3-2010

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

Miroslav Pajic
University of Pennsylvania, pajic@seas.upenn.edu

Shreyas Sundaram
University of Waterloo

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

Rahul Mangharam
University of Pennsylvania, rahulm@seas.upenn.edu

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

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 (based on linear matrix inequalities) to determine the 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).

Keywords

Wireless Networks

Disciplines

Electrical and Computer Engineering | Engineering

Comments

Suggested Citiation:
Pajic, M., S. Sundaram, G. J. Pappas and R. Mangharam, The Wireless Control Network, Technical Report, March 2010.

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

Miroslav Pajic, Shreyas Sundaram, George J. Pappas and Rahul Mangharam

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 (based on linear matrix inequalities) to determine the 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).

This research has been partially supported by the NSF-CNS 0931239 and NSF-MRI Grant 0923518. Preliminary versions of some of these results are presented in [1].

M. Pajic, G. J. Pappas and R. Mangharam are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA 19014. Email: {pajic, pappasg, rahulm}@seas.upenn.edu. S. Sundaram is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, N2L 3G1. Email: ssundara@uwaterloo.ca

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 (when the plant contains hard-to-reach or dangerous areas), expensive (due to the labor and materials involved), space intensive (due to the large amounts of wiring required) 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 instance, wireless technology allows the construction of *modular* systems, in which a faulty module can be easily exchanged for a functioning backup module without the need to manually reconnect the module to the communication bus.[1] In addition, 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. 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.

While wireless networks possess many beneficial characteristics, they 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 factors 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], [8], [9], [10], [11], [12]. The vast majority of work in this area considers the case

[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.

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 (dropping packets with a certain probability, for example). 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, with severe resource constraints, allowing only a modest amount of computation and storage (examples of this are discussed in Section III). Furthermore, by adding one (or a few) specialized nodes capable of performing computationally expensive procedures, the control infrastructure becomes susceptible to failure on the 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.

Another key factor when implementing networked control algorithms is to minimize end-to-end delays in the network and to maximize network lifetime (by designing energy efficient algorithms). This is done via the use of *wireless link protocols* that have been developed by the wireless networking community. Broadly speaking, these protocols fall into two categories: synchronous or asynchronous (or loosely synchronous). 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. Furthermore, the lack of a communication schedule introduces possible packet collisions, causing nodes to retransmit multiple times and thereby expend a great deal of energy. In general, a higher degree of synchronization between nodes improves the energy efficiency of the network [14]. Also, from the perspective of analyzing system stability, synchronous protocols have the advantage that network induced delay (and therefore its impact on system stability) is known. 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 where only one plant being controlled with a multi-hop network, the task of constructing these schedules in order to meet strict end-to-end delay requirements is very complex (as shown in [15], [16]).

In this paper, we introduce the concept of a *Wireless Control Network* (WCN), which is a paradigm change for control over a wireless network. In a WCN the entire network *itself* acts as a controller: the computation is spread over all of the nodes, instead of assigning a particular node with the execution of the control procedure. 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 (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. The main contribution of this work is an algorithm for each node to follow in order to transform this wireless network into a bona-fide controller. Specifically, 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. The actuators of the plant also apply linear combinations of the states of the nodes in their neighborhood. Given a linear plant model and the topology of the wireless network, we devise a numerical design procedure that produces the coefficients of the linear combinations for each node (and actuator) to apply in order to stabilize the plant. We also show how our design procedure can be modified to account for packet drops in the network. As we will discuss in further detail in Section III, this approach to control over a wireless network has many benefits over traditional schemes (which assume that 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; the mathematical formulation that we provide in that section will set up the design procedures that we develop in subsequent sections. 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 provide an algorithm to determine an appropriate set of stabilizing linear combinations. In

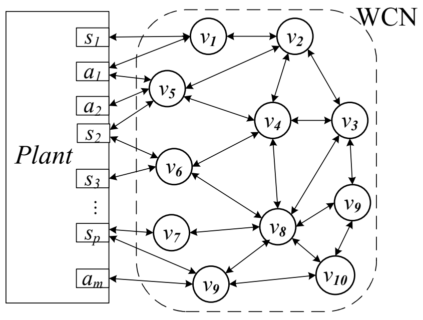

Figure 1. A multi-hop wireless control network used as a distributed controller.

Section V, we show how our algorithm can be modified to account for packet drops in the network. For the sake of pedagogy, we assume in both of these preceding sections that each node can only perform calculations on a single (scalar) value; in Section VI, we describe how our algorithms can be extended to handle more general nodes in the network. We provide examples of our scheme in Section VII. In Section VIII we discuss the relationship between the WCN and the traditional notions of delay introduced by the feedback loop, and show how the WCN is able to gracefully degrade under node failures. Finally, we finish in Section IX by describing some shortcomings of our scheme and pointing out some possible directions for future work.

A. 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 diag $(\cdot)$ indicates a square matrix with the quantities inside the brackets on the diagonal, and zeros elsewhere. The notation 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. The 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]

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

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 $\mathbf{y}[k] = \begin{bmatrix} y_1[k] & y_2[k] & \dots & y_p[k] \end{bmatrix}^T$ contains measurements of the plant state vector $\mathbf{x}[k]$ provided by the sensors $s_1, \dots, s_p$. The input vector $\mathbf{u}[k] = \begin{bmatrix} u_1[k] & u_2[k] & \dots & u_m[k] \end{bmatrix}^T$ corresponds to the signals applied to the plant by actuators $a_1, \dots, a_m$.

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 $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ 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 $\mathcal{S} = \{s_1, s_2, \dots, s_p\}$ corresponds to the plant's sensors, while $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$ corresponds to the plant's actuators. Define the edge sets

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

[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 *mean square stable*. For the purposes of clarity, we will therefore omit the noise terms in our discussion.

[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 results, we will 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.

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

We then obtain $\bar{\mathcal{G}} = \{\mathcal{V} \cup \mathcal{S} \cup \mathcal{A}, \mathcal{E} \cup \mathcal{E}_I \cup \mathcal{E}_O\}$. Let N_l denote the number of links in $\bar{\mathcal{G}}$ ($N_l = |\mathcal{E} \cup \mathcal{E}_I \cup \mathcal{E}_O|$). We can also define an injective 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 $\Omega(a, b)$ for convenience.

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), the WCN employs a fully distributed control scheme where the entire network itself acts as a controller. To see how this is achieved, suppose that 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]

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

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]

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

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

[4]The proposed scheme is very similar to the algorithms used in linear network coding (in information theory), where nodes in a network do not merely forward received messages - a node combines the received messages before they are retransmitted.

[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, only one node in the actuator's neighborhood informs it about its state. Thus, in this case for each $i \in \{1, \cdots, p\}$, $g_{ij} = 1$ for exactly one $j \in \{1, \cdots N\}$, and all other weights are equal to zero.

vector $\mathbf{z}[k] = \begin{bmatrix} z_1[k] & z_2[k] & \dots & z_N[k] \end{bmatrix}^T$, the behavior of the entire network can be represented as:

$$\mathbf{z}[k+1] = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix} \mathbf{z}[k] + \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1p} \\ h_{21} & h_{22} & \cdots & h_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{Np} \end{bmatrix} \mathbf{y}[k] \triangleq \mathbf{Wz}[k] + \mathbf{Hy}[k] \ ,$$

$$\mathbf{u}[k] = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{mN} \end{bmatrix} \mathbf{z}[k] \triangleq \mathbf{Gz}[k]$$

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 $i \in \{1, \dots, N\}$, $w_{ij} = 0$ if $v_j \notin \mathcal{N}_{v_i}$, $h_{ij} = 0$ if $s_j \notin \mathcal{N}_{v_i}$, and $g_{ij} = 0$ if $v_j \notin \mathcal{N}_{a_i}$. 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. If we denote the overall system state by $\hat{\mathbf{x}}[k] = \begin{bmatrix} \mathbf{x}[k]^T & \mathbf{z}[k]^T \end{bmatrix}^T$, the closed-loop system becomes:

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

In the next section, we describe the implementation advantages of the above control mechanism, and then we spend the rest of the paper describing algorithms to find an element of Ψ that causes the matrix $\hat{\mathbf{A}}$ to be Schur[7] (whenever such an element exists). We first consider the case with reliable communication links (without any communication packet drops) and provide a numerical algorithm based on Linear Matrix Inequalities (LMIs). Next, we show that the procedure can be used to accommodate Bernoulli link failures in the network, where the linear combinations are chosen to ensure mean square stability (MSS) under a sufficiently low probability of packet loss. Finally, we discuss the extension of our design procedure to handle nodes with more relaxed memory and computational constraints (which we model by allowing nodes to have vector states of limited size).

[7]A square matrix is Schur if all of its eigenvalues are inside the unit circle.

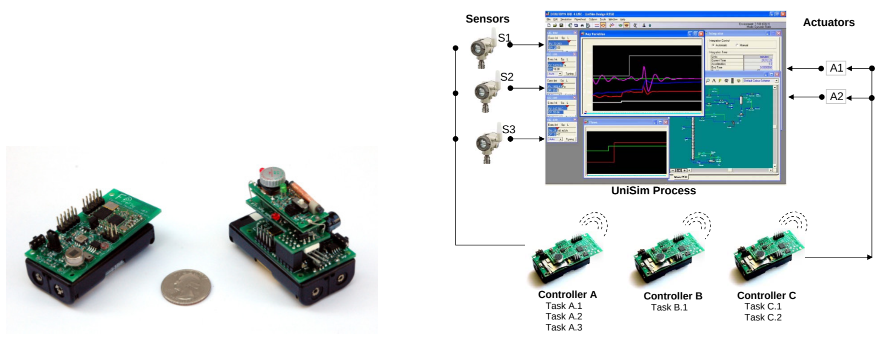

Figure 2. (a) On the left of the coin, a low-power FireFly node; on the right, a FireFly node with an add-on AM receiver for hardware-based out-of-band synchronization; (b) An example of FireFly nodes in a process-in-the-loop simulation using the Honeywell Unisim process (plant) modeling tool.

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.

1. 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),[8] using very simple, periodic tasks executed on a real-time operating system (such as nano-RK [18] or TinyOS [19]). Furthermore, unlike in 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 that is 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 [20] or wirelessHART [2]). For example, if the node's state

[8]These nodes usually contain an 8-bit or 16-bit microcontroller operating on 8 MHz (or lately 16MHz) 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 60mW when both CPU and radio are active, or 3uW when the CPU and radio are in sleep mode [17].

$z_i[k]$ is a 16 bit 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.

The requirement that all nodes in the WCN be at least loosely synchronized introduces some small overhead into our control scheme. However, this 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 [14] and wirelessHART [21]) or with hardware synchronization (e.g., using an additional AM receiver, as shown in Fig. 2(a) [14]).[9] As we will discuss in the next point, the computation of communication and computation schedules for synchronized communication in the WCN is much simpler than in traditional networked control schemes.

2. 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). Thus, if d_i is the maximal degree of the interference graph,[10] a *static* conflict-free schedule can be derived using graph coloring, with at most 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

[9]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 [22]). 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).

[10]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).

the end-to-end delay, causing the minimal sampling period to directly depend on the network diameter.

3. 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 $\rightarrow$ channel $\rightarrow$ controller/estimator $\rightarrow$ channel $\rightarrow$ actuator" setup that is commonly adopted in networked control design. Even in the few works that consider networked control over arbitrary topologies ([11], [12]), an assumption is made that there is a single actuation point 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 will contain multiple actuation and sensing locations, in which case it is no longer clear that the conclusions in those papers will 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.

4. 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 them from the various sensors on the different plants. This requires the calculation of a new collision-free schedule for the *entire* network.[11] 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 [15]. Finally,

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

it is necessary to have software support that allows run-time changes of the control procedures executed on the nodes (i.e., defining and activating new procedures). Since these procedures are usually implemented as tasks executed on top of a real-time operating system, the software support has to enable run-time instantiation (i.e.. 'download' of new tasks) and activation of these tasks in the light of node/link faults and topology changes, as in [23], [4].

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, rather than recalculating a stabilizing set of linear combinations that would work for all plants simultaneously, 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.[12] Upon reception of the P different states from each of its neighbors, composing all of its P control schemes, 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.

Furthermore, as the newly added plants are controlled with the *same* algorithm (i.e., for each plant, each node performs exactly the same set of actions, but with different coefficients), the 'software updates' for each node only need the new set of coefficients and a command to activate periodic execution of the linear procedure. 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 [15], [24].

IV. Stabilizing The Closed-loop System

In this section we present a numerical algorithm that can be used to design the WCN (i.e., determine the linear combinations that should be used by each node and actuator to stabilize the closed-loop system). From equation (4), the closed-loop system is stable if the matrix $\hat{\mathbf{A}} =$

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

$\hat{\mathbf{A}}(\mathbf{W},\mathbf{H},\mathbf{G})$ 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,

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

This 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 LMI via an appropriate transformation of the system matrices (e.g., as done in [25]). However, the fact that the matrices are *structured* in our framework prevents us from directly applying these standard procedures.[13] While this direct attempt to cast the controller design problem as a LMI does not work in our context, the following alternative characterization of stability of structured systems from [27] offers a solution.

Theorem 1: ([27]) A matrix $\hat{\mathbf{A}}$ is Schur if and only if there exist symmetric, positive-definite matrices $\mathbf{X}$ and $\mathbf{Y}$ such that $\left[\begin{smallmatrix} \mathbf{X} & \hat{\mathbf{A}}^T \\ \hat{\mathbf{A}} & \mathbf{Y} \end{smallmatrix}\right] \succ 0$, $\mathbf{X} = \mathbf{Y}^{-1}$. □

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. However, as pointed out in [27], constraints of this form commonly occur in the design of static output feedback controllers, and there are various numerical methods to address this issue. One particularly appealing approach that is suggested in [28], [29] is to approximate the constraint $\mathbf{X} = \mathbf{Y}^{-1}$ with an optimization problem using the following lemma (it is used without proof in the aforementioned papers, so we present its proof here for completeness).

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

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

and the optimal cost of the problem is n.

[13]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 [26]. 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.

Proof: **Necessity**: If there exist $\mathbf{X_0}, \mathbf{Y_0} \in \mathbb{S}^n_{++}$ such that $\mathbf{X_0} = \mathbf{Y_0}^{-1}$ then $\mathbf{X_0}, \mathbf{Y_0}$ are feasible points of problem (P) and $tr(\mathbf{X_0}\mathbf{Y_0}) = n$. Also for all (P) feasible $\mathbf{X}, \mathbf{Y}$

$$\mathbf{X} \succeq \mathbf{Y}^{-1} \Rightarrow \mathbf{Y}^{1/2}\mathbf{X}\mathbf{Y}^{1/2} \succeq \mathbf{I} \Rightarrow tr(\mathbf{Y}^{1/2}\mathbf{X}\mathbf{Y}^{1/2}) = tr(\mathbf{X}\mathbf{Y}) \geq n.$$

Thus, the matrices $\mathbf{X_0}, \mathbf{Y_0}$ are the optimal points.

Sufficiency: Assume that the matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{S}^n_{++}$ are optimal points for the problem (P) and $tr(\mathbf{X}\mathbf{Y}) = n$. Then for $\tilde{\mathbf{X}} = \mathbf{Y}^{1/2}\mathbf{X}\mathbf{Y}^{1/2}$ we have

$$\tilde{\mathbf{X}} \succeq \mathbf{I} \Rightarrow \mathbf{e}_i^T\tilde{\mathbf{X}}\mathbf{e}_i \geq \mathbf{e}_i^T\mathbf{e}_i \Rightarrow \tilde{X}_{i,i} \geq 1 \Rightarrow \tilde{X}_{i,i} = 1$$

since $\sum_{i=1}^n \tilde{X}_{i,i} = n$. Similarly,

$$\tilde{\mathbf{X}} \succeq \mathbf{I} \Rightarrow (\mathbf{e}_i \pm \mathbf{e}_j)^T\tilde{\mathbf{X}}(\mathbf{e}_i \pm \mathbf{e}_j) \geq (\mathbf{e}_i \pm \mathbf{e}_j)^T(\mathbf{e}_i \pm \mathbf{e}_j)$$

$$\Rightarrow \tilde{X}_{i,i} + \tilde{X}_{j,j} \pm 2\tilde{X}_{i,j} \geq 2 \Rightarrow \pm\tilde{X}_{i,j} \geq 0 \Rightarrow \tilde{X}_{i,j} = 0.$$

Therefore, $\tilde{\mathbf{X}} = \mathbf{I} \Rightarrow \mathbf{X}\mathbf{Y} = \mathbf{I}$. ■

Using the Schur complement, the constraint $\mathbf{X} \succeq \mathbf{Y}^{-1}$ in the above lemma can be readily transformed to the form $\left[\begin{smallmatrix}\mathbf{X} & \mathbf{I}\\ \mathbf{I} & \mathbf{Y}\end{smallmatrix}\right] \succeq 0$. Theorem 1 and Lemma 1 yield 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 constraint $(\mathbf{W}, \mathbf{H}, \mathbf{G}) \in \Psi$ if and only if the following optimization problem

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

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

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

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 [29], [30], a problem of the form (P) from Lemma 1 (or (5)-(7) from Corollary 1) is known as a *cone complementarity problem* (CCP). For such problems, El Ghaoui *et al.* [29] showed that the nonconvex function tr$(\mathbf{X}\mathbf{Y})$ can be replaced with a linear approximation

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

for any given matrices $\mathbf{X}_0$ and $\mathbf{Y}_0$. With this insight, [28], [29] showed that an iterative algorithm can be used to minimize tr$(\mathbf{XY})$, while ensuring satisfaction of LMI constraints. For our scheme, 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

$$\min tr(\mathbf{Y_k X_{k+1}} + \mathbf{X_k Y_{k+1}})$$

$$\begin{bmatrix} \mathbf{X}_{k+1} & \hat{\mathbf{A}}_{k+1}^T \\ \hat{\mathbf{A}}_{k+1} & \mathbf{Y}_{k+1} \end{bmatrix} \succ 0, \quad \begin{bmatrix} \mathbf{X_{k+1}} & \mathbf{I} \\ \mathbf{I} & \mathbf{Y_{k+1}} \end{bmatrix} \succeq 0,$$

$$\hat{\mathbf{A}}_{k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{G}_{k+1} \\ \mathbf{H}_{k+1}\mathbf{C} & \mathbf{W_{k+1}} \end{bmatrix},$$

$$(\mathbf{W_{k+1}}, \mathbf{H}_{k+1}, \mathbf{G}_{k+1}) \in \Psi, \;\; \mathbf{X}_{k+1}, \mathbf{Y}_{k+1} \in \mathbb{S}_{++}^{n+N}.$$

3. If the matrix

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

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

In [29] 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 $2(n + N)$. $\square$

Note that while each iteration of the above algorithm is a convex optimization problem (which can be efficiently solved using standard LMI toolboxes), we do not currently have a characterization of the number of iterations required for the algorithm to converge.[14]

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

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

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 a design procedure that can guarantee mean-square stability (MSS) of the closed loop system, defined as:

Definition 1: ([25], [31]) 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 [10], [9]. 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], [32] 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

[14]In fact, 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.

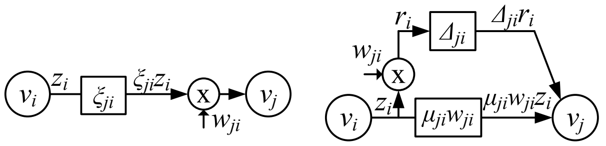

Figure 3. Remote control over fading channel; (a) A link between nodes v_i and v_j; (b) Link transformation into a robust control form.

of a designated controller within the network. It is worth noting that the papers [12] and [32] 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 [31], where it was shown that such networks can be cast in a robust control framework, allowing an elegant approach to analysis and design. Due to its ability to capture arbitrary network topologies, we will adapt this approach to the problem of designing a wireless control network with unreliable links. In the framework of [31], a communication link is modeled over time as a memoryless, discrete, independent and identically distributed (IID) random process ξ,[15] which maps each transmitted value $t_x[k]$ into a received value $r_x[k] = \xi[k]t_x[k]$.[16] 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)). In the rest of the paper we will also denote this link as $t = \Omega(v_i, v_j)$ 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 $\sigma_t^2 = \mathbf{E}[(\xi_t[k] - \mu_t)^2]$.

Following the approach in [31], 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

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

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

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

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

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:

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

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

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

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

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

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

$$\left[\mathbf{J}_u^{dst}\right]_{i,t} = \begin{cases} 1, \text{ if } i \leq m,\ \exists v_j, \Omega(v_j, a_i) = t \\ 0, \text{ else} \end{cases}$$

$$\left[\mathbf{J}^{dst}\right]_{i,t} = \begin{cases} 1, \text{ if } 1 \leq i \leq N,\ \exists v_j, \Omega(v_j, v_i) = t \text{ or } \exists s_j, \Omega(s_j, v_i) = t, \\ 0, \text{ else.} \end{cases} \tag{11}$$

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:

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

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 [31], we obtain the following result.[17]

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

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

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

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.

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 $2(n+N)$. □

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

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, $(k \geq 0)$ from $\mathbf{X_k}, \mathbf{Y_k}$ obtain the matrices $\mathbf{X_{k+1}}, \mathbf{Y_{k+1}}, \mathbf{W}_{\mu,\mathbf{k+1}}, \mathbf{H}_{\mu,\mathbf{k+1}}, \mathbf{G}_{\mu,\mathbf{k+1}}$ and a vector $\alpha_{k+1} \in \mathbb{R}^{N_l}$ by solving the following LMI problem

$$\min tr(\mathbf{Y_k}\mathbf{X_{k+1}} + \mathbf{X_k}\mathbf{Y_{k+1}})$$

$$\begin{bmatrix} \mathbf{X}_{k+1} - (\hat{\mathbf{J}}^{dst})^T diag\{\alpha_{k+1}\}(\hat{\mathbf{J}}^{dst})^T & \hat{\mathbf{A}}_{\mu,k+1} \\ \hat{\mathbf{A}}_{\mu,k+1}^T & \mathbf{Y}_{k+1} \end{bmatrix} \succ 0,$$

$$\begin{bmatrix} \mathbf{X_{k+1}} & \mathbf{I} \\ \mathbf{I} & \mathbf{Y_{k+1}} \end{bmatrix} \succeq 0,$$

$$\hat{\mathbf{A}}_{\mu,k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{B}\mathbf{G}_{\mu,\mathbf{k+1}} \\ \mathbf{H}_{\mu,\mathbf{k+1}}\mathbf{C} & \mathbf{W}_{\mu,\mathbf{k+1}} \end{bmatrix},$$

$$\begin{bmatrix} \alpha_{i,k+1} & \sigma_i(\hat{\mathbf{J}}_{k+1}^{or})_i \\ \sigma_i(\hat{\mathbf{J}}_{k+1}^{or})_i^T & \mathbf{Y}_{k+1} \end{bmatrix} \succeq 0, \ 1 \leq i \leq N_l,$$

$$(\mathbf{W}_{\mu,\mathbf{k+1}}, \mathbf{H}_{\mu,k+1}, \mathbf{G}_{\mu,k+1}) \in \Psi, \ \ \mathbf{X}_{k+1}, \mathbf{Y}_{k+1} \in \mathbb{S}_{++}^{n+N}$$

3. Stop the algorithm if the following conditions are true

$$\mathbf{X}_{k+1} \succ \hat{\mathbf{A}}_{\mu,k+1}\mathbf{X}_{k+1}\hat{\mathbf{A}}_{\mu,k+1}^T + \hat{\mathbf{J}}^{dst} diag\{\alpha_{k+1}\}(\hat{\mathbf{J}}^{dst})^T$$

$$\alpha_{i,k+1} \geq \sigma_i^2(\mathbf{J}_{k+1}^{or})_i\mathbf{X}_{k+1}(\mathbf{J}_{k+1}^{or})_i^T, \ 1 \leq i \leq N_l.$$

Otherwise, set $k = k + 1$ and go to step **2**.

Remark 1: It is worth noting that any matrices $\hat{\mathbf{A}}_\mu, \mathbf{X}, \mathbf{J}^{dst}, \mathbf{J}^{or}$ and vector α that satisfy the constraints from (13) for the link quality vector $\bar{\sigma} = [\bar{\sigma}_1, \ldots, \bar{\sigma}_{N_l}]^T$, also satisfy the constraints for any vector σ such that $\sigma \preceq \bar{\sigma}$ (where "$\preceq$" implies elementwise inequality). Thus, finding a vector $\underline{\sigma} = \sigma \mathbf{1}, \sigma \in \mathbb{R}$ for which there exists matrices $\mathbf{W}, \mathbf{G}, \mathbf{H}$ 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 $\underline{\sigma}$. The largest value of σ for which the system is MSS can be found by allowing $\sigma \in \mathbb{R}$ 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 [25]). □

Remark 2: 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. 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 Eq. 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 $(v_i^j, j \in \{1, ..., n_i\})$, where each node would maintain a scalar value as its state. This would allow the use of the **Algorithm** 2 to determines 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,[18] 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 $C_s = \sum_{t=1}^{N_l} c'_t$. As in Section V, the overall system is given by equations (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.[19] In addition, $\mathbf{r}[k] \in \mathbb{R}^{C_s}$ and $\mathbf{\Delta}[k] = \mathbf{blkdiag}(\{\mathbf{\Delta}_t[k]\}_{t=1}^{N_l})$, where $\mathbf{blkdiag}$ is a block-diagonal operator and $\mathbf{\Delta}_t[k] = \Delta_t[k]\mathbf{I}_{c'_t\times c'_t}$. This brings us to the following theorem (the proof is provided in the Appendix).

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

$$\begin{aligned} \mathbf{X} &\succ \hat{\mathbf{A}}_\mu \mathbf{X}\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst}\cdot \mathbf{blkdiag}\{\alpha_1,\dots,\alpha_{N_l}\}\cdot(\hat{\mathbf{J}}^{dst})^T \\ \alpha_i &\succeq \sigma_i^2(\hat{\mathbf{J}}^{or})_i\cdot\mathbf{X}\cdot(\hat{\mathbf{J}}^{or})_i^T,\ \forall i\in\{1,\dots,N_l\} \end{aligned} \quad (14)$$

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 equivalent to **Algorithm** 2 can be used to determine the (vector-valued) update for each node to apply in order to guarantee MSS.

[18]This is a reasonable assumption, since a transmission packet contains up to 128 bytes even in (bandwidth limited) 802.15.4 networks.

[19]As a substitute for scalars w_t, g_t or h_t, matrices $\mathbf{w}_t, \mathbf{g}_t$ and $\mathbf{h}_t$ should be used. Also, instead of the scalar '1' in (11), the identity matrix I_{n_i} should be used, where n_i is the state vector size for the link's receiving node.

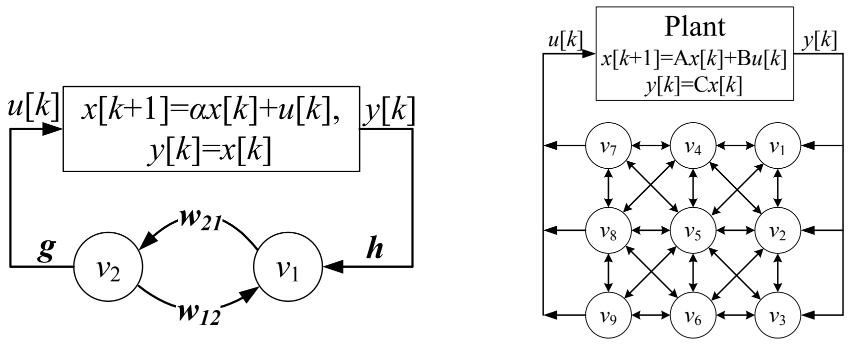

Figure 4. Two examples of WCNs; (a) A plant with a scalar state controlled by a WCN where each node maintains a scalar state; (b) A single-input-single output plant with $\mathbb{R}^3$ state controlled by a WCN where each node maintains a scalar state.

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 $\sigma^2 = p(1-p)$). Obviously, for $\alpha > 1$ the plant is unstable, even for reliable communication links ($p = 0$). To solve the optimization problem from **Algorithm** 2 we used `CVX`, a package for specifying and solving convex programs [33]. For $\alpha = 2$ and $p = 0.5\%$, if each node maintains a scalar state, **Algorithm** 2 converges to a stable configuration

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

after 51 iterations.[20]

Using the bisection method described in the Remark 1, 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 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}; a possible hypothesis for this is that the single link between node v_N and the actuator is the bottleneck for stability. Also, adding more powerful nodes does not increases the robustness of the system to packet drops in the wireless network.

[20]The number of iterations needed before the algorithm converges to a stable configuration depends on initial points $\mathbf{X_0}, \mathbf{Y_0}$.

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

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

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

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

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:

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

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.056 & -1.620 & 0.233 \end{bmatrix}. \tag{16}$$

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 1 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

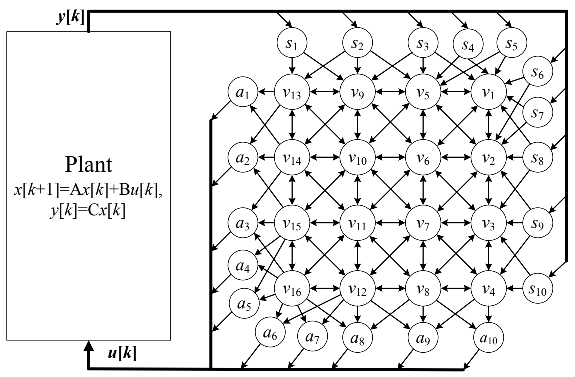

Figure 5. An example of plant with 30 stated controlled by a WCN consisted of mesh network with 16 nodes.

Fig. 4(b), where all nodes in the WCN maintain two scalar states and calculate the first update using the coefficients from (16), 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 also 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 no new computation of communication or computation schedules is 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 [33]. 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 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 essentially 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

$$\begin{bmatrix} x[k+1] \\ z_1[k+1] \\ z_2[k+1] \end{bmatrix} = \begin{bmatrix} \alpha & 0 & g \\ h & w_{11} & w_{12} \\ 0 & w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x[k] \\ z_1[k] \\ z_2[k] \end{bmatrix} , \tag{17}$$

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 $g = h = 1$, $w_{11} = 0$, $w_{12} = \frac{1}{\alpha}$, $w_{21} = -\alpha^3$ 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 (17). 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 $|\alpha| < \frac{3}{2}$. 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 must be the case that

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

from which we see that $-3 - \alpha < w_{22} < 3 - \alpha$ and $-\frac{3}{\alpha} < w_{22} < \frac{3}{\alpha}$ 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 $\alpha \geq \frac{3+\sqrt{21}}{2}$). 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 is not satisfactory, 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.[21] 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* [34]). 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.

IX. Conclusion and Future Work

We have introduced the concept of a *Wireless Control Network*, where the network itself acts as a controller. 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.

[21]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.

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 [32], [12]), 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. Also, multiple failures in any given neighborhood might impose a large amount of overhead on the part of the remaining nodes. A more robust scheme to handle different fault models in nodes is desirable.

• We do not currently have a characterization of the number of iterations required for our algorithms to converge. While the number of variables in our optimization algorithms scale well with the number of nodes and links (quadratically and linearly, respectively), our experiments show that the number of iterations required for the algorithms to converge to a stabilizing configuration (if one exists) can be quite large, and is dependent on the initial feasible points. A quantitative metric of the complexity of solving these problems would be very useful.

• This paper assumes that the topology of the WCN is specified *a priori*, and presents a numerical algorithm to design the linear 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.

APPENDIX
PROOF OF THEOREM 5

Proof: (A slight generalization of the approach in [31] is used to prove the Theorem.) Consider a linear system of the form:

$$\hat{\mathbf{x}}[k+1] = \hat{\mathbf{A}}_{\mu}\hat{\mathbf{x}}[k] + \hat{\mathbf{J}}^{dst}\mathbf{\Delta}[k]\mathbf{r}[k], \tag{18}$$

$$\mathbf{r}[k] = \hat{\mathbf{J}}^{or}\hat{\mathbf{x}}[k], \tag{19}$$

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

$$\begin{aligned}\mathbf{M}[k+1] = \mathbb{E}\{\hat{x}[k+1]\hat{x}[k+1]^T\} &= \hat{\mathbf{A}}_\mu \mathbb{E}\{\hat{\mathbf{x}}[k]\hat{\mathbf{x}}[k]^T\}\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst}\mathbb{E}\{\mathbf{\Delta}[k]\mathbf{r}[k]\hat{\mathbf{x}}[k]^T\}\hat{\mathbf{A}}_\mu^T \\ &+ \hat{\mathbf{A}}_\mu \mathbb{E}\{\hat{\mathbf{x}}[k]\mathbf{r}[k]^T\mathbf{\Delta}[k]^T\}\hat{\mathbf{J}}^{dst^T} + \hat{\mathbf{J}}^{dst}\mathbb{E}\{\mathbf{\Delta}[k]\mathbf{r}[k]\mathbf{r}[k]^T\mathbf{\Delta}[k]\}\hat{\mathbf{J}}^{dst^T} \\ &= \hat{\mathbf{A}}_\mu \mathbf{M}[k]\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst}\mathbb{E}\{\mathbf{\Delta}[k]\mathbf{r}[k]\mathbf{r}[k]^T\mathbf{\Delta}[k]\}\hat{\mathbf{J}}^{dst^T}. \end{aligned} \tag{20}$$

The second and third terms in the second 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 $\mathbf{T} = \mathbb{E}\{\mathbf{\Delta}[k]\mathbf{r}[k]\mathbf{r}[k]^T\mathbf{\Delta}[k]\}$, and note that $\mathbf{r}_t[k] = (\hat{\mathbf{J}}^{or})_t\hat{\mathbf{x}}[k]$, where $(\hat{\mathbf{J}}^{or})_t$ is defined as in the Theorem statement (denotes the t^{th} block-row of the matrix $\hat{\mathbf{J}}^{or}$). Since $\mathbf{\Delta}[k] = \mathbf{blkdiag}(\{\mathbf{\Delta}_t[k]\}_{t=1}^{N_l})$ and $\mathbf{\Delta}_t[k] = \Delta_t[k]\mathbf{I}_{c'_t\times c'_t}$, the $(t_1,t_2)^{th}$ block submatrix of $\mathbf{T}$ is given by $\mathbf{T}_{t_1t_2}[k] \triangleq \mathbb{E}\{\mathbf{\Delta}_{t_1}[k]\mathbf{r}_{t_1}[k]\mathbf{r}_{t_2}[k]^T\mathbf{\Delta}_{t_2}[k]\}$. 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

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

where $\alpha_t \in \mathbf{R}^{c'_t\times c'_t}$. Therefore we have:

$$\begin{aligned}\mathbf{M}[k+1] &= \hat{\mathbf{A}}_\mu\mathbf{M}[k]\hat{\mathbf{A}}_\mu^T + \hat{\mathbf{J}}^{dst}\cdot\mathbf{blkdiag}\{\alpha_t,\ldots,\alpha_{N_l}\}\cdot\hat{\mathbf{J}}^{dst^T} \\ \alpha_t &= \sigma_t^2(\hat{\mathbf{J}}^{or})_t\mathbf{M}[k](\hat{\mathbf{J}}^{or})_t^T.\end{aligned} \tag{21}$$

This is essentially of the same form as the equations in Theorem 6.4 from [31] (except for the fact that the α_t variables are matrices in our case). Therefore, 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 $\alpha_i \in \mathbb{R}^{c'_i\times c'_i},\ i\in\{1,\ldots,N_l\}$ that satisfy the theorem's conditions [31], [35]. ■

Acknowledgments

The authors thank Paul McLaughlin and Alexander Chernoguzov from Honeywell Process Solutions and Jerome Le Ny from the University of Pennsylvania for fruitful discussions during the preparation of this paper.

REFERENCES

[1] M. Pajic, S. Sundaram, J. Le Ny, G. J. Pappas, and R. Mangharam, "The wireless control network: Synthesis and robustness," in *Proceedings of the 49th IEEE Conference on Decision and Control*, 2010, submitted.

[2] "Why WirelessHART?" White Paper, HART Communication Foundation, Oct. 2007.

[3] S. Amidi and A. Chernoguzov, "Wireless process control network architecture overview," White Paper, Honeywell International Inc., Mar. 2009.

[4] M. Pajic and R. Mangharam, "Embedded Virtual Machine for Robust Wireless Control and Actuation," *Proc. of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, 2010.

[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. 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. CRC Press, 2005.

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

[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," *Proceedings of the 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," *Proceedings of the 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 Journal on Selected Areas in Communications*, 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 Transactions on Automatic 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," *Communications in Information and Systems*, vol. 3, no. 4, pp. 279–304, Apr. 2003.

[14] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks," *Proceedings of the IEEE Conference on Sensors, Mesh and Ad-hoc Communication and Networks, SECON*, 2006.

[15] R. Alur, A. D'Innocenzo, K. Johansson, G. Pappas, and G. Weiss, " Modeling and Analysis of Multi-hop Control Networks," *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, 2009.

[16] G. Weiss, A. D'Innocenzo, R. Alur, K. Johansson, and G. Pappas, "Robust stability of multi-hop networks," *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009.

[17] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: A Cross-layer Platform for Real-time Embedded Wireless

Networks," *Real-Time System Journal*, 2007.

[18] "The nano-RK Sensor Real-Time Operating System," http://nanork.org.

[19] 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*, 2005.

[20] "ISA100.11a: Wireless systems for industrial automation: Process control and related applications," Draft standard, 2009.

[21] 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 *Proceedings of the 14th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, 2008.

[22] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. of the 2nd International conference on Embedded networked sensor systems, ACM Sensys*, 2004, pp. 95–107.

[23] S. Graham, G. Baliga, and P. Kumar, "Abstractions, architecture, mechanisms, and a middleware for networked control," *Automatic Control, IEEE Transactions on*, vol. 54, no. 7, pp. 1490 –1503, july 2009.

[24] G. Weiss and R. Alur, "Automata based interfaces for control and scheduling," *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control, HSCC*, 2007.

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

[26] L. Bakule, "Decentralized control: An overview," *Annual Reviews in Control*, vol. 32, no. 1, pp. 87–98, 2008.

[27] M. C. de Oliveira, J. E. Camino, and R. E. Skelton, "A convexifying algorithm for the design of structured linear controllers," *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000.

[28] P. Naghshtabrizi and J. P. Hespanha, "Anticipative and non-anticipative controller design for network control systems," in *Networked Embedded Sensing and Control*, ser. Lect. Notes in Contr. and Inform. Sci. Springer, 2006, vol. 331, pp. 203–218.

[29] L. El Ghaoui, F. Oustry, and M. Ait Rami, "A cone complementarity linearization algorithm for static output-feedback and related problems," *IEEE Transactions on Automatic Control*, 1997.

[30] O. Mangasarian and J. Pang, "The extended linear complementarity problem," *SIAM Journal on Matrix Analysis and and Applications*, 1995.

[31] N. Elia, "Remote stabilization over fading channels," *Systems & Control Letters*, vol. 54, pp. 237–249, 2005.

[32] C. L. Robinson and P. R. Kumar, "Sending the most recent observation is not optimal in networked control," in *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007, pp. 334–339.

[33] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming (web page and software). http://stanford.edu/ boyd/cvx," June 2009.

[34] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[35] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, "Linear matrix inequalities in system and control theory," in *SIAM Studies in Applied Mathematics*, 1994, vol. 15, p. 131.