an initial stab at a problem that unravels series of difficulties at the heart of networked Cyber-Physical Systems. We have investigated several fundamental challenges with the use of wireless networks for time-critical closed-loop control problems. Wireless Networked Cyber-Physical Systems are fundamentally constrained by the tight coupling and closed-loop control of physical processes.

Unlike standard control approaches that statically map a set of tasks to a specific physical node at design time, to deal with the inherit unreliability of wireless nodes and links, for time-critical and safety-critical applications we proposed programming abstractions where control functionalities are assigned to a group of wireless nodes as a single component. Furthermore, by providing ***composable*** distributed control schemes and architectures, we have been able to harness the benefits of the use of wireless networks and to design modular, 'plug-n-play' control systems.

Our first approach, Embedded Virtual Machine (EVM), was to build the networking infrastructure to maintain state across physical node boundaries, *allowing tasks to be decoupled from the underlying unreliable physical substrate*. We presented a modular architecture used for control applications in wireless sensor/actuator/controller networks that allows component integration and system reconfiguration at runtime, without any negative effects on the execution of already assigned functionalities. The EVM enables a simple transition from the controller design in widely used simulation tools to the actual, physical 'plug-and-play' deployment for wireless networks.

Our second approach was the Wireless Control Network (WCN), where the network itself acts as a *fully distributed controller*. We have first addressed the WCN synthesis problem to guarantee *optimal* performance of the plant with respect to standard cost functions. Second, by including the observer style updates in the simple, linear iterative procedure, we have been able to significantly increase *robustness* of the closed-loop system to link failures. We have also proposed a method to extract a stabilizing configuration for the WCN that can deal with node failures. Finally, we have extended the synthesis procedure to deal with continuous-time plants, and demonstrated how the WCN can be used on an industrial application, using a process-in-the-loop setup with real hardware. In future, we aim to introduce complex control operations (e.g., Kalman filtering, model predictive control) and investigate heterogeneous nodes with varied computation/communication capabilities. Distributed control over networked CPS is a challenging problem with widespread application.

XVI. Acknowledgements

We wish to thank George Pappas and Shreyas Sundaram for fruitful discussions. The authors would also like to thank Paul McLaughlin and Alex Chernoguzov from Honeywell Process Solutions for their support and feedback. We are grateful to the reviewers for very valuable comments that were essential in improving the paper. This work builds on our efforts in [6], [19], [20], [22], [23], [24], [70], [71], [72] and [73].

References

[1] Nielsen Research, *Downtime Costs Auto Industry*, 2006.
[2] A. Willig, K. Matheus, and A. Wolisz, "Wireless technology in industrial networks," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1130–1151, 2005.
[3] W. Zhang and M. Branicky, "Stability of networked control systems with time-varying transmission period," in *Allerton Conference on Communication, Control, and Computing*, 2001.
[4] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE, Special Issue on Technology of Networked Control Systems*, vol. 95, no. 1, pp. 138–162, 2007.
[5] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHART Networks," in *31st IEEE Real-Time Systems Symposium*, 2010, pp. 150 –159.
[6] M. Pajic and R. Mangharam, "Embedded virtual machines for robust wireless control and actuation," in *RTAS '10: Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010, pp. 79–88.
[7] R. Alur, A.D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss, "Compositional modeling and analysis of multi-hop control networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2345–2357, 2011.
[8] G. Fiore, V. Ercoli, A. Isaksson, K. Landernäs, and M. D. Di Benedetto, "Multi-hop Multi-channel Scheduling for Wireless Control in WirelessHART Networks," in *IEEE Conference on Emerging Technology & Factory Automation*, 2009, pp. 1 – 8.
[9] A. D'Innocenzo, G. Weiss, R. Alur, A. Isaksson, K. Johansson, and G. Pappas, "Scalable scheduling algorithms for wireless networked control systems," in *CASE'09: IEEE International Conference on Automation Science and Engineering*, 2009, pp. 409–414.
[10] M. Pajic and R. Mangharam, "Embedded virtual machines for robust wireless control and actuation," in *RTAS'10: 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010, pp. 79–88.
[11] S. Graham, G. Baliga, and P. Kumar, "Abstractions, architecture, mechanisms, and a middleware for networked control," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1490–1503, 2009.
[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, 2000.
[13] H. Kopetz and G. Bauer, "The Time-Triggered Architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.
[14] R. Alur, A. D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss, "Modeling and analysis of multi-hop control networks," in *RTAS '09: Proceedings of the 2009 15th IEEE Symposium on Real-Time and Embedded Technology and Applications*, 2009, pp. 223–232.
[15] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, 2004.
[16] C. Robinson and P. Kumar, "Optimizing controller location in networked control systems with packet drops," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 661–671, 2008.
[17] P. Jalote, *Fault tolerance in distributed systems*. Prentice-Hall, Inc., 1994.
[18] P. A. Lee and T. Anderson, *Fault Tolerance - Principles and Practice*, J. C. Laprie, A. Avizienis, and H. Kopetz, Eds. Springer Verlag, 1990.

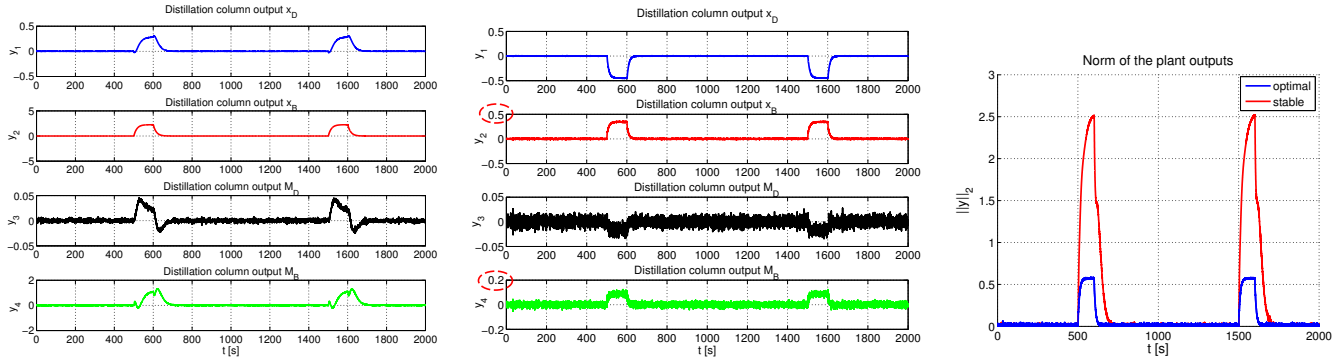


Fig. 22. Distillation column outputs; (a) For a stable WCN configuration; (b) For an optimal WCN configuration (note the axes scales); (c) Comparison of the output vector norms for the stable and the optimal WCN configurations.

[19] M. Pajic, A. Chernoguzov, and R. Mangharam, "Robust Architectures for Embedded Wireless Network Control and Actuation," *ACM Transactions on Embedded Computing Systems*, vol. 11, no. 4, pp. 82:1–82:24, 2012.

[20] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam, "The Wireless Control Network: A New Approach for Control over Networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2305–2318, 2011.

[21] C. N. Hadjicostis and R. Touri, "Feedback control utilizing packet dropping network links," in *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002, pp. 1205–1210.

[22] M. Pajic, R. Mangharam, G. J. Pappas, and S. Sundaram, "Topological Conditions for In-Network Stabilization of Dynamical Systems," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 794–807, 2013.

[23] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam, "Topological Conditions for Wireless Control Networks," in *Proceedings of the 50th IEEE Conference on Decision and Control*, 2011, pp. 2353–2360.

[24] S. Sundaram, M. Pajic, C. Hadjicostis, R. Mangharam, and G. Pappas, "The Wireless Control Network: Monitoring for malicious behavior," in *Proceedings of the 49th IEEE Conference on Decision and Control*, 2010, pp. 5979–5984.

[25] P. Levis and D. Culler, "Maté: a tiny virtual machine for sensor networks," *SIGARCH Computer Architecture News*, vol. 30, no. 5, pp. 85–95, 2002.

[26] P. Stanley-Marbell and L. Iftode, "Scylla: A smart virtual machine for mobile embedded systems," in *WMCSA '00: Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, 2000, pp. 41–50.

[27] R. Müller, G. Alonso, and D. Kossmann, "A virtual machine for sensor networks," in *EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, 2007, pp. 145–158.

[28] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, "A dynamic operating system for sensor nodes," in *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 163–176.

[29] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 455–462.

[30] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 563–579, 2005.

[31] K. Lorincz, B.-r. Chen, J. Waterman, G. Werner-Allen, and M. Welsh, "Resource aware programming in the Pixie OS," in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 211–224.

[32] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks," in *Proceedings of the 7th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN'08, 2008, pp. 233–244.

[33] M. Brown, S. Gilbert, N. Lynch, C. Newport, T. Nolte, and M. Spindel, "The Virtual Node Layer: A programming abstraction for wireless sensor networks," *SIGBED Review*, vol. 4, no. 3, pp. 7–12, 2007.

[34] R. Newton, G. Morrisett, and M. Welsh, "The regiment macroprogramming system," in *Proceedings of the 6th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN'07, 2007, pp. 489–498.

[35] R. Gummadi, O. Gnawali, and R. Govindan, "Macro-programming wireless sensor networks using Kairos," in *Distributed Computing in Sensor Systems*. Springer Berlin, 2005, pp. 126–140.

[36] K. Gatsis, M. Pajic, A. Ribeiro, and G. J. Pappas, "Power-aware communication for wireless sensor-actuator systems," in *Proceedings of the 52th IEEE Conference on Decision and Control*, 2013.

[37] V. Gupta, A. F. Dana, J. Hespanha, R. M. Murray, and B. Hassibi, "Data transmission over networks for estimation and control," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1807–1819, 2009.

[38] M. Pajic, S. Sundaram, and G. J. Pappas, "Stabilizability over Deterministic Relay Networks," in *Proceedings of the 52th IEEE Conference on Decision and Control*, 2013.

[39] E. K. Conklin and E. D. Rather, *FORTH Programmer's Handbook*. FORTH Inc, 2007.

[40] M. Pajic and R. Mangharam, "Embedded virtual machines," University of Pennsylvania, Tech. Rep., Sept. 2009.

[41] "Simulink documentation, MathWorks," 2012.

[42] nanoRK, "Sensor RTOS - http://www.nanork.org," 2010.

[43] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A global time-synchronized link protocol for sensor networks," *Ad Hoc Networks*, vol. 6, no. 8, pp. 1201–1220, 2008.

[44] A. Schrijver, " Theory of Linear and Integer Programming," *John Wiley & sons*, 1998.

[45] " HART Field Communication Protocol Specification, Rev 7," 2007.

[46] A. Cervin, J. Eker, B. Bernhardsson, and K. E. Arzen, "Feedback feedforward scheduling of control tasks," *Real-Time System Journal*, vol. 23, no. 1-2, pp. 25–53, 2002.

[47] Z. Fu, Y. Mahajan, and S. Malik, "New Features of SAT'04 version of zChaff," in *The International Conference on Theory and Applications of Satisfiability Testing*, 2004.

[48] T. Bhme, F. Gring, and J. Harant, "Menger's Theorem," *Journal of Graph Theory, vol. 37*, vol. 31, no. 1, pp. 35–36, 2001.

[49] B. Yang, S. Zheng, and E. Lu, "Finding two disjoint paths in a network with α^+-min-sum objective function," *Algorithms and Computation, Lecture Notes in Computer Science*, pp. 954–963, 2005.

[50] J. Liu, *Real-Time Systems*. Prentice Hall, Inc., 2000.

[51] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham, "Mode change protocols for priority-driven preemptive scheduling," *Real-Time Systems Journal*, vol. 1, no. 3, pp. 126–140, 1989.

[52] J. Real and A. Crespo, "Mode change protocols for real-time systems: a survey and a new proposal," *Real-Time Systems Journal*, vol. 26, no. 2, pp. 161–197, 2004.

[53] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: A Cross-layer Platform for Real-time Embedded Wireless Networks," *Real-Time System Journal*, vol. 37, no. 3, pp. 183–231, 2007.

[54] "EVM website - http://mlab.seas.upenn.edu/evm," 2009.

[55] D. R. Lewin, *Using Process Simulators in Chemical Engineering: A Multimedia guide for the Core Curriculum*. Wiley, 2009.

[56] D. Prett and M. Morari, "The shell process control workshop," *Butterworths*, 1986.

[57] P. Seiler and R. Sengupta, "Analysis of communication losses in vehicle control problems," in *Proceedings of the American Control Conference*, 2001, pp. 1491–1496.

[58] N. Elia, "Remote stabilization over fading channels," *Systems & Control Letters*, vol. 54, no. 3, pp. 237–249, 2005.

[59] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN'10, 2010, pp. 151–161.
[60] "Why WirelessHART? HART communication foundation," White Paper, 2007.
[61] "ISA100.11a: Wireless systems for industrial automation, process control and related applications," Standard, 2009.
[62] R. E. Skelton, T. Iwasaki, and K. Grigoriadis, *A unified algebraic approach to linear control design*. CRC Press, 1998.
[63] J. Han and R. Skelton, "An LMI optimization approach for structured linear controllers," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003, pp. 5143–5148.
[64] L. El Ghaoui, F. Oustry, and M. Ait Rami, "A cone complementarity linearization algorithm for static output-feedback and related problems," *IEEE Transactions on Automatic Control*, vol. 42, no. 8, pp. 1171–1176, 1997.
[65] P. Antsaklis and A. Michel, *Linear Systems*. McGraw Hill, 1997.
[66] K. S. Pister and L. Doherty, "Tsmp: Time synchronized mesh protocol," in *International Symposium on Distributed Sensor Networks (DSN)*, 2008, pp. 391–398.
[67] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Wiley, 1996.
[68] "Real-Time Windows Target - Run Simulink models on a PC in real time. http://www.mathworks.com/products/rtwt. MathWorks."
[69] "CVX: Matlab Software for Disciplined Convex Programming, version 2.0, http://cvxr.com/cvx. CVX Research, Inc." 2012.
[70] M. Pajic, S. Sundaram, J. Le Ny, G. J. Pappas, and R. Mangharam, "The Wireless Control Network: Synthesis and Robustness," in *Proceedings of the 49th IEEE Conference on Decision and Control*, 2010, pp. 7576–7581.
[71] M. Pajic, S. Sundaram, G. Pappas, and R. Mangharam, "Network synthesis for dynamical system stabilization," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2011, pp. 821–825.
[72] M. Pajic, S. Sundaram, J. Le Ny, G. J. Pappas, and R. Mangharam, "Closing the loop: A simple distributed method for control over wireless networks," in *Proceedings of the 11th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN'12, 2012, pp. 25–36.
[73] F. Miao, M. Pajic, R. Mangharam, and G. J. Pappas, "Mapping Discrete-Time Controllers into Structured Computational Substrate," in *American Control Conference*, 2013, pp. 3002–3007.

Rahul Mangharam (M'02) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2000, 2002, and 2008 respectively.

He is the Stephen J Angello Chair and Assistant Professor in the Dept. of Electrical & Systems Engineering and Dept. of Computer & Information Science at the University of Pennsylvania. He is the Director of the Real-Time and Embedded Systems Lab. His current interests are in real-time scheduling and control algorithms for networked embedded systems with applications in automotive systems, medical devices, energy-efficient buildings and wireless control networks.

Dr. Mangharam received the 2013 NSF CAREER Award, 2012 Intel Early Faculty Career Award and was selected by the National Academy of Engineering for the 2012 US Frontiers of Engineering.

Miroslav Pajic (S'06) received the Dipl. Ing. and M.S. degrees in electrical engineering from the University of Belgrade, Serbia, in 2003 and 2007, respectively, and the M.S. and Ph.D. degrees in electrical engineering from the University of Pennsylvania, Philadelphia, in 2010 and 2012, respectively.

He is a Postdoctoral Fellow in the Department of Electrical & Systems Engineering at the University of Pennsylvania. His research interests include cyber-physical systems, embedded and distributed/networked control systems, real-time and embedded systems, and high-confidence medical device systems.

Dr. Pajic received several awards including 2011 ACM SIGBED Frank Anger Memorial Award, and the Best Student Paper award at the 2012 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). He was awarded the University of Pennsylvania, Joseph and Rosaline Wolf Award for the Best Dissertation in 2013.