The Wireless Control Network: A New Approach for Control Over Networks

Miroslav Pajic, *Student Member, IEEE*, Shreyas Sundaram, *Member, IEEE*, George J. Pappas, *Fellow, IEEE*, and Rahul Mangharam, *Member, IEEE*

***Abstract*—We present a method to stabilize a plant with a network of resource constrained 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 treats the network itself as the controller. Specifically, we formulate a strategy for each node in the network to follow, where at each time-step, each node updates its internal state to be a linear combination of the states of the nodes in its neighborhood. We show that this causes the entire network to behave as a linear dynamical system, with sparsity constraints imposed by the network topology. We provide a numerical design procedure to determine appropriate linear combinations to be applied by each node so that the transmissions of the nodes closest to the actuators will stabilize the plant. We also show how our design procedure can be modified to maintain mean square stability under packet drops in the network, and present a distributed scheme that can handle node failures while preserving stability. We call this architecture a Wireless Control Network, and show that it introduces very low computational and communication overhead to the nodes in the network, allows the use of simple transmission scheduling algorithms, and enables compositional design (where the existing wireless control infrastructure can be easily extended to handle new plants that are brought online in the vicinity of the network).**

***Index Terms*—Cooperative control, decentralized control, linear systems, networked control systems, wireless sensor networks.**

I. Introduction

INDUSTRIAL control systems are often deployed in large, spatially distributed plants that involve numerous sensors, actuators and internal process variables. The traditional means of interconnecting the various components of these systems has been via physical wires; this is often difficult to do (in hard-to-reach or dangerous areas), expensive (due to the labor and materials involved) and fault-prone (due to degradation of wires, miswiring due to human error, etc.). Over the last decade, low-cost and reliable wireless networks have emerged as a practical method to alleviate these issues in automation systems [2], [3]. Besides the obvious physical benefit of reducing excessive wiring, these networks introduce a set of *logical* benefits [4]. For example, wireless communication allows one to "hot-swap" between a faulty module and a backup module via a simple activation command, and facilitates "plug-n-play" automation architectures which reduce downtime.[1] Along the same lines, through the use of certain algorithms that satisfy the principle of *compositionality* (where the existing setup can be easily extended for increased functionality and robustness), wireless networks make it possible to incrementally upgrade systems in a straightforward manner.

Wireless networks also pose some interesting new challenges. One problem arises due to the fundamental unreliability of wireless communication; the probability that a wireless transmission will be received by a node's neighbors depends on various factors, including the amount of power used to transmit information, environmental conditions that affect the propagation characteristics of the channel, and collisions that might occur due to multiple nodes transmitting at the same time. Another problem is the need to maintain a reasonable end-to-end delay in the network. The topic of designing controllers that are tolerant to these types of issues has been intensively studied by researchers over the past decade [7]–[12]. The vast majority of work in this area considers the case of a single sensing point and a single actuation point on the plant, and adopts the convention of having a dedicated controller/estimator located somewhere in the network. The stability of the closed loop system is then studied, assuming that the sensor-estimator and/or controller-actuator communication channels are unreliable (e.g., dropping packets with a certain probability). While these and other works have made great inroads into understanding the problem of feedback control over networks, they have some potential drawbacks when it comes to implementation. First, the state vectors maintained by the controllers in these existing works typically have sizes on the order of the size of the plant's state vector, and intermediate nodes are also expected to perform operations on state vectors of similar size. However, devices in wireless networks are often battery operated and have severe resource constraints, allowing only a modest amount of computation and storage (examples of this are discussed in Section III). Also, by adding one (or a few) specialized nodes capable of performing computationally expensive procedures, the control infrastructure becomes susceptible to failure on the

Manuscript received April 02, 2010; accepted July 11, 2011. Date of publication August 08, 2011; date of current version October 05, 2011. This paper was presented in part at the 49th IEEE Conference on Decision and Control, Atlanta, GA, Dec. 2010. This work was supported in part by the DARPA MuSyC, the NSF-CNS 0931239 and NSF-MRI Grant 0923518, and a grant from NSERC. Recommended by Associate Editor K. H. Johansson.

M. Pajic, G. J. Pappas, and R. Mangharam are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA 19014 (e-mail: pajic@seas.upenn.edu; pappasg@seas.upenn.edu; rahulm@seas.upenn.edu).

S. Sundaram is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: ssundara@uwaterloo.ca).

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

Digital Object Identifier 10.1109/TAC.2011.2163864

[1] In traditional wired automation systems, modules are connected with an industrial bus (e.g., PROFIBUS [5], CAN [6]) and a large number of I/O connectors, where the wiring is usually inaccessible, which significantly increases the time and cost needed for replacements.

part of those nodes. A second drawback of these traditional approaches is that they do not capture the real-world scenario of multiple sensing and actuation points that are geographically dispersed throughout the plant, with signals that are injected into and out of different nodes throughout the network.

In order to minimize end-to-end delays in the network and to maximize the network lifetime, one typically uses *wireless link protocols* which broadly fall into two categories: synchronous or asynchronous. Asynchronous protocols have the advantage of not requiring a communication schedule between nodes, since a node simply transmits its packet as soon as it is received. However, this simplicity results in large communication jitter which makes end-to-end timing analysis very difficult in multi-hop scenarios [13], and complicates the task of characterizing the stability of the system.

Time synchronized network protocols are the norm in the control automation industry and recent standards (such as WirelessHART [2] and ISA 100.11a [14]) employ a time division multiplexing link protocol. Full network synchronization allows the use of Time-Triggered Architectures (TTAs) where communication and computation are scheduled at particular instances of time (i.e., time slots) [15]. From the perspective of analyzing system stability, TTAs have the advantage that the network-induced delay is known. Furthermore, a closed-loop system based on a TTA can be modeled as a switched control system [16], which allows utilization of existing techniques for switched-system analysis. However, in this case the system performance depends on communication and computation schedules that have to be carefully designed and interleaved on a node-by-node basis. Even in the case with only one plant being controlled over a multi-hop network, the task of constructing these schedules in order to meet strict end-to-end delay requirements is very complex [16], [17].

The goal of this paper, which expands on the work presented in [1], is to introduce a new way of looking at the problem of control over wireless networks. Instead of assigning the computation of the control law to a particular node in the network, we show how to cause the entire network *itself* to act as the controller. We call this fully-distributed paradigm the *Wireless Control Network* (WCN). To develop this idea, we consider a setup where several resource constrained wireless nodes are deployed in the proximity of a plant, with some nodes having access to the sensor measurements (i.e., outputs) of the plant, and some nodes placed within the listening range of the plant's actuators. Each node in the network is capable of maintaining only a limited internal state. Given this setup, we present a simple linear iterative strategy for each node to follow, where each node periodically updates its state to be a linear combination of the states of the nodes in its immediate neighborhood. In addition, the plant actuators apply linear combinations of the states of the nodes in their neighborhood. The key insight of our work is that this simple scheme causes the entire network itself to behave as a structured dynamical compensator. Based on this insight, we adapt numerical algorithms from the literature on structured and static output feedback control design to synthesize the stabilizing linear combinations employed by each node and actuator. We also show how this design procedure can be modified to account for packet drops in the network. As discussed in Section III, this approach to control over a wireless network has many benefits over traditional schemes (where information is routed to and from a dedicated controller). Specifically, the WCN requires very little overhead, is very simple to schedule, is capable of handling plants with dispersed sensing and actuation points, can explicitly account for computational constraints at each node, and satisfies the principle of compositionality (allowing ease of incremental upgrades to the plant).

The rest of the paper is organized as follows. In Section II, we introduce and describe the Wireless Control Network in more detail, including the mathematical formulation that will set up the design procedures. In Section III, we list the implementation advantages of the WCN in comparison to existing networked control schemes. We then delve into the problem of synthesizing the WCN in Section IV, where we adapt existing algorithms from the literature on static output feedback to determine an appropriate set of stabilizing linear combinations. In Section V, we show how these algorithms can be modified to account for probabilistic packet drops in the network. For the sake of clarity, we assume in both of these preceding sections that each node can only perform calculations on a single (scalar) value; in Section VI, we generalize our discussion to vector states at each node. We provide examples of our scheme in Section VII. In Section VIII we show how the WCN is able to gracefully degrade under node failures, and we discuss the relationship between the WCN and the traditional notions of delay introduced by the feedback loop. Finally, we finish in Section IX by describing some possible directions for future improvements on this scheme.

Notation

We use $\mathbf{e}_i$ to denote the i^{th} unit column vector (of appropriate dimension) and the symbol $\mathbf{1}$ denotes the column vector (of appropriate size) consisting of all 1's. The symbol $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. The notation $\mathrm{diag}\,(\cdot)$ indicates a square matrix with the quantities inside the brackets on the diagonal, and zeros elsewhere. The notation $\mathrm{tr}\,(\cdot)$ indicates the trace of a square matrix. We will denote the cardinality of a set $\mathcal{S}$ by $|\mathcal{S}|$. The set of nonnegative integers is denoted by $\mathbb{N}$. The notation $\mathbf{A} \succ \mathbf{0} (\succeq \mathbf{0})$ indicates that matrix $\mathbf{A}$ is positive (semi)definite. The set of all $n \times n$ positive definite matrices is denoted by $\mathbb{S}^n_{++}$. A graph is an ordered pair $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is a set of vertices (or nodes), and $\mathcal{E}$ is a set of ordered pairs of different vertices, called directed edges. The vertices in the set $\mathcal{N}_{v_i} = \{v_j | (v_j, v_i) \in \mathcal{E}\}$ are said to be neighbors of vertex v_i.

II. Wireless Control Network

Consider the system presented in Fig. 1, where the plant is to be controlled using a multi-hop, fully synchronized wireless network. In this paper we focus on plants of the form[2]

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

[2]The plant model can be generalized to include update and measurement noise; if the noise is taken to be independent and identically distributed with a bounded variance, all of our analysis and results will still ensure that the system is bounded in a mean square sense. For the purposes of clarity, we will therefore omit the noise terms in our discussion.

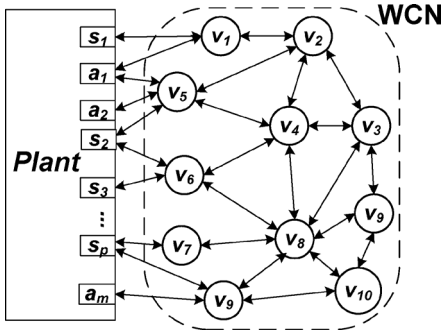


Fig. 1. Multi-hop wireless control network used as a distributed controller.

with $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$ and $\mathbf{C} \in \mathbb{R}^{p\times n}$. The output vector $\mathbf{y}[k] = [\, y_1[k]\ y_2[k]\ \ldots\ y_p[k]\,]^T$ contains measurements of the plant state vector $\mathbf{x}[k]$ provided by the sensors $s_1, s_2, \ldots, s_p$. The input vector $\mathbf{u}[k] = [\, u_1[k]\ u_2[k] \ldots\ u_m[k]\,]^T$ corresponds to the signals applied to the plant by actuators $a_1, a_2 \ldots, a_m$.

The WCN consists of a set of nodes that communicate with each other and with the sensors and actuators installed on the plant. Each node in the network is equipped with a radio transceiver along with (limited) memory and computational capabilities.[3] Similarly, each sensor and actuator on the plant contains a radio transceiver, allowing them to communicate with neighboring nodes. The wireless network is described by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ 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). We also define $\mathcal{V}_S \subset \mathcal{V}$ as the set of nodes that can receive information directly from at least one sensor, and $\mathcal{V}_A \subset \mathcal{V}$ as the set of nodes whose transmissions can be heard by at least one actuator.

To facilitate our development, we will find it convenient to consider a new graph $\bar{\mathcal{G}}$ that includes the plant's sensors and actuators. This graph is obtained by taking the graph $\mathcal{G}$ and adding $p + m$ new vertices $\mathcal{S} \cup \mathcal{A}$, where $\mathcal{S} = \{s_1, s_2, \ldots, s_p\}$ corresponds to the plant's sensors, while $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ corresponds to the plant's actuators. Define the edge sets

$$\mathcal{E}_O = \Big\{ (s_l, v_i) \;\Big|\; \begin{array}{c} s_l \in \mathcal{S}, v_i \in \mathcal{V}_S, \\ v_i \text{ can receive values from sensor } s_l \end{array} \Big\}$$

$$\mathcal{E}_I = \Big\{ (v_i, a_l) \;\Big|\; \begin{array}{c} a_l \in \mathcal{A}, v_i \in \mathcal{V}_A, \\ a_i \text{ can receive values from } v_i \end{array} \Big\}.$$

We then obtain $\bar{\mathcal{G}} = \{\mathcal{V} \cup \mathcal{S} \cup \mathcal{A}, \mathcal{E} \cup \mathcal{E}_I \cup \mathcal{E}_O\}$.

Unlike traditional networked control schemes where a particular node $v_i \in \mathcal{V}$ is designated as the controller and all other nodes are used to route information between v_i and the plant, we propose a fully distributed control scheme where the entire network itself acts as a controller (becoming a Wireless Control Network). To achieve this, we have each node in the network utilize a linear iterative strategy where, at each time-step, it updates its value to be a linear combination of its previous value and the values of its neighbors.[4] In addition, the update procedure of each node from the set $\mathcal{V}_S$ includes a linear combination of the sensor measurements (i.e., plant outputs) from all sensors in its neighborhood. If we let $z_i[k]$ denote node v_i's (scalar) state at time step k, we obtain the update procedure:[5]

$$z_i[k+1] = w_{ii} z_i[k] + \sum_{v_j \in \mathcal{N}_{v_i}} w_{ij} z_j[k] + \sum_{s_j \in \mathcal{N}_{v_i}} h_{ij} y_j[k]. \quad (2)$$

Each plant input $u_i[k], i \in \{1, \ldots, m\}$ is taken to be a linear combination of values from the nodes in the neighborhood of the actuator a_i:[6]

$$u_i[k] = \sum_{j \in \mathcal{N}_{a_i}} g_{ij} z_j[k]. \quad (3)$$

Remark 1: Note that the time-step k of the network in the above updates is the same as the time-step of the plant. This is in contrast to the typical paradigm of control over networks, where the sampling rate of the plant must be slow enough that information can be routed within the wireless network without much of an impact on the closed-loop system. This paradigm automatically enforces that the nodes in the network operate at a faster rate than the plant. However, the above WCN algorithm only requires each node to operate on information from one-hop neighbors. Thus, we can either slow the network nodes down to the sampling rate of the plant (potentially obtaining benefits in reliability, efficiency, etc.), or speed up the sampling rate of the plant to match the rate of the network (potentially yielding better control performance). □

The scalars w_{ij}, h_{ij} and g_{ij} specify the linear combinations that are computed by each node and actuator in the network. If we aggregate the values of all nodes at time step k into the value vector $\mathbf{z}[k] = [\, z_1[k]\ z_2[k]\ \ldots\ z_N[k]\,]^T$, the behavior of the entire network can be represented as

$$\begin{aligned} \mathbf{z}[k+1] &= \mathbf{W}\mathbf{z}[k] + \mathbf{H}\mathbf{y}[k], \\ \mathbf{u}[k] &= \mathbf{G}\mathbf{z}[k] \end{aligned}$$

for all $k \in \mathbb{N}$ ($\mathbf{W} \in \mathbb{R}^{N\times N}, \mathbf{H} \in \mathbb{R}^{N\times p}, \mathbf{G} \in \mathbb{R}^{m\times N}$). In the above equation, for all $i \in \{1, \ldots, N\}$, $w_{ij} = 0$ if $v_j \notin \mathcal{N}_{v_i} \cup \{v_i\}$, $h_{ij} = 0$ if $s_j \notin \mathcal{N}_{v_i}$, and $g_{ij} = 0$ if $v_j \notin \mathcal{N}_{a_i}$. Thus, the matrices $\mathbf{W}, \mathbf{H}$ and $\mathbf{G}$ are *structured*, meaning that they have sparsity constraints determined by the topology of the WCN. 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. The above representation leads to our first key observation: the linear strategy employed by the nodes causes the network to act as a *structured dynamical*

[3]We will model these resource constraints by limiting the size of the state vector that can be maintained by each node. To present our idea, we will initially focus on the case where each node's state is represented as a scalar. The more general case, where each node can maintain a vector state with possibly different dimensions, is considered in Section VI.

[4]The proposed scheme is similar in flavor to the algorithms used in linear network coding [18], where nodes in a network mathematically combine packets before transmitting them. In our case, the objective is to choose these linear combinations to stabilize the plant, rather than simply transmit information through the network.

[5]The neighborhood $\mathcal{N}_v$ of a vertex v is with respect to the graph $\bar{\mathcal{G}}$.

[6]Here we assume that each actuator, in addition to having a radio transceiver, has computational capabilities to be able to calculate the weighted sum of its neighboring nodes' states. However, in cases where an actuator is equipped only with a transceiver, the state of only one node in the actuator's neighborhood is used by that actuator. Thus, in this case for each $i \in \{1, \ldots, p\}$, $g_{ij} = 1$ for exactly one $j \in \{1, \ldots N\}$, and all other weights are equal to zero.

compensator. If we denote the overall system state by $\hat{\mathbf{x}}[k] = [\,\mathbf{x}[k]^T \quad \mathbf{z}[k]^T\,]^T$, the closed-loop system becomes

$$\hat{\mathbf{x}}[k+1] = \begin{bmatrix} \mathbf{A} & \mathbf{BG} \\ \mathbf{HC} & \mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{x}[k] \\ \mathbf{z}[k] \end{bmatrix} \triangleq \hat{\mathbf{A}}\hat{\mathbf{x}}[k]. \tag{4}$$

Remark 2: The WCN is a fully distributed controller, implemented upon a substrate provided by a wireless network. One of the key differences between the WCN and classical decentralized control approaches is that the latter typically assume that each controller has direct access to some of the plant inputs and outputs [19]. In contrast, if we view each node in the WCN as a separate controller, many of these controllers will not be directly connected to the plant, but only to other nodes (or controllers). Furthermore, we enforce size constraints on the state of each controller, and study how to leverage the interconnections between the nodes in order to stabilize the system. □

In the next section, we describe the implementation advantages of the above control mechanism, and then we present algorithms to find an element of Ψ that causes the closed loop system to be stable.

III. Advantages of the Wireless Control Network

With the mathematical description of the WCN from the previous section in hand, we will now discuss some advantages of this architecture in the context of multi-hop embedded wireless networks for control.

A. Low Overhead

The proposed scheme is computationally inexpensive since each node only needs to compute a linear combination of its value and values of its neighbors. Thus, the WCN can be easily implemented even on resource constrained, low-power wireless nodes (such as those shown in Fig. 2),[7] using very simple, periodic tasks executed on a real-time operating system (such as nano-RK [21] or TinyOS [22]). Furthermore, unlike traditional networked control schemes, our approach can explicitly account for these computational and resource constraints during the design procedure (i.e., by limiting the size of the state vector maintained by each node).

In addition, as the only requirement of the scheme is that a node transmits its state once per time-step (also known as a *frame* in the wireless networking literature), the proposed scheme can be easily "piggy-backed" into wireless networks that already assign a transmission slot for each node to maintain network related information (e.g., wireless systems for factory automation based on the ISA100.11a standard [14] or wirelessHART [2]). For example, if the node's state $z_i[k]$ is a 16-b scalar, each node only needs to transmit 2 additional bytes per frame in order to control the system, which can be easily accommodated in each transmitted message. This also allows the possibility of using the proposed scheme as a backup (fault-tolerant) mechanism in traditional networked control systems. Specifically, if the primary control mechanism (i.e., dedicated controller) in the existing networked control infrastructure fails, the wireless network itself can take over the role of stabilizing the plant (i.e., operate as a WCN) until the functionality of the primary controller is restored.

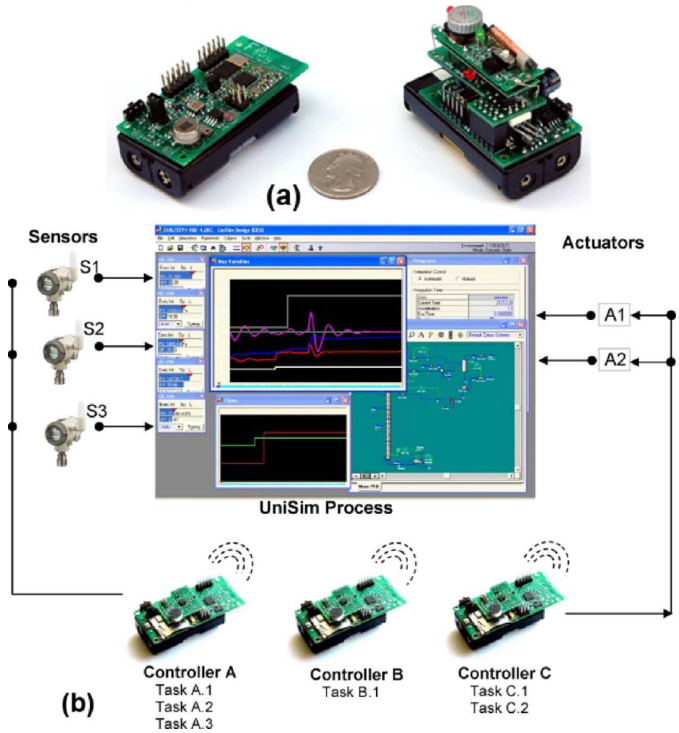


Fig. 2. (a) On the left of the coin, a low-power FireFly node; on the right, a FireFly node with an add-on AM receiver for hardware-based out-of-band synchronization; (b) An example of FireFly nodes in a process-in-the-loop simulation using the Honeywell Unisim process (plant) modeling tool.

The requirement that all nodes in the WCN be at least loosely synchronized can be easily accomplished using in-band synchronization as a part of a TDMA-based Medium Access Control (MAC) protocol (i.e., nodes implicitly synchronize with another nodes using the current transmission slot, as in RT-Link [23] and wirelessHART [24]) or with hardware synchronization (e.g., using an additional AM receiver, as shown in Fig. 2(a) [23]).[8]

B. Simple Scheduling

The presented scheme does not require complex communication scheduling, since each node needs to transmit exactly once in a frame and the WCN does not impose end-to-end delay constraints (i.e., nodes close to the actuators do not need to wait for information to propagate all the way from the sensors). The only requirement of the communication schedule is to be conflict-free (i.e., two nodes within the same transmission range should not broadcast at the same time). Since the WCN does not use any routing to and from a dedicated controller, the communication schedule does not have to change when link qualities change. On the other hand, standard routing techniques require

[7]These nodes usually contain an 8-b or 16-b microcontroller operating on 8 MHz (or lately 16 MHz) clock, with up to 16 KB of RAM and a low-power radio (typically IEEE 802.15.4 compatible radio, with 250 Kbps physical layer data rate). Their power consumption is also very low; for example, the FireFly node uses 60 mW when both CPU and radio are active, or 3 μW when the CPU and radio are in sleep mode [20].

[8]Although the nodes in the network have to be synchronized, the WCN scheme can also be implemented in wireless networks that utilize asynchronous MAC protocols (e.g., B-MAC [25]). In this case, the effects of (increased) message collisions have to be taken into account while calculating the probability of message failure, since they will have negative effects on the system's stability (as shown in Section V).

a recalculation of the routes and communication schedules if the packet drop probability on one of the links increases dramatically. Thus, if d_i is the maximal degree of the interference graph,[9] a *static* conflict-free schedule can be derived using graph coloring, with d_i slots in a frame. Since the duration of a frame is equal to the plant's sampling period, the minimal sampling period of the plant is equal to $d_i T_{slot}$, where T_{slot} is the duration of a communication slot. In contrast, traditional networked control systems often impose a requirement that the sampling period be greater than the end-to-end delay, causing the minimal sampling period to directly depend on the network diameter.

C. Multiple Sensing/Actuation Points

The WCN can readily handle plants with multiple geographically distributed sensors and actuators, a case that is not easily handled by the "sensor → channel → controller/estimator → channel → actuator" setup that is commonly adopted in networked control design. Even in the few works that consider networked control over arbitrary topologies (e.g., [11], [12]), an assumption is made that there is a single actuation and a single sensing point on the plant. Under this assumption, those papers recommend placing the controller at the actuation point, so that the controller will know all of the inputs that are applied to the plant, and can thus correctly estimate the state from the information that it receives from the sensing point (via the other nodes). However, real-world plants contain multiple actuation and sensing locations, in which case it is no longer clear that the conclusions in those papers hold. Our approach, on the other hand, does not rely on the existence of dedicated controllers, and inherently captures the case of nodes exchanging values with the plant at various points in the network.

D. Compositionality

The WCN allows *compositionality*, meaning that an existing design can be easily extended to accommodate new subsystems that are added to the plant. In subsequent sections we describe how to synthesize the WCN to stabilize a given plant. However, suppose that some new subsystems (or plants) are added in the proximity of the wireless network, and the existing wireless infrastructure is to be used to control these new systems (in addition to the original plant). In traditional networked control schemes (with a dedicated controller node, and all other nodes functioning as routers), reusing the network is complicated due to several factors. First, each node would potentially have to transmit multiple times during a given frame, based on when the information reaches it from the various sensors on the different plants. This requires the calculation of a new collision-free schedule for the *entire* network.[10] Second, with each change of communication schedules, it is necessary to analyze the schedule of computations on each node to determine whether a controller assigned with the execution of one (or more) control procedure(s) can schedule and execute them in time (between packet reception and subsequent transmission) to provide outputs that are to be transmitted to actuators [16].

[9]The interference graph is defined as $\bar{\mathcal{G}}_{Int} = \{\mathcal{V} \cup \mathcal{S} \cup \mathcal{A}, \mathcal{E}_{Int}\}$, where a link between two nodes (or a node and a sensor/actuator) indicates that they can interfere with each other (i.e., cannot transmit simultaneously).

[10]Here we consider networks that utilize TDMA MAC protocols.

Compositionality is inherent in the WCN due to the fact that each node is only required to transmit once per frame (and end-to-end delay requirements do not enter into the picture). If P is the total number of plants, one can calculate a separate stabilizing set of linear combinations for each of the new plants, with corresponding separate states maintained by each node. To control all plants simultaneously, each node groups all of its P (possibly vector) states into a single transmission packet.[11] Upon reception of the P different states from each of its neighbors, each node updates its P different internal states using the appropriate linear combinations. This enables a completely decoupled computation of the matrices $\{\mathbf{W}_i, \mathbf{H}_i, \mathbf{G}_i\}_{i=1}^{P}$ that guarantee stability for each of the P plants, although physically realized by the same WCN. As controlling an additional plant does not change the communication schedule for the WCN, one can avoid the complex rescheduling of communications and computations that is inherent in traditional networked control systems [16], [17].

IV. Stabilizing the Closed-Loop System

In this section, we discuss the problem of determining the linear combinations that should be used by each node and actuator to stabilize the closed-loop system via the WCN. From (4), the closed-loop system is stable if the matrix $\hat{\mathbf{A}} = \hat{\mathbf{A}}(\mathbf{W}, \mathbf{H}, \mathbf{G})$ is Schur. The traditional approach to achieving this would be to attempt to find a positive definite matrix $\mathbf{X}$ satisfying the Lyapunov inequality $\mathbf{X} - \hat{\mathbf{A}}^T \mathbf{X} \hat{\mathbf{A}} \succ 0$, or equivalently,

$$\begin{bmatrix} \mathbf{X} & \hat{\mathbf{A}}^T \mathbf{X} \\ \mathbf{X}\hat{\mathbf{A}} & \mathbf{X} \end{bmatrix} \succ 0.$$

The above condition is not linear in the design parameters $\mathbf{X}, \mathbf{W}, \mathbf{H}, \mathbf{G}$; this is of no consequence in standard controller design (when there are no structural constraints on the design matrices), because this condition can be converted to a Linear Matrix Inequality (LMI) via an appropriate transformation of the system matrices (e.g., as done in [26]). However, the fact that the matrices are *structured* in our framework prevents us from directly applying these standard procedures.[12] While this direct attempt to cast the controller design problem as a LMI does not work in our context, we note that problems of the above form appear in the design of static output feedback controllers with sparsity constraints on the gain matrix [28]. Although this is a computationally difficult problem in general (e.g., various versions of this problem are NP-hard [29]–[31]), various numerical approximation algorithms have been proposed in the literature

[11]Usually this is not a severe limitation even for low-bandwidth, 802.15.4 networks (where each transmission can carry up to 1024 b). In these networks, if each plant is controlled using the scheme where a node maintains a scalar 16-b state value, then up to 64 plants can be controlled in parallel.

[12]For matrices that have particular structures (such as being block diagonal), a common approach is to consider $\mathbf{X}$ to be block diagonal, which would maintain the structure of the design matrices after the linearization [27]. However, for the (arbitrary) network topologies that we study in this paper, our experiments show that this approach is overly conservative and fails to find feasible solutions even when they exist.

(e.g., [32]–[34]). We now describe one such procedure that can be used to design the WCN, and later we will show how this procedure can be modified to deal with unreliable communication links in the network. We start with the following characterization of stability of structured systems from [35].

Theorem 1: ([35]) A matrix $\hat{\mathbf{A}}$ is Schur if and only if there exist symmetric, positive-definite matrices $\mathbf{X}$ and $\mathbf{Y}$ such that

$$\begin{bmatrix} \mathbf{X} & \hat{\mathbf{A}}^T \\ \hat{\mathbf{A}} & \mathbf{Y} \end{bmatrix} \succ 0,\ \mathbf{X} = \mathbf{Y}^{-1}.$$

□

The above theorem provides a matrix inequality that is *linear* in the design variables $\mathbf{W}, \mathbf{G}$ and $\mathbf{H}$, but suffers from the fact that the constraint $\mathbf{X} = \mathbf{Y}^{-1}$ is nonconvex. One appealing approach to deal with this was suggested in [32], [33], where the constraint $\mathbf{X} = \mathbf{Y}^{-1}$ is approximated with an optimization problem via the following lemma.

Lemma 1: Positive-definite matrices $\mathbf{X}, \mathbf{Y}$ satisfy the constraint $\mathbf{X} = \mathbf{Y}^{-1}$ if and only if they are optimal points for the problem

$$(P_1): \quad \min tr(\mathbf{XY}),\ \ s.t.\ \mathbf{X} \succeq \mathbf{Y}^{-1},\ \ \mathbf{X}, \mathbf{Y} \in \mathbb{S}^n_{++}$$

and the optimal cost of the problem is n.

Using the Schur complement, the constraint $\mathbf{X} \succeq \mathbf{Y}^{-1}$ in the above lemma can be transformed to the form $\begin{bmatrix} \mathbf{X} & \mathbf{I} \\ \mathbf{I} & \mathbf{Y} \end{bmatrix} \succeq 0$. Theorem 1 and Lemma 1 yield the following corollary.

Corollary 1: There exist a Schur matrix $\hat{\mathbf{A}} = \hat{\mathbf{A}}(\mathbf{W}, \mathbf{H}, \mathbf{G})$ where the matrices $\mathbf{W}, \mathbf{H}$ and $\mathbf{G}$ satisfy the desired sparsity constraints ($(\mathbf{W}, \mathbf{H}, \mathbf{G}) \in \Psi$) if and only if the following optimization problem:

$$\min tr(\mathbf{XY}), \tag{5}$$

$$\begin{bmatrix} \mathbf{X} & \hat{\mathbf{A}}^T \\ \hat{\mathbf{A}} & \mathbf{Y} \end{bmatrix} \succ \mathbf{0}, \begin{bmatrix} \mathbf{X} & \mathbf{I} \\ \mathbf{I} & \mathbf{Y} \end{bmatrix} \succeq \mathbf{0}, \hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{BG} \\ \mathbf{HC} & \mathbf{W} \end{bmatrix} \tag{6}$$

$$(\mathbf{W}, \mathbf{H}, \mathbf{G}) \in \Psi,\ \mathbf{X}, \mathbf{Y} \in \mathbb{S}^{n+N}_{++} \tag{7}$$

is feasible with optimal cost $n + N$.

Note that with the exception of the objective function (5), all of the constraints in the above corollary are linear in the unknown parameters, and can readily be solved using LMI tools. As described in [32], [36], a problem of the form (P_1) from Lemma 1 (or (5)–(7) from Corollary 1) is known as a *cone complementarity problem* (CCP). For such problems, El Ghaoui *et al.* [32] showed that the nonconvex function $\mathrm{tr}(\mathbf{XY})$ can be replaced with a linear approximation

$$\phi_{lin}(\mathbf{X}, \mathbf{Y}) = constant + tr(\mathbf{Y_0 X} + \mathbf{X_0 Y})$$

for any given matrices $\mathbf{X}_0$ and $\mathbf{Y}_0$. With this insight, [32], [33] showed that an iterative algorithm can be used to minimize $tr(\mathbf{XY})$, while ensuring satisfaction of LMI constraints. For our application, the iterative approach proposed in those papers can be formulated as Algorithm 1.

Algorithm 1 Stabilizing closed-loop system with the WCN

1. Find feasible points $\mathbf{X}_0, \mathbf{Y}_0, \mathbf{W}_0, \mathbf{H}_0, \mathbf{G}_0$ that satisfy the constraints (6)-(7). If a feasible point does not exist, then it is not possible to stabilize the system with this network topology.
2. At iteration k ($k \geq 0$), from $\mathbf{X}_k, \mathbf{Y}_k$ obtain the matrices $\mathbf{X}_{k+1}, \mathbf{Y}_{k+1}, \mathbf{W}_{k+1}, \mathbf{H}_{k+1}, \mathbf{G}_{k+1}$ by solving the following LMI problem:

$$\min\ tr(\mathbf{Y_k X}_{k+1} + \mathbf{X_k Y}_{k+1})$$
$$\begin{bmatrix} \mathbf{X}_{k+1} & \hat{\mathbf{A}}^T_{k+1} \\ \hat{\mathbf{A}}_{k+1} & \mathbf{Y}_{k+1} \end{bmatrix} \succ 0,\ \begin{bmatrix} \mathbf{X}_{k+1} & \mathbf{I} \\ \mathbf{I} & \mathbf{Y}_{k+1} \end{bmatrix} \succeq 0,$$
$$\hat{\mathbf{A}}_{k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{BG}_{k+1} \\ \mathbf{H}_{k+1}\mathbf{C} & \mathbf{W}_{k+1} \end{bmatrix},$$
$$(\mathbf{W}_{k+1}, \mathbf{H}_{k+1}, \mathbf{G}_{k+1}) \in \Psi,\ \ \mathbf{X}_{k+1}, \mathbf{Y}_{k+1} \in \mathbb{S}^{n+N}_{++}.$$

3. If the matrix

$$\hat{\mathbf{A}}_{k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{BG}_{k+1} \\ \mathbf{H}_{k+1}\mathbf{C} & \mathbf{W} \end{bmatrix}_{k+1}$$

is Schur, stop the algorithm. Otherwise, set $k = k + 1$ and go to the step **2**.

In [32] the authors showed that the sequence $t_k = tr(\mathbf{Y}_k\mathbf{X}_{k+1} + \mathbf{X}_k\mathbf{Y}_{k+1})$ will always converge. In addition, if the sequence converges to $2(n + N)$ the condition $\mathbf{Y} = \mathbf{X}^{-1}$ can be satisfied under the given LMI constraints. A similar proof can be constructed in this case, which leads us to the following theorem.

Theorem 2: **Algorithm** 1 determines a tuple $(\mathbf{W}, \mathbf{H}, \mathbf{G}) \in \Psi$ that causes the matrix $\hat{\mathbf{A}}(\mathbf{W}, \mathbf{H}, \mathbf{G})$ to be Schur if the sequence $t_k = tr(\mathbf{Y}_k\mathbf{X}_{k+1} + \mathbf{X}_k\mathbf{Y}_{k+1})$ converges to $2(n + N)$. □

Note that while each iteration of the above algorithm is a convex optimization problem (which can be efficiently solved using standard LMI toolboxes), currently there is no way to characterize the number of iterations required for the algorithm to converge. This problem has attracted substantial attention in the context of static output feedback design. For example, [34] compared the convergence rates of various algorithms, and similar experiments can be run for the synthesis of the WCN. Furthermore, in [33] the authors propose a simple modification of the utilized algorithm in order to provide faster convergence. However, the convergence rate of the algorithm depends on the initial points $\mathbf{X}_0$ and $\mathbf{Y}_0$, but we do not currently have a way to pick the 'best' such initial points. Indeed, due to the computational complexity of the static output feedback problem, it is difficult to obtain fast convergence of such algorithms in general.

V. Stabilization Despite Unreliable Communication Links

In the previous section, we considered the case where messages exchanged between nodes are always delivered. Since unreliability of the communication links is one of the main drawbacks of wireless networks, in this section we focus on more "realistic" system models, where potential message drops are taken into account. In this case the system's evolution can be described as

$$\hat{\mathbf{x}}[k+1] = \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{G}_{\boldsymbol{\theta}(\mathbf{k})} \\ \mathbf{H}_{\boldsymbol{\theta}(\mathbf{k})}\mathbf{C} & \mathbf{W}_{\boldsymbol{\theta}(\mathbf{k})} \end{bmatrix} \hat{\mathbf{x}}[k] \triangleq \hat{\mathbf{A}}_{\theta(k)}\hat{\mathbf{x}}[k] \quad (8)$$

where $\hat{\mathbf{x}}[k] \in \mathbb{R}^{n+N}$ is the overall system's state and the subscript $\theta(k)$ describes time-variations in the matrices $(\mathbf{W}, \mathbf{H}, \mathbf{G})$ caused by (probabilistic) drops of communication packets. The focus of this section is to adapt the design procedure described in the previous section to generate a set of weights that guarantee stability of the system in a probabilistic sense, defined below.

Definition 1: ([26], [37]) The system is mean-square stable if for any initial state $(\hat{\mathbf{x}}[0], \theta(0))$, $\lim_{k\to\infty} \mathbb{E}\left[\|\hat{\mathbf{x}}[k]\|^2\right] = 0$, where the expectation is with respect to the probability distribution of the packet drop sequence $\theta(k)$. □

A vast amount of research has focused on the topic of designing controllers to stabilize plants over communication channels that are subject to Bernoulli packet drops, typically assuming the existence of a single unreliable channel between the plant sensor and the controller, and the controller and the plant actuators [9], [10]. However, there are relatively few results that explicitly consider packet drops in networked control systems with general topologies. The paper [11] considered the problem of the optimal location for a controller in a network and showed that placing the controller at the plant's actuator would maximize the amount of information available to the controller (since at any other location, it would not know whether a control signal that it sent to the actuator was dropped along the way). The papers [12], [38] considered the issue of allowing intermediate nodes to encode information that they are routing to the controller, so that the controller would receive enough information to stabilize the plant. All of these papers assume a single sensor and actuation point on the plant, and consider the existence of a designated controller within the network. It is worth noting that the papers [12] and [38] allow the intermediate nodes in the network to perform linear operations on the data that they send, but this is done purely to provide the dedicated controller with enough information about the state of the plant (as opposed to our approach, where the linear combinations are chosen so that the transmissions of the network as a whole are stabilizing).

The topic of modeling networks with unreliable channels was also considered in [37], where it was shown that such networks can be cast in a robust control framework, allowing an elegant approach to analysis and design. We will adapt this approach to the problem of designing a wireless control network with unreliable links. In the framework of [37], a communication link is modeled over time as a memoryless, discrete, independent and identically distributed (IID) random process ξ,[13] which maps each transmitted value $t_x[k]$ into a received value $r_x[k] = \xi[k]t_x[k]$.[14] For arbitrary nodes v_i and v_j, consider a communication link $(v_i, v_j) \in \mathcal{E}$ with weight w_{ji} [as shown in Fig. 3(a)].

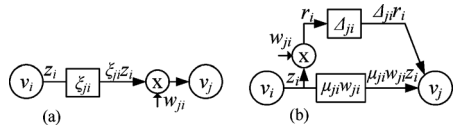


Fig. 3. Remote control over fading channel; (a) A link between nodes v_i and v_j; (b) Link transformation into a robust control form.

Let N_l denote the number of links in the graph $\bar{\mathcal{G}}$ defined in Section II, which contains the original network, together with the plant's sensors and actuators, along with the corresponding edges (i.e., $N_l = |\mathcal{E} \cup \mathcal{E}_I \cup \mathcal{E}_O|$). For convenience, we define a bijective mapping $\Omega : \mathcal{E} \cup \mathcal{E}_I \cup \mathcal{E}_O \to \{1, \ldots, N_l\}$ to enumerate all links in the network. In the rest of the paper, we will sometimes denote a link $(a, b) \in \mathcal{E} \cup \mathcal{E}_I \cup \mathcal{E}_O$ by its label $t = \Omega(a, b)$ for convenience, and its weight as w_t, h_t or g_t. In addition, all variables related to the link will be denoted with index t (e.g., $\xi_t[k]$, instead $\xi_{ji}[k]$). The contribution of the node v_i to the linear combination calculated by node v_j at time k can be represented as $w_t\xi_t[k]z_i[k]$, where ξ_t has mean $\mu_t = \mathbf{E}[\xi_t[k]]$ and a finite variance $\sigma_t^2 = \mathbf{E}\left[(\xi_t[k] - \mu_t)^2\right]$.

Following the approach in [37], we consider the link transformation shown in Fig. 3(b). By writing $\xi_t[k] = \mu_t + \Delta_t[k]$, where $\Delta_t[k]$ is a zero-mean random variable with variance σ_t^2, the original unreliable link is modeled as a combination of a deterministic link (without message drops) with gain μ_t and a random link described with gain $\Delta_t[k]$. Let $r_t[k]$ denote the signal transmitted over the t^{th} link, scaled by the weight on that link

$$r_t[k] = \begin{cases} h_t y_i[k] & \text{if } t = \Omega(s_i, v_j), \\ w_t z_i[k] & \text{if } t = \Omega(v_i, v_j), \\ g_t z_i[k] & \text{if } t = \Omega(v_i, a_j). \end{cases}$$

Stacking all of the $r_t[k]$'s in a vector $\mathbf{r}[k]$ of length N_l, we can write

$$\mathbf{r}[k] = \mathbf{J}^{or}\begin{bmatrix} \mathbf{y}[k] \\ \mathbf{z}[k] \end{bmatrix} = \mathbf{J}^{or}\begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_N \end{bmatrix}\hat{\mathbf{x}}[k] \triangleq \hat{\mathbf{J}}^{or}\hat{\mathbf{x}}[k] \quad (9)$$

where each row of the matrix $\mathbf{J}^{or} \in \mathbb{R}^{N_l \times (N+p)}$ contains a single nonzero element, equal to a gain w_t, h_t or g_t. More precisely, the matrix $\mathbf{J}^{or}$ is defined as

$$[\mathbf{J}^{or}]_{t,i} = \begin{cases} h_t, & \text{if } i \le p \text{ and } \exists v_j, \Omega(s_i, v_j) = t, \\ w_t, & \text{if } p < i \le N+p \text{ and } \exists v_j, \Omega(v_{i-p}, v_j) = t, \\ g_t, & \text{if } p < i \le N+p \text{ and } \exists a_j, \Omega(v_{i-p}, a_j) = t, \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

[13]Here IID implies that the random variables $\{\xi[k]\}_{k\ge 0}$ are IID.

[14]Note that a Bernoulli packet drop channel can be modeled by setting $\xi[k] = 0$ with probability p and 1 with probability $1 - p$.

Based on the link transformation shown in Fig. 3(b), and using (2), the update equation for each node v_j is

$$z_j[k+1] = w_{jj} z_j[k] + \sum_{t=\Omega(v_i,v_j)} \mu_t w_t z_i[k] + \sum_{t=\Omega(s_i,v_j)} \mu_t h_t y_i[k] + \sum_{t=\Omega(v_i,v_j)} \Delta_t[k] r_t[k] + \sum_{t=\Omega(s_i,v_j)} \Delta_t[k] r_t[k].$$

Also, from (3), the input value applied by each actuator at time step k is

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

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

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

where each nonzero entry of matrices $\mathbf{W}_\mu$, $\mathbf{H}_\mu$ and $\mathbf{G}_\mu$ (except the diagonal entries of $\mathbf{W}_\mu$) is of the form $\mu_t w_t$, $\mu_t h_t$ and $\mu_t g_t$, respectively. Each entry in the matrices $\mathbf{J}_v^{dst}$ and $\mathbf{J}_u^{dst}$ is either 0 or 1. Specifically, each row of those matrices simply selects which elements of the vector $\mathbf{\Delta}[k]\mathbf{r}[k]$ are added to the linear combinations calculated by the actuators and the wireless nodes. More precisely, matrices $\mathbf{J}_v^{dst} \in \mathbb{R}^{N \times N_l}$, $\mathbf{J}_u^{dst} \in \mathbb{R}^{m \times N_l}$ and $\mathbf{J}^{dst} \in \mathbb{R}^{(m+N) \times N_l}$ are given by

$$\left[\mathbf{J}_u^{dst}\right]_{i,t} = \begin{cases} 1, & \text{if } i \le m,\ \exists v_j \in \mathcal{V}, \Omega(v_j, a_i) = t \\ 0, & \text{else} \end{cases}$$
$$\left[\mathbf{J}^{dst}\right]_{i,t} = \begin{cases} 1, & \text{if } 1 \le i \le N, \text{ and } \exists v \in \mathcal{V} \cup \mathcal{S}, \Omega(v, v_i) = t, \\ 0, & \text{else.} \end{cases} \tag{11}$$

Defining $\mathbf{J}^{dst} = \begin{bmatrix} \mathbf{J}_u^{dst} \\ \mathbf{J}_v^{dst} \end{bmatrix}$ the overall system (with potential message drops) can be represented as

$$\hat{\mathbf{x}}[k+1] = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{G}_{\boldsymbol{\mu}} \\ \mathbf{H}_{\boldsymbol{\mu}}\mathbf{C} & \mathbf{W}_{\boldsymbol{\mu}} \end{bmatrix}}_{\hat{\mathbf{A}}_\mu} \hat{\mathbf{x}}[k] + \underbrace{\begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_N \end{bmatrix} \mathbf{J}^{dst}}_{\hat{\mathbf{J}}^{dst}} \mathbf{\Delta}[k] \mathbf{r}[k] \tag{12}$$

with $\mathbf{r}[k]$ given by (9).

As previously mentioned, an assumption is made that $\Delta_t[1], \Delta_t[2], \ldots \Delta_t[k], \ldots$ are independent zero-mean random variables with variance σ_t^2. In addition, we assume that all random variables, $\Delta_1, \ldots, \Delta_{N_l}$ are independent (i.e., link failures are independent across time and space). With this assumption, using the approach in [37], we obtain the following result.[15]

Theorem 3: The system from (12) is MSS if and only if there exists a positive-definite matrix $\mathbf{X}$ and scalars $\alpha_1, \ldots, \alpha_{N_l}$ satisfying the LMIs

$$\mathbf{X} \succ \hat{\mathbf{A}}_\mu \mathbf{X} \hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst} \text{diag}\{\boldsymbol{\alpha}\} (\hat{\mathbf{J}}^{dst})^T$$
$$\alpha_i \ge \sigma_i^2 (\hat{\mathbf{J}}^{or})_i \mathbf{X} (\hat{\mathbf{J}}^{or})_i^T,\ \forall i \in \{1, \ldots, N_l\} \tag{13}$$

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

[15]The proof of this theorem is a special case of the proof of Theorem 5 provided in the Appendix.

Using the Schur complement and the cone complementarity condition as in Section IV, **Algorithm** 2 (shown below) can be constructed to solve the inequalities presented in the above theorem (note that the matrix $\hat{\mathbf{J}}^{dst}$ and σ_i's are constants). As in the previous section, we obtain the following theorem.

Algorithm 2 Stabilizing the closed-loop system with unreliable communication links

1. Find feasible points $\mathbf{X}_0$, $\mathbf{Y}_0$, $\mathbf{W}_0$, $\mathbf{H}_0$, $\mathbf{G}_0$ that satisfy the constraints (13), where
$$\begin{bmatrix} \mathbf{X}_0 & \mathbf{I} \\ \mathbf{I} & \mathbf{Y}_0 \end{bmatrix} \succeq 0,\ (\mathbf{W}_0, \mathbf{H}_0, \mathbf{G}_0) \in \Psi,\ \mathbf{X}_0, \mathbf{Y}_0 \in \mathbb{S}_{++}^{n+N}.$$
If there is no feasible point, it is not possible to obtain MSS with this network topology and distribution on the communication links.
2. At iteration k, $(k \ge 0)$ from $\mathbf{X}_k$, $\mathbf{Y}_k$ obtain the matrices $\mathbf{X}_{k+1}$, $\mathbf{Y}_{k+1}$, $\mathbf{W}_{\boldsymbol{\mu},k+1}$, $\mathbf{H}_{\boldsymbol{\mu},k+1}$, $\mathbf{G}_{\boldsymbol{\mu},k+1}$ and a vector $\boldsymbol{\alpha}_{k+1} \in \mathbb{R}^{N_l}$ by solving the following LMI problem:
$$\min\ tr(\mathbf{Y_k}\mathbf{X}_{k+1} + \mathbf{X_k}\mathbf{Y}_{k+1})$$
$$\begin{bmatrix} \mathbf{X}_{k+1} - (\hat{\mathbf{J}}^{dst})^T diag\{\boldsymbol{\alpha}_{k+1}\} (\hat{\mathbf{J}}^{dst})^T & \hat{\mathbf{A}}_{\mu,k+1} \\ \hat{\mathbf{A}}_{\mu,k+1}^T & \mathbf{Y}_{k+1} \end{bmatrix} \succ 0,$$
$$\begin{bmatrix} \mathbf{X}_{k+1} & \mathbf{I} \\ \mathbf{I} & \mathbf{Y}_{k+1} \end{bmatrix} \succeq 0,$$
$$\hat{\mathbf{A}}_{\mu,k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{G}_{\boldsymbol{\mu},k+1} \\ \mathbf{H}_{\boldsymbol{\mu},k+1}\mathbf{C} & \mathbf{W}_{\boldsymbol{\mu},k+1} \end{bmatrix},$$
$$\begin{bmatrix} \alpha_{i,k+1} & \sigma_i (\hat{\mathbf{J}}_{k+1}^{or})_i \\ \sigma_i (\hat{\mathbf{J}}_{k+1}^{or})_i^T & \mathbf{Y} \end{bmatrix}_{k+1} \succeq 0,\ 1 \le i \le N_l,$$
$$(\mathbf{W}_{\boldsymbol{\mu},k+1}, \mathbf{H}_{\mu,k+1}, \mathbf{G}_{\mu,k+1}) \in \Psi,\ \mathbf{X}_{k+1}, \mathbf{Y}_{k+1} \in \mathbb{S}_{++}^{n+N}.$$
3. Stop the algorithm if the following conditions are true:
$$\mathbf{X}_{k+1} \succ \hat{\mathbf{A}}_{\mu,k+1} \mathbf{X}_{k+1} \hat{\mathbf{A}}_{\mu,k+1}^T + \hat{\mathbf{J}}^{dst} diag\{\boldsymbol{\alpha}_{k+1}\} (\hat{\mathbf{J}}^{dst})^T$$
$$\alpha_{i,k+1} \ge \sigma_i^2 (\mathbf{J}_{k+1}^{or})_i \mathbf{X}_{k+1} (\mathbf{J}_{k+1}^{or})_i^T,\ 1 \le i \le N_l.$$
Otherwise, set $k = k+1$ and go to step **2**.

Theorem 4: **Algorithm** 2 will determine the tuple $(\mathbf{W}, \mathbf{G}, \mathbf{H}) \in \Psi$ that guarantees MSS of the system under the given distribution of the link failures in the network if the sequence $t_k = tr(\mathbf{Y}_k \mathbf{X}_{k+1} + \mathbf{X}_k \mathbf{Y}_{k+1})$ converges to $2(n+N)$. □

Remark 3: It is worth noting that any matrices $\hat{\mathbf{A}}_\mu$, $\mathbf{X}$, $\mathbf{J}^{dst}$, $\mathbf{J}^{or}$ and vector $\boldsymbol{\alpha}$ that satisfy the constraints from (13) for the link quality vector $\bar{\boldsymbol{\sigma}} = [\bar{\sigma}_1, \ldots, \bar{\sigma}_{N_l}]^T$, also satisfy the constraints for any vector $\boldsymbol{\sigma}$ such that $\boldsymbol{\sigma} \preceq \bar{\boldsymbol{\sigma}}$ (where "$\preceq$" implies elementwise inequality). Thus, finding a vector $\underline{\boldsymbol{\sigma}} = \sigma \mathbf{1},\ \sigma \in \mathbb{R}$ for which there exists matrices $\mathbf{W}$, $\mathbf{G}$ and $\mathbf{H}$ that guarantee MSS allows the use of the same matrices $\mathbf{W}$, $\mathbf{G}$, $\mathbf{H}$ even when the link qualities are better than that specified by the vector $\underline{\boldsymbol{\sigma}}$. The largest value of σ for which the system is MSS can be found by allowing $\boldsymbol{\sigma} \in \mathbb{R}$ to be a variable. This causes the last matrix inequality in step 2 of Algorithm 2 to be a bilinear constraint, but

this can be handled by using bisection on the parameter $\sigma \in \mathbb{R}$ (e.g., as done in [26]). □

Remark 4: Note that the number of constraints in **Algorithm** 2 grows *linearly* with the number of links in the network, rather than exponentially (i.e., we do not have to consider all possible combinations of failed links at any given time-step). This is due to the fact that stability under link failures is viewed as a problem of *robustness* in a linear system in this framework, which is one of its main benefits (note that we can model the WCN as a linear system precisely due to the specific linear iterative strategy that we are having each node follow). □

VI. Incorporating More Powerful Nodes

Our development so far has treated the case where each node in the WCN maintains a single scalar state, which allows very simple nodes to be used for controlling the plant. However, our approach can also be used to design heterogeneous networks where nodes may have different memory and computing capabilities (including the case where some nodes are, in fact, dedicated controllers). Mathematically, this can be modeled by describing node v_i's state with a *vector* $\mathbf{z}_i \in \mathbb{R}^{n_i}$, with update procedure (similar to (2))

$$\mathbf{z}_i[k+1] = \mathbf{w}_{ii}\mathbf{z}_i[k] + \sum_{v_j \in \mathcal{N}_{v_i}} \mathbf{w}_{ij}\mathbf{z}_j[k] + \sum_{s_j \in \mathcal{N}_{v_i}} \mathbf{h}_{ij} y_j[k]$$

where $\mathbf{w}_{ij} \in \mathbb{R}^{n_i \times n_j}$ and $\mathbf{h}_{ij} \in \mathbb{R}^{n_i}$. In addition, a plant's input u_i at time-step k has the form

$$u_i[k] = \sum_{j \in \mathcal{N}_{a_i}} \mathbf{g}_{ij}\mathbf{z}_j[k]$$

with $\mathbf{g}_{ij} \in \mathbb{R}^{1 \times n_j}$. If each value from a node's state is transmitted in separate packets, a node v_i could be modeled as n_i different nodes $(v_i^j, j \in \{1, \ldots, n_i\})$, where each node would maintain a scalar value as its state. This would allow the use of **Algorithm** 2 to determine matrices $\mathbf{W}$, $\mathbf{H}$, $\mathbf{G}$ that have the desired sparsity pattern and guarantee MSS of the system. However, this scheme would also require more than one transmission per node per frame, which would increase the minimal required sampling period of the plant and complicate the task of scheduling transmissions. Instead, if each node transmits its whole state vector in a single message, a node cannot be modeled as a set of separate independent nodes, as in this case the assumption that all channels are independent is not valid (if the packet is not received, all values from the packet are lost). In this case, each link $t = \Omega(v_i, v_j)$ or $t = \Omega(v_i, a_j)$ will carry $c_t = n_i$ scalar values at each time-step. As in Section V, let $\mathbf{r}_t[k]$ denote the signal transmitted over the t-th link, scaled by the weight (matrix) on that link (i.e., $\mathbf{r}_t[k]$ is $\mathbf{h}_t y_i[k]$, $\mathbf{w}_t\mathbf{z}_i[k]$ or $\mathbf{g}_t\mathbf{z}_i[k]$, depending on whether the link is from a sensor to a node, between two nodes, or from a node to an actuator, respectively). Let c'_t denote the size of $\mathbf{r}_t[k]$, and let $C_s = \sum_{t=1}^{N_l} c'_t$, where N_l denotes the number of links in the network. As in Section V, the overall system is given by (9) and (12), where $\hat{\mathbf{x}}[k] \in \mathbb{R}^{N_s}$, $\mathbf{G}_\mu \in \mathbb{R}^{m \times N_s}$, $\mathbf{H}_\mu \in \mathbb{R}^{N_s \times p}$, $\mathbf{W}_\mu \in \mathbb{R}^{N_s \times N_s}$ and $\mathbf{J}^{dst} \in \mathbb{R}^{(m+N_s) \times C_s}$, $\mathbf{J}^{or} \in \mathbb{R}^{C_s \times (p+N_s)}$, with $N_s = \sum_{i=1}^{N} n_i$ representing the total number of states over all nodes. The matrices $\mathbf{J}^{or}$ and $\mathbf{J}^{dst}$ are defined as in (10) and (11), with a small difference that instead of scalars, matrices of appropriate dimensions are used.[16] In addition, $\mathbf{r}[k] \in \mathbb{R}^{C_s}$ and $\boldsymbol{\Delta}[k] = \mathbf{blkdiag}(\{\boldsymbol{\Delta}_t[k]\}_{t=1}^{N_l})$, where **blkdiag** is a block-diagonal operator and $\boldsymbol{\Delta}_t[k] = \Delta_t[k]\mathbf{I}_{c'_t \times c'_t}$. This brings us to the following theorem (the proof is provided in the Appendix).

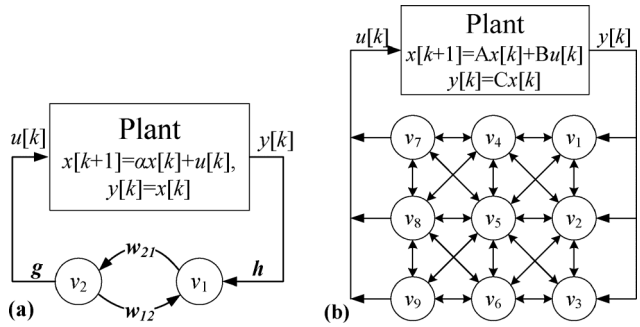


Fig. 4. Two examples of WCNs; (a) A plant with a scalar state controlled by a WCN where each node maintains a scalar state; (b) A single-input-single output plant with $\mathbb{R}^3$ state controlled by a WCN where each node maintains a scalar state.

Theorem 5: The system described with (12) and (9) (with vector states at each node) is MSS if and only if there exist positive-definite matrices $\mathbf{X}$ and $\boldsymbol{\alpha}_i \in \mathbb{R}^{c_i' \times c_i'}$, $i \in \{1, \ldots, N_l\}$ that satisfy the following LMIs:

$$\begin{aligned} &\mathbf{X} \succ \hat{\mathbf{A}}_\mu \mathbf{X} \hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst} \mathbf{blkdiag}\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{N_l}\} (\hat{\mathbf{J}}^{dst})^T \\ &\boldsymbol{\alpha}_i \succeq \sigma_i^2 (\hat{\mathbf{J}}^{or})_i \cdot \mathbf{X} \cdot (\hat{\mathbf{J}}^{or})_i^T, \ \forall i \in \{1, \ldots, N_l\} \end{aligned} \tag{14}$$

where $(\hat{\mathbf{J}}^{or})_i$ denotes the i^{th} block-row of the matrix $\hat{\mathbf{J}}^{or}$ (containing c'_i rows from $\sum_{t=1}^{i-1} c'_t + 1$ to $\sum_{t=1}^{i} c'_t$). □

Using the theorem above, an algorithm similar to **Algorithm** 2 can be used to determine the (vector-valued) update for each node to apply in order to guarantee MSS.

VII. Examples

To illustrate the application of our design procedure from the previous sections, consider the single state plant shown in Fig. 4(a) and suppose that each link in the network is modeled as an independent Bernoulli process with probability of losing a packet equal to p (the variance of each process is $\sigma^2 = p(1-p)$). To solve the optimization problem from **Algorithm** 2 we used CVX, a package for specifying and solving convex programs [39]. For $\alpha = 2$ and $p = 0.5\%$, if each node maintains a scalar state, **Algorithm** 2 converges after 51 iterations[17] to a stable configuration

$$\mathbf{W} = \begin{bmatrix} 0.228 & 0.965 \\ -2.872 & -1.660 \end{bmatrix}, \ \mathbf{H} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{G} = \begin{bmatrix} 0 & 1.837 \end{bmatrix}. \tag{15}$$

Using the bisection method described in the Remark 3, we extracted the maximal probabilities of message drops, $p_{\max}$, for which there exists a tuple $(\mathbf{W}, \mathbf{H}, \mathbf{G}) \in \Psi$ that guarantees MSS. We considered two cases, one where all nodes in the network maintain a scalar state and the other where they maintain a

[16]Specifically, matrices $\mathbf{w}_t$, $\mathbf{g}_t$ and $\mathbf{h}_t$ should be substituted for scalars w_t, g_t and h_t, respectively. Also, instead of the scalar '1' in (11), the identity matrix $\mathbf{I}_{n_j}$ should be used, where n_i is the state vector size for the link's receiving node.

[17]The number of iterations needed before the algorithm converges to a stable configuration depends on initial points $\mathbf{X}_0$, $\mathbf{Y}_0$.

TABLE I
MAXIMAL MESSAGE DROP PROBABILITY FOR MSS IN FIG. 4(A)

	$n_i = 1$ (scalar state)	$n_i = 2$ ($\mathbb{R}^2$ state)
$N = 2$	$p_{max} = 0.69\%$	$p_{max} = 0.72\%$
$N = 3$	$p_{max} = 0.74\%$	$p_{max} = 0.77\%$
$N = 4$	$p_{max} = 0.77\%$	$p_{max} = 0.79\%$

vector state in $\mathbb{R}^2$. In addition, networks with $N = 3$ and $N = 4$ nodes were considered, where the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is complete ($\mathcal{V} = \{v_1, \ldots, v_N\}$). The obtained results are presented in Table I. As can be seen, adding additional nodes does not significantly increase $p_{\max}$ for this example; a possible hypothesis for this is that the single link between node v_N and the actuator is the bottleneck for stability. Similarly, adding more powerful nodes with larger WCN states does not increase the robustness of the system to packet drops in this particular example.[18]

To show compositionality, consider the system presented in Fig. 4(b), with a single-input-single-output plant of the form (1), where

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0.5 & 1 \end{bmatrix}$$

is controlled by a WCN consisted of nine nodes with a mesh topology. The nodes v_1, v_2 and v_3 are in the neighborhood of the plant's sensor, while nodes v_7, v_8 and v_9 can communicate with the plant's actuator. As in the previous example, all links in the network are modeled as independent Bernoulli processes with probability of losing a packet equal to p. We consider the case where $p = 0.1\%$ for all links except the links between the sensor and nodes v_1, v_2 and v_3 and the links between nodes v_7, v_8 and v_9 and the actuator. In this case, **Algorithm** 2 converged to the stable configuration shown in (16), (17), as shown at the bottom of the page.

Now consider a scenario in which the plant from Fig. 4(a) is added to the system in a way that its sensor can communicate only to node v_4, while its actuator can receive packets only from node v_7. If these two links are modeled as Bernoulli links with $p \leq 0.5\%$ nodes v_4 and v_7 can be used to control the plant with configuration derived in the previous example (from Remark 3, the derived configuration guarantees MSS for networks where packet loss probabilities for all links are less or equal 0.5%). In this case, both plants can be controlled with the WCN from Fig. 4(b), where all nodes in the WCN maintain two scalar states and calculate the first update using the coefficients from (16), (17) while the second update is calculated using a matrix $\mathbf{W}$ where only w_{44}, w_{74}, w_{47}, w_{77} are nonzero and are derived from (15). As described in Section III, we were not required to model both plants as a single plant in order to extract a stable configuration (under the assumption that each node can maintain a vector state in $\mathbb{R}^2$), but were rather able to *compose* previously computed stable configurations. It is worth noting that, although the previous two examples were calculated for networks with different topologies, in cases when a network is a subgraph of another network, the former stable configuration can be simply 'extended' by adding zeros to unused links in the latter network. Furthermore, note that new communication and computation schedules are not required for the WCN; the new plant is controlled simply by increasing the information in the (single) transmission by each node.

To test our algorithm on an even more complex example, we generated a random plant with $n = 50$ states, $p = 10$ inputs and $m = 10$ inputs, with approximately three eigenvalues in the interval $(1, 1.1]$. The plant is connected to the WCN with topology shown in Fig. 5, where each sensor s_j can measure the j^{th} output (y_j), while each input u_i is controlled by the actuator a_i. **Algorithm** 1 converged in less than 27 minutes to a stable configuration. However, as the considered network has 132 unidirectional links (since a bidirectional link is considered as two unidirectional links), the optimization problem considered in **Algorithm** 2 has 132 additional constraints compared to the optimization problem in **Algorithm** 1. This increase in the number of constraints proved to be too much for CVX to handle, causing it to exceed the memory available on our computers; this was not unexpected, however, as CVX is not designed to deal with large scale problems [39]. Part of our future work in this area will be to write a dedicated solver to fully test our scheme on large-scale systems.

VIII. DISCUSSION

A. Relationship Between the WCN and Multi-Hop Delays

At first glance, the WCN might seem to introduce some delay into the feedback loop (since the sensor nodes and actuator

[18]However, more powerful nodes can be shown to improve resilience to packet drops in other scenarios [40].

$$\mathbf{W} = \begin{bmatrix} 0.799 & 0.030 & 0 & 0.047 & 0.018 & 0 & 0 & 0 & 0 \\ -3.677 & -2.097 & -2.768 & 0.078 & 0.107 & 0.188 & 0 & 0 & 0 \\ 0 & 0.021 & 0.787 & 0 & 0.029 & -0.002 & 0 & 0 & 0 \\ -10.770 & 4.247 & 0 & -0.560 & 0.067 & 0 & 0.035 & -1.148 & 0 \\ -0.992 & 0.713 & -1.008 & 0.163 & 0.757 & -0.025 & -0.030 & 0.291 & -0.141 \\ 0 & -8.576 & 12.909 & 0 & -0.314 & 1.370 & 0 & 3.769 & 0.038 \\ 0 & 0 & 0 & -3.587 & 1.366 & 0 & 0.691 & 4.861 & 0 \\ 0 & 0 & 0 & 1.661 & -0.060 & -0.459 & -0.015 & -0.144 & -0.090 \\ 0 & 0 & 0 & 0 & 0.286 & -0.204 & 0 & 0.805 & 0.744 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 0.017 \\ 0.927 \\ 0.020 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (16)$$

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.056 & -1.620 & 0.233 \end{bmatrix} \quad (17)$$

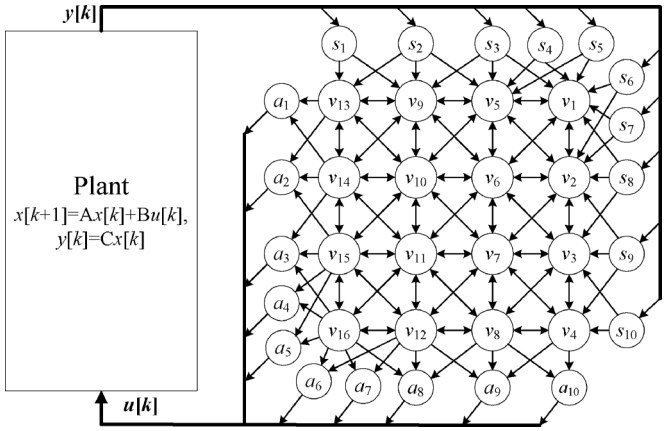


Fig. 5. Example of plant with 30 states controlled by a WCN consisted of mesh network with 16 nodes.

nodes might be separated by multiple intermediate nodes, each taking one time-step to propagate information), which might limit the class of plants that can be stabilized with this method. However, the relationship between the WCN and the traditional notions of delay introduced by the feedback loop is not as obvious as it might appear at first glance. Specifically, note that we allow each node in the network to maintain a value that is a function of its previous value and the values of all its neighbors, rather than simply routing values to a controller. This simple modification causes the network to act as a *linear dynamical system* with sparsity constraints in the system matrices; in other words, this control scheme should be viewed as a dynamic compensator, rather than a static feedback gain at the end of a chain of delay elements. The following example shows that this fact allows our scheme to stabilize plants that cannot be stabilized with delayed static feedback.

Consider once again the single-state plant shown on the left side of Fig. 4(a) (with $\alpha > 1$), which is to be controlled by a network with two nodes v_1 and v_2. Node v_1 receives the plant output $y[k] = x[k]$ at each time-step k, and the input to the plant is taken to be a scaled version of the transmission of the node v_2 (i.e., $u[k] = gz_2[k]$, for some scalar g). If the nodes apply the linear strategy that we study in this paper, the closed loop system evolves according to

$$\begin{bmatrix} x[k+1] \\ z_1[k+1] \\ z_2[k+1] \end{bmatrix} = \begin{bmatrix} \alpha & 0 & g \\ h & w_{11} & w_{12} \\ 0 & w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x[k] \\ z_1[k] \\ z_2[k] \end{bmatrix} \tag{18}$$

for some scalars $w_{11}, w_{12}, w_{21}, w_{22}, g$ and h. Recall from the example in Section VII that these scalars can be chosen so that the closed loop system is stable. In fact, if one chooses the values $g = h = 1, w_{11} = 0, w_{12} = 1/\alpha, w_{21} = -\alpha^3$ and $w_{22} = -\alpha$, the closed-loop system will have all poles at zero for any $\alpha \neq 0$.

Now, consider a control scheme where node v_1 simply forwards the state measurement to v_2 at each time-step, and v_2 sends this value to the actuator where the input $u[k] = gz_2[k]$ is applied. This can be modeled by setting $w_{11} = w_{12} = w_{22} = 0$, $w_{21} = 1$, and $h = 1$ in (18). The characteristic polynomial of this system is $z^2(z - \alpha) - g$, and one can show (e.g., using the root locus) that it is possible to find a g such that this polynomial has all roots inside the unit circle if and only if $|\alpha| < 3/2$. In other words, the delay introduced by this routing scheme limits the class of plants that can be stabilized. As a further example, consider the case where we allow w_{22} to be nonzero (thereby allowing v_2 to be a "controller" with dynamics, while v_1 is still a router). In this case, the characteristic polynomial is $z^3 - (\alpha + w_{22})z^2 + \alpha w_{22} z - g$. Letting p_1, p_2, p_3 denote the roots of this polynomial, we see that $\alpha + w_{22} = p_1 + p_2 + p_3$ and $\alpha w_{22} = p_1p_2 + p_1p_3 + p_2p_3$. Now, if all roots are inside the unit circle, it follows that:

$$-3 < p_1 + p_2 + p_3 < 3, \quad -3 < p_1p_2 + p_1p_3 + p_2p_3 < 3$$

from which we see that $-3 - \alpha < w_{22} < 3 - \alpha$ and $-3/\alpha < w_{22} < 3/\alpha$ for stability. For certain values of α, it will not be possible to find a parameter w_{22} that satisfies both of these inequalities (e.g., for any $\alpha \geq (3 + \sqrt{21})/2$). Thus, stability is not achievable even in the case where v_2 has (scalar) dynamics but v_1 is a router. One obtains stability for arbitrary values of α and with scalar computations at each node only by allowing both v_1 and v_2 to update their values with a linear strategy (as demonstrated above).

B. Adapting to Node Failures

The stability of the system can be affected by crash failures (nodes that stop working and drop out of the network). One obvious approach to deal with up to f crash failures is to precalculate a set of $\sum_{j=0}^{f} \binom{N}{j}$ different tuples $(\mathbf{W}, \mathbf{H}, \mathbf{G})$ (corresponding to all possible choices of f or fewer failed nodes), and have each node maintain a table of these different configurations. The neighbors of failed nodes can broadcast the news of the failures throughout the network, which will prompt all nodes to switch to the appropriate choice of $(\mathbf{W}, \mathbf{H}, \mathbf{G})$. This approach may not be satisfactory in practice, as it requires precomputation and storage of a large number of matrices.

Fortunately, the WCN allows a more elegant (and distributed) method to handle node failures. Specifically, when a node fails, all neighbors of that node increase their transmission power to be able to communicate with all other neighbors of the failed nodes.[19] Furthermore, the computations of the failed node are passed on to one of its neighbors (this can be performed in a distributed manner during run-time via a simple *leader-election protocol* [41]). This neighbor then becomes a *virtual node*, emulating the behavior of the failed node by maintaining its state, and transmitting and receiving in the time-slots assigned to the failed node (in addition to maintaining and transmitting its own state, as usual). With this scheme, all other nodes in the network (with the exception of the neighbors of the failed node) continue to operate as normal. Note that this method causes some nodes to expend more power after failures (due to the fact that neighbors of the failed node have to transmit over longer distances and one neighbor performs extra computations). However, it presents a simple distributed approach to ensure graceful degradation of the network under failures.

[19]This can be done without causing collisions in the transmissions if redundancy is incorporated into the interference graph during the design of the transmission schedules, e.g., so that 2-hop neighbors of each node are also included in the interference graph, and so forth.

IX. Conclusion

We have introduced the concept of a Wireless Control Network, where the network itself acts as a controller for the plant. Each node in a WCN executes a simple procedure by updating its state to be a linear combination of the states of its neighbors. We presented a procedure that can be used to design the linear combinations in order to stabilize the closed loop system. In addition, we showed that the aforementioned procedure can be made robust to link failures in the network.

While the proposed scheme has several benefits in comparison to traditional control schemes (as described in Section III), there are also some drawbacks which will be addressed through future research. We discuss some of these below.

- Our approach can readily handle plants with multiple actuation and sensing points, and can explicitly account for computational constraints in the nodes in the network. However, in plants with single sensing and actuation points, it is worth noting that our scheme will generally under-perform traditional networked control approaches when it comes to maximizing the probability of packet drops under which MSS can be maintained. This is because a sufficiently powerful controller effectively emulates a fully connected network (since there are no sparsity constraints imposed *a priori* on the controller matrices), without any packet drops between the nodes. Furthermore, by allowing intermediate nodes in the network to encode information based on the actual sequence of packet drops that occur (e.g., as done in [12], [38]), the nodes are able to send information more 'intelligently' to the controller (as opposed to our very simple scheme where all nodes update their values in the same way at each time-step, incorporating their neighbors' values only if they are received). While our design procedure is capable of handling powerful nodes in the network (as described in Section VI), extensions that allow nodes to perform more complicated operations (e.g., such as Kalman filtering) will be an avenue for future work.
- We have assumed independent link failures in the network (both in time and in space). Other works on networked control (such as [12]) have studied methods of dealing with arbitrary models of link failures, and it will be of interest to extend our design algorithms to such cases.
- Our scheme to handle node failures (described in Section VIII-B) can only be applied up to a certain point, as the transmission ranges of nodes cannot be increased indefinitely. In addition, multiple failures in any given neighborhood might impose a large amount of overhead on the part of the remaining nodes. A more robust and adaptive scheme to handle different fault models in nodes is desirable.
- This paper assumes that the topology of the WCN is specified *a priori*, and presents a numerical algorithm to design the link weights for each node. The dual approach of finding appropriate topologies that will be capable of stabilizing a given system is an avenue for future work. Along similar lines, it would be of interest to find methods to map existing controller designs onto the substrate provided by the WCN.

Appendix
Proof of Theorem 5

Proof: (A slight generalization of the approach from [37] is used in the proof.) Consider a linear system of the form

$$\begin{aligned}\hat{\mathbf{x}}[k+1] &= \hat{\mathbf{A}}_\mu \hat{\mathbf{x}}[k] + \hat{\mathbf{J}}^{dst}\boldsymbol{\Delta}[k]\mathbf{r}[k],\\ \mathbf{r}[k] &= \hat{\mathbf{J}}^{or}\hat{\mathbf{x}}[k].\end{aligned} \tag{19}$$

Definition 1 for MSS is equivalent to saying that the state covariance matrix $\mathbf{M}[k] \triangleq \mathbf{E}\{\hat{\mathbf{x}}[k]\hat{\mathbf{x}}[k]^T\}$ is bounded for all k, and goes to zero as $k \to \infty$. From (19), we obtain

$$\begin{aligned}&\mathbf{M}[k+1]\\ &= \hat{\mathbf{A}}_\mu \mathbf{M}[k]\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst}\mathbb{E}\{\boldsymbol{\Delta}[k]\mathbf{r}[k]\mathbf{r}[k]^T\boldsymbol{\Delta}[k]\}\hat{\mathbf{J}}^{dst^T}\\ &\quad + \hat{\mathbf{J}}^{dst}\mathbb{E}\{\boldsymbol{\Delta}[k]\mathbf{r}[k]\hat{\mathbf{x}}[k]^T\}\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{A}}_\mu\mathbb{E}\{\hat{\mathbf{x}}[k]\mathbf{r}[k]^T\boldsymbol{\Delta}[k]^T\}\hat{\mathbf{J}}^{dst^T}\\ &= \hat{\mathbf{A}}_\mu \mathbf{M}[k]\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst}\mathbb{E}\{\boldsymbol{\Delta}[k]\mathbf{r}[k]\mathbf{r}[k]^T\boldsymbol{\Delta}[k]\}\hat{\mathbf{J}}^{dst^T}.\end{aligned}$$

The third and fourth terms in the first equation are zero since all Δ's in have zero means and are independent from $\mathbf{r}[\mathbf{k}]$ and $\hat{\mathbf{x}}[k]$.

Now, consider the term $\mathbf{T} = \mathbb{E}\{\boldsymbol{\Delta}[k]\mathbf{r}[k]\mathbf{r}[k]^T\boldsymbol{\Delta}[k]\}$, and note that $\mathbf{r}_t[k] = (\hat{\mathbf{J}}^{or})_t\hat{\mathbf{x}}[k]$, where $(\hat{\mathbf{J}}^{or})_t$ denotes the t^{th} block-row of the matrix $\hat{\mathbf{J}}^{or}$ (corresponding to the link $t = \Omega(v_i, v_j)$). Since $\boldsymbol{\Delta}[k] = \mathbf{blkdiag}(\{\boldsymbol{\Delta}_t[k]\}_{t=1}^{N_l})$ and $\boldsymbol{\Delta}_t[k] = \Delta_t[k]\mathbf{I}_{c_t' \times c_t'}$, the $(t_1, t_2)^{th}$ block submatrix of $\mathbf{T}$ is given by $\mathbf{T}_{t_1t_2}[k] \triangleq \mathbb{E}\{\boldsymbol{\Delta}_{t_1}[k]\mathbf{r}_{t_1}[k]\mathbf{r}_{t_2}[k]^T\boldsymbol{\Delta}_{t_2}[k]\}$. When $t_1 \neq t_2$

$$\mathbf{T}_{t_1t_2}[k] = \mathbb{E}\{\Delta_{t_1}[k]\mathbf{I}_{c_{t_1}}\mathbf{r}_{t_1}[k]\mathbf{r}_{t_2}[k]^T\mathbf{I}_{c_{t_2}}\Delta_{t_2}[k]\} = \mathbf{0}_{c_{t_1}' \times c_{t_2}'}$$

as Δ_{t_1} and Δ_{t_2} are independent, zero mean, random variables. When $t_1 = t_2 = t$, using the same approach as in the previous case gives us

$$\begin{aligned}\mathbf{T}_{tt}[k] &= \sigma_t^2\mathbb{E}\{\mathbf{r}_{t_1}[k]\mathbf{r}_{t_2}[k]^T\} = \sigma_t^2\mathbb{E}\{(\hat{\mathbf{J}}^{or})_t\hat{\mathbf{x}}[k]\hat{\mathbf{x}}[k]^T(\hat{\mathbf{J}}^{or})_t^T\}\\ &= \sigma_t^2(\hat{\mathbf{J}}^{or})_t\mathbf{M}[k](\hat{\mathbf{J}}^{or})_t^T \triangleq \boldsymbol{\alpha}_t\end{aligned}$$

where $\boldsymbol{\alpha}_t \in \mathbf{R}^{c_t' \times c_t'}$. Therefore, we have

$$\begin{aligned}\mathbf{M}[k+1] &= \hat{\mathbf{A}}_\mu \mathbf{M}[k]\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst}\mathbf{blkdiag}\{\boldsymbol{\alpha}_t, \ldots, \boldsymbol{\alpha}_{N_l}\}\hat{\mathbf{J}}^{dst^T}\\ \boldsymbol{\alpha}_t &= \sigma_t^2(\hat{\mathbf{J}}^{or})_t\mathbf{M}[k](\hat{\mathbf{J}}^{or})_t^T.\end{aligned}$$

This is essentially of the same form as the equations in Theorem 6.4 from [37] (except for the fact that the $\boldsymbol{\alpha}_t$ variables are matrices in our case). Therefore, $\mathbf{M}$ can be expressed using a linear recursion and thus mean square stability is equivalent to the existence of positive-definite matrices $\mathbf{X}$ and $\boldsymbol{\alpha}_i \in \mathbb{R}^{c_i' \times c_i'}$, $i \in \{1, \ldots, N_l\}$ that satisfy conditions (14) of Theorem 5 [37], [42]. ■

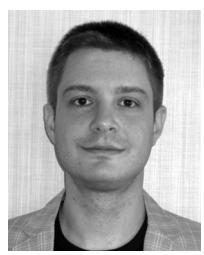
Acknowledgment

The authors thank P. McLaughlin and A. Chernoguzov, Honeywell Process Solutions, and J. Le Ny and A. Jadbabaie, University of Pennsylvania, for fruitful discussions during the preparation of this paper, and the reviewers for constructive comments.

References

[1] M. Pajic, S. Sundaram, J. Le Ny, G. J. Pappas, and R. Mangharam, "The wireless control network: Synthesis and robustness," in *Proc. 49th IEEE Conf. Decision Control*, 2010, pp. 7576–7581.

[2] Why WirelessHART? White Paper HART Communication Foundation, 2007.
[3] S. Amidi and A. Chernoguzov, "Wireless process control network architecture overview," in *White Paper*. Morristown, NJ: Honeywell International, Inc., 2009.
[4] M. Pajic and R. Mangharam, "Embedded virtual machine for robust wireless control and actuation," in *Proc. 16th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS'10)*, 2010, pp. 79–88.
[5] U. Jecht, W. Stripf, and P. Wenzel, "Profibus: Open solutions for the world of automation," in *The Industrial Information Technology Handbook*, R. Zurawski, Ed. Orlando, FL: CRC Press, 2005.
[6] G. Cena and A. Valenzano, "Operating principles and features of CAN networks," in *The Industrial Information Technology Handbook*, R. Zurawski, Ed. Orlando, FL: CRC Press, 2005.
[7] C. N. Hadjicostis and R. Touri, "Feedback control utilizing packet dropping network links," in *Proc. 41st IEEE Conf. Decision Control*, 2002, pp. 1205–1210.
[8] O. C. Imer, S. Yuksel, and T. Basar, "Optimal control of LTI systems over unreliable communication links," *Automatica*, vol. 42, no. 9, pp. 1429–1439, Sep. 2006.
[9] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proc. IEEE*, vol. 95, no. 1, pp. 138–162, Jan. 2007.
[10] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry, "Foundations of control and estimation over lossy networks," *Proc. IEEE*, vol. 95, no. 1, pp. 163–187, Jan. 2007.
[11] C. L. Robinson and P. R. Kumar, "Optimizing controller location in networked control systems with packet drops," *IEEE J. Selected Areas Commun.*, vol. 26, no. 4, pp. 661–671, May 2008.
[12] V. Gupta, A. F. Dana, J. Hespanha, R. M. Murray, and B. Hassibi, "Data transmission over networks for estimation and control," *IEEE Trans. Autom. Control*, vol. 54, no. 8, pp. 1807–1819, Aug. 2009.
[13] N. Gupta and P. R. Kumar, "A performance analysis of the IEEE 802.11 wireless lan medium access control," *Commun. Inform. Syst.*, vol. 3, no. 4, pp. 279–304, 2003.
[14] *Wireless Systems for Industrial Automation: Process Control and Related Applications*, ISA100.11a, 2009.
[15] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proc. IEEE*, vol. 91, no. 1, pp. 112–126, Jan. 2003.
[16] R. Alur, A. D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss, "Modeling and analysis of multi-hop control networks," in *Proc. 15th IEEE Symp. Real-Time Embedded Technol. Appl.*, 2009, pp. 223–232.
[17] G. Weiss and R. Alur, "Automata based interfaces for control and scheduling," in *Proc. 10th Int. Conf. Hybrid Syst.: Comput. Control*, 2007, pp. 601–613.
[18] R. Yeung, S.-Y. Li, and N. Cai, *Network Coding Theory*. Hanover, MA: Now Publishers, 2006.
[19] D. D. Siljak, *Decentralized Control of Complex Systems*. New York: Academic Press, 1991.
[20] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: A cross-layer platform for real-time embedded wireless networks," *Real-Time Syst. J.*, vol. 37, no. 3, pp. 183–231, 2007.
[21] *Nano-RK Sensor Real-Time Operating System*, Std., 2011 [Online]. Available: http://nanork.org
[22] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. New York: Springer, 2005, pp. 115–148.
[23] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-link: A time-synchronized link protocol for energy-constrained multi-hop wireless networks," in *Proc. 3rd Annu. IEEE Commun. Soc. Sensor Ad Hoc Commun. Netw. (SECON'06)*, 2006, pp. 402–411.
[24] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying wireless technology in real-time industrial process control," in *Proc. 14th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS'08)*, 2008, pp. 377–386.
[25] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst., ACM Sensys*, 2004, pp. 95–107.
[26] P. Seiler and R. Sengupta, "Analysis of communication losses in vehicle control problems," in *Proc. Amer. Control Conf.*, 2001, pp. 1491–1496.
[27] L. Bakule, "Decentralized control: An overview," *Annu. Rev. Control*, vol. 32, no. 1, pp. 87–98, 2008.
[28] V. Syrmos, C. Abdallah, P. Dorato, and K. Grigoriadis, "Static output feedback – A survey," *Automatica*, vol. 33, no. 2, pp. 125–137, 1997.
[29] V. Blondel and J. Tsitsiklis, "NP-hardness of some linear control design problems," *SIAM J. Control Optim.*, vol. 35, no. 6, pp. 2118–2127, 1997.
[30] O. Toker and H. Ozbay, "On the NP-hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback," in *Proc. Amer. Control Conf.*, 1995, pp. 2525–2526.
[31] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, 2000.
[32] L. El Ghaoui, F. Oustry, and M. Ait Rami, "A cone complementarity linearization algorithm for static output-feedback and related problems," *IEEE Trans. Autom. Control*, vol. 42, no. 8, pp. 1171–1176, Aug. 1997.
[33] P. Naghshtabrizi and J. P. Hespanha, "Anticipative and non-anticipative controller design for network control systems," in *Networked Embedded Sensing and Control*. New York: Springer, 2006, vol. 331, pp. 203–218.
[34] M. de Oliveira and J. Geromel, "Numerical comparison of output feedback design methods," in *Proc. 16th Amer. Control Conf.*, 1997, pp. 72–76.
[35] M. C. de Oliveira, J. E. Camino, and R. E. Skelton, "A convexifying algorithm for the design of structured linear controllers," in *Proc. 39th IEEE Conf. Decision Control*, 2000, pp. 2781–2786.
[36] O. Mangasarian and J. Pang, "The extended linear complementarity problem," *SIAM J. Matrix Anal. Appl.*, vol. 16, no. 2, pp. 359–368, 1995.
[37] N. Elia, "Remote stabilization over fading channels," *Syst. Control Lett.*, vol. 54, no. 3, pp. 237–249, 2005.
[38] C. L. Robinson and P. R. Kumar, "Sending the most recent observation is not optimal in networked control," in *Proc. 46th IEEE Conf. Decision Control*, 2007, pp. 334–339.
[39] M. Grant and S. Boyd, CVX: Matlab Software for Disciplined Convex Programming (Web Page and Software) 2009 [Online]. Available: http://stanford.edu/~boyd/cvx
[40] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam, "A simple distributed method for control over wireless networks," in *Proc. 1st Workshop Real-Time Wireless Ind. Appl.*, 2011, [CD ROM].
[41] N. A. Lynch, *Distributed Algorithms*. Tackett, WA: Morgan Kaufmann Publishers, Inc., 1996.
[42] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, "Linear matrix inequalities in system and control theory," in *Studies Applied Mathematics*. Philadelphia, PA: SIAM, 1994, vol. 15, p. 131.



Miroslav Pajic (S'06) received the Engineer Diploma degree (with highest honors) and the M.S. degree in electrical engineering from the School of Electrical Engineering, University of Belgrade, Belgrade, Serbia, in 2003 and 2007, respectively, and the M.S. degree in electrical engineering from the University of Pennsylvania, Philadelphia, in 2010, where he is currently pursuing the Ph.D. degree.

His research interests include real-time embedded systems, networked control systems, medical devices, and cyber-physical systems.

Shreyas Sundaram (M'09) received the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, in 2005 and 2009, respectively.

He is an Assistant Professor in the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He was a Postdoctoral Researcher in the GRASP Laboratory, University of Pennsylvania, Philadelphia, from 2009 to 2010. His research interests include secure and fault-tolerant control of distributed systems and networks, analysis of complex networks, structured system theory, and the application of graph theory to systems analysis.

Dr. Sundaram was a Finalist for the Best Student Paper Award at the 2007 and 2008 American Control Conferences.

George J. Pappas (S'90–M'91–SM'04–F'09) received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley (where he received the Eliahu Jury Award for Excellence in Systems Research), in 1998.

He is currently the Joseph Moore Professor of Electrical and Systems Engineering at the University of Pennsylvania, Philadelphia. He is a member of the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory and serves as the Deputy Dean for Research in the School of Engineering and Applied Science. His current research interests include hybrid and embedded systems, hierarchical control systems, distributed control systems, nonlinear control systems, with applications to robotics, unmanned aerial vehicles, biomolecular networks, and green buildings.

Dr. Pappas has received numerous awards, including the National Science Foundation (NSF) CAREER Award in 2002, the NSF Presidential Early Career Award for Scientists and Engineers in 2002, the 2009 George S. Axelby Outstanding Paper Award, and the 2010 Anotnio Ruberti Outstanding Young Researcher Prize.

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 Department of Electrical and Systems Engineering and Department of Computer and Information Science, University of Pennsylvania, Philadelphia. His interests are in real-time scheduling algorithms for networked embedded systems with applications in automotive systems, medical devices and control networks.