Passivity-Based Control Design for Cyber-Physical Systems

Xenofon Koutsoukos*, Nicholas Kottenstette†, Joe Hall*, Panos Antsaklis†, Janos Sztipanovits*

*ISIS/Vanderbilt University †University of Notre Dame

***Abstract*— Real-life Cyber-Physical Systems (CPSs), such as automotive vehicles, building automation systems, and groups of unmanned air vehicles are monitored and controlled by networked control systems. The overall system dynamics emerges from the interaction among physical dynamics, computational dynamics, and communication networks. Network uncertainties such as time-varying delay and packet loss cause significant challenges that probihit the application of traditional component-based design methods. This paper proposes a passive control architecture for designing CPSs that are insensitive to network uncertainties. The proposed method improves orthogonality across the controller design and implementation design layers with respect to network uncertainties, thus empowering model-driven development. The paper presents the architecture for a simplified system consisting of a robotic manipulator controlled by a digital controller over a wireless network and simulation results that show that the system is insensitive to time-varying network delays.**

I. Introduction

The heterogeneous composition of computing, sensing, actuation, and communication components has enabled a modern grand vision for real-world Cyber Physical Systems (CPSs). Real-world CPSs, such as automotive vehicles, building automation systems, and groups of unmanned air vehicles are monitored and controlled by networked control systems and the overall system dynamics emerges from the interaction among physical dynamics, computational dynamics, and communication networks. Design of CPSs requires controlling real-world system behavior and interactions in dynamic and uncertain conditions. This paper, in particular, aims at applications that integrate computational and physical devices using wireless networks such as medical device networks, groupls of unmanned vehicles, and transportation networks.

This work is motivated by the rapidly increasing use of network control system archictures in constructing real-world CPSs and aims at addressing fundamental problems caused by networks effects, such as time-varying delay, jitter, data rate limitations, and packet loss. To deal with these implementation uncertainties, we propose a model-design flow on top of passivity, a very significant concept from system theory [1]. A precise mathematical definition requires many technical details, but the main idea is that a passive system cannot apply an infinite amount of energy to its environment. The inherent safety that passive systems provide is fundamental in building systems that are insensitive to implementation uncertainties. Passive systems have been exploited for the design of diverse systems such as smart exercise machines [2], teleoperators [3], digital filters [4], and networked control systems [5]–[7].

The main idea of our approach is that by imposing passivity constraints on the component dynamics, the design becomes insensitive to network effects, thus establishing orthogonality (with respect to network effects) across the controller design and implementation design layers. This separation of concerns empowers the model based design process to be extended for networked control systems. It should be noted that passive structures have additional advantages with regard to robustness in the presence of finite length representations and saturation [4] but this paper focuses on network effects that is one of the most significant concerns in the development of CPSs.

The paper first discusses challenges in applying component-based system design techniques for CPSs in Section II. The challenges stem from the heterogeneity of CPSs not only in terms of their components but also in terms of essential design requirements. For example, control design typically depends on assumptions that neglect the network uncertainties. A major problem for model-based design is decoupling control specification from implementation uncertainties. To address this problem, we propose a passive control architecture that accounts for the effects of network uncertainties.

After a brief backgound on passivity in Section III, we present the architecture for a simplified system consisting of a robotic manipulator controlled by a digital controller over a wireless network in Section IV. We focus on the technical details required for implementing the architecture and we outline the technical results that ensure passivity and stability of the overall system. Section V evaluates the passive control design for a typical 6 degree-of-freedom robotic arm controlled by a digital controller over a 802.llb wireless network. We present simulation results based on a detailed model that contains components for the robotic arm, the wireless network, and the digital controller that show that the system operation is insensitive to time-varying delays. We also compare the proposed method with a standard non-passive controller to illustrate the advantages of passive control design. Finally, Section VI presents the main directions of our future work.

II. Model-Based Control Design for CPSs

Building systems from components is central in all engineering disciplines to manage complexity, decrease time-to-market, and contain cost. The feasibility of component-based

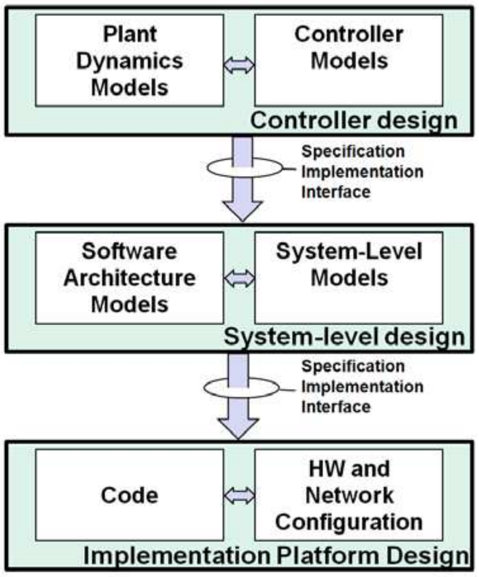
Plant Dynamics Models
Controller Models
Controller design
Specification Implementation Interface
Software Architecture Models
System-Level Models
System-level design
Specification Implementation Interface
Code
HW and Network Configuration
Implementation Platform Design

Fig. 1. Simplified CPS design flow.

system design depends on two key conditions: compositionality - meaning that system-level properties can be computed from local properties of components - and composability - meaning that component properties are not changing as a result of interactions with other components. Lack of compositionality and composability lead to behavioral properties that can be verified or measured only by system-level analysis (and/or testing), which becomes inefficient for real-world complex systems.

CPSs are inherently heterogeneous not only in terms of their components but also in terms of essential design requirements. Besides functional properties, CPSs are subject to a wide range of physical requirements, such as dynamics, power, physical size, and fault tolerance in addition to system-level requirements, such as safety and security. This heterogeneity does not go well with current methods of compositional design for several reasons. The most important principle used in achieving multi-objective compositionality is separation of concerns (in other words, defining design viewpoints). Separation of concerns works if the design views are orthogonal, i.e. design decisions in one view does not influence design decisions in other views. Unfortunately, achieving compositionality for multiple physical and functional properties simultaneously is a very hard problem because of the lack of orthogonality among the design views.

Figure 1 represents a simplified model-based design flow of a CPS composed of a physical plant and a networked control system. In a conventional design flow, the controller dynamics is synthesized with the purpose of optimizing performance. The selected design platform (abstractions and tools used for control design in the design flow) is frequently provided by a modeling language and a simulation tool, such as Matlab/Simulink. The controller specification is passed to the implementation design layer through a “Specification/Implementation Interface”. The implementation in itself has a rich design flow that we compressed here only in two layers: System-level design and Implementation platform design. The software architecture and its mapping on the (distributed) implementation platform are generated in the system-level design layer. The results - expressed again in the form of architecture and system models - are passed on through the next Specification and Implementation Interface to generate code as well as the hardware and network design. This simplified flow reflects the fundamental strategy in platform-based design [8]. Design progresses along precisely defined abstraction layers. The design flow usually includes top-down and bottom-up elements and iterations (not shown in the figure).

Effectiveness of the platform-based design largely depends on how much the design concerns (captured in the abstraction layers) are orthogonal, i.e., how much the design decisions in the different layers are independent. Heterogeneity causes major difficulties in this regard. The controller dynamics is typically designed without considering implementation side effects (e.g. numeric accuracy of computational components, timing accuracy caused by shared resource and schedulers, time varying delays caused by network effects, etc.). Timing characteristics of the implementation emerge at the confluence of design decisions in software componentization, system architecture, coding, and HW/network design choices. Compositionality in one layer depends on a web of assumptions to be satisfied by other layers. For example, compositionality on the controller design layer depends on assumptions that the effects of quantization and finite word-length can be neglected and the discrete-time model is accurate. Since these assumptions are not satisfied by the implementation layer, the overall design needs to be verified after implementation - even worst - changes in any layer may require re-verification of the full system.

An increasingly accepted way to address these problems is to enrich abstractions in each layer with implementation concepts. An excellent example for this approach is TrueTime [9] that extends Matlab/Simulink with implementation related modeling concepts (networks, clocks, schedulers) and supports simulation of networked and embedded control systems with the modeled implementation effects. While this is a major step in improving designers’ understanding of implementation effects, it does not help in decoupling design layers and improving orthogonality across the design concerns. A controller designer can now factor in implementation effects (e.g., network delays), but still, if the implementation changes, the controller may need to be redesigned.

Decoupling the design layers is a very hard problem and typically introduces significant restrictions and/or overdesign. For example, the Timed Triggered Architecture (TTA) orthogonalizes timing, fault tolerance, and functionality [10], but it comes on the cost of strict synchrony, and static structure. In an analogous manner, we propose to encompass passivity into traditional model-driven development processes in order to decouple the design layers and account for the effect of network uncertainties.

Our approach advocates a concrete and important transformation of model-based methods that can improve orthogonality across the design layers and facilitate compositional component-based design of CPSs. By imposing passivity constraints on the component dynamics, the design becomes insensitive to network effects, thus establishing orthogonality (with respect to network effects) across the controller design and implementation design layers. This separation of concerns empowers the model-based design process to be applied for networked control systems. Network effects need not be considered at the contoller design layer and the theoretical guarantees about stability and performance are independent of the implementation uncertainties. Further, stability is maintained even in the presence of disturbance traffic in the network.

The remaining of the paper presents in detail a simplified system consisting of a robotic arm controlled by a digital controller over a wireless network. We focus on the controller design layer and we manually generate Matlab/Simulink/TrueTime models of the overall system and analyze the behavior using simulations. Our results support the advantages of the passive control architecture regarding the orthogonality between the control design and the network uncertainties. Implementing an end-to-end tool chain for passive control design of systems consisting of multiple plants and multiple controllers communicating over wireless networks is a subject of current and future work.

III. Background on Passivity

There are various precise mathematical definitions for passive systems [7]. Essentially all the definitions state that the output energy must be bounded so that the system does not produce more energy than was initially stored. Strictly-output passive systems and strictly input passive systems with finite gain have a special property in that they are l_2-stable. Also, passive systems are Lyapunov-stable in terms of all observable states. Passive systems have a unique property that when connected in either a parallel or negative feedback manner the overall system remains passive. By simply closing the loop with any positive definite matrix, any discrete time passive plant can be rendered strictly output passive. This is an important result because it makes it possible to directly design low-sensitivity strictly-output passive controllers using the wave digital filters described in [4].

When delays are introduced in negative feedback configurations, the network is no longer passive. One way to recover passivity is to interconnect the two systems with wave variables. Wave variables were introduced by Fettweis in order to circumvent the problem of delay-free loops and guarantee that the implementation of wave digital filters is realizable [4]. Wave variables define a bilinear transformation under which a stable minimum phase continuous system is mapped to a stable minimum phase discrete-time system, and thus, the transformation preserves passivity.

Networks consisting of a passive plant and a controller are typically interconnected using power variables. Power variables are generally denoted with an effort and flow pair whose product is power. However, when these power variables are subject to communication delays, the communication channel ceases to be passive which can lead to instabilities. Wave variables allow effort and flow variables to be transmitted over a network while remaining passive when subject to arbitrary fixed time delays and data dropouts. If additional information is transmitted along with the continuous wave variables, the communication channel will also remain passive in the presence of time-varying delays [5]. More recently it has been shown that discrete wave variables can remain passive in spite of certain classes of time-varying delays and dropouts [6], [11]. In addition, a method which states how to properly handle time-varying discrete wave variables and maintain passivity has been developed in [7] and is used in our passive control architecture.

IV. Passive Control Architecture

This section presents a passive control architecture for a simplified system consisting of a robotic manipulator controlled by a digital controller over a wireless network. The architecture accounts for time-varying delays in the network. Specifically, all components are designed to preserve passivity ensuring stability of the closed loop system. We present an overview of the architecture focusing on the technical details required for the implementation. The theoretical foundations for control of passive plants over wireless networks can be found in [12].

A. Robotic System

Our control strategy takes advantage of the *passive* structure of a robotic system [13]. The robot dynamics which are denoted by $G_{robot}(\tau)$ in Figure 2 are described by

$$\tau = M(\Theta)\ddot{\Theta} + C(\Theta, \dot{\Theta})\dot{\Theta} + g(\Theta). \tag{1}$$

The state variables Θ denote the robot joing angles, τ is the input torque vector, $M(\Theta)$ is the mass matrix, $C(\Theta, \dot{\Theta})$ is the matrix of centrifugal and coriolis effects, and $g(\Theta)$ is the gravity vector.

Despite the complexity of robotic manipulators, simple control laws can be used in a number of cases. A fundamental consequence of the passivity property is that a simple independent joint continuous-time proportional-derivative (PD) control can achieve global asymptotic stability for set-point tracking in the absence of gravity [14]. Therefore, we employ a PD controller but we consider a discrete-time equivalent implementation that communicates with the robotic system via a wireless network. To compensate gravity, we select as the control command $\tau_u = \tau - g(\Theta)$. Then it can be shown that the robot is a *passive dissipative* system which is also *lossless* in which all supplied energy is stored as kinetic energy in the robot [15].

Furthermore, the robot can be made to be *strictly-output passive* by adding negative velocity feedback [7]. Therefore,

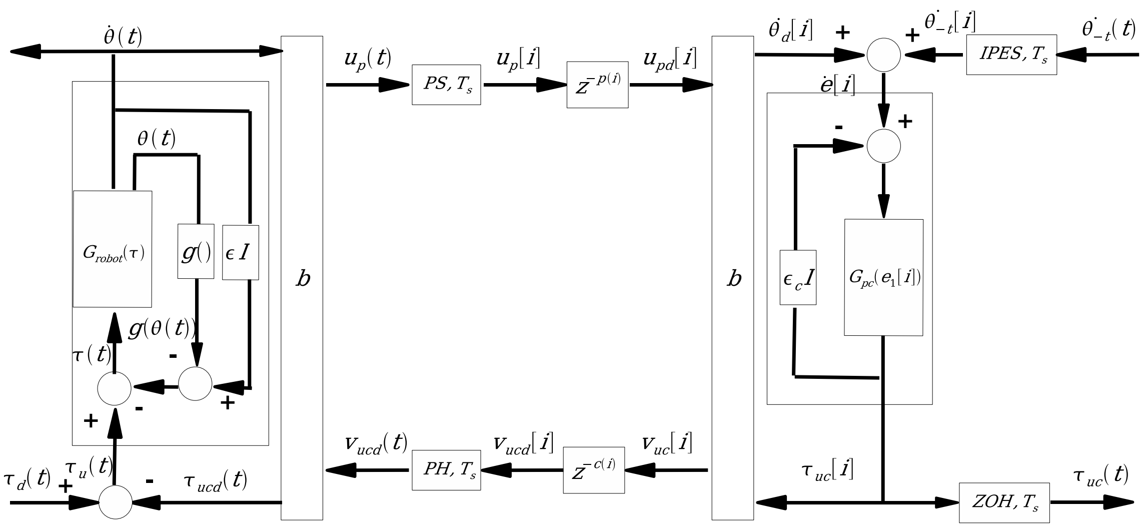

Fig. 2. Proposed Wireless Passive Control Architecture

we select the control command τ_u to have the following final form

$$\tau_u = \tau - g(\Theta) + \epsilon\dot{\Theta},\ \epsilon \geq 0.$$

The gravity compensation and the velocity damping are implemented locally at the robotic system and it can be shown that the gravity compensated system with velocity damping denoted $G : \tau_u \mapsto \dot{\Theta}$ is *passive* when $\epsilon = 0$ and *strictly-output passive* for any $\epsilon > 0$ respectively.

B. Wireless Control Architecture

Figure 2 depicts the proposed wireless control architecture. The robotic system $G : \tau_u \mapsto \dot{\Theta}$ is controlled by a a *passive* digital controller $G_{pc} : \dot{e}[i] \mapsto \tau_{uc}[i]$ using wave variables defined by the bilinear transformation denoted as b in Figure 2. The communication of the wave variables is subject to time-varying delays incurred in the wireless network that must be accounted for in order to ensure passivity and stability of the overall closed loop system.

The digital controller G_{pc} is interconnected to the robot via a *passive* sampler (PS) at sample rate T_s which converts the continuous *wave variable* $u_p(t)$ to an appropriate scaled discrete *wave variable* $u_p[i]$. Conversely, a *passive* hold device (PH) converts the discrete time *wave variable* $v_{ucd}[i]$ to an appropriately scaled *wave variable* $v_{ucd}(t)$ which is held for T_s seconds.

The *inner-product equivelant sampler* (*IPES*) and zero-order-hold (*ZOH*) blocks at the input of the digital controller are used to ensure that the overall system $G_{net} : [\dot{\Theta}^{\mathsf{T}}_{-t}(t), \tau_d^{\mathsf{T}}(t)]^{\mathsf{T}} \mapsto [\tau_{uc}^{\mathsf{T}}(t), \dot{\Theta}^{\mathsf{T}}(t)]^{\mathsf{T}}$ is (*strictly output*) *passive*. $\dot{\Theta}_{-t}(t)$ denotes a (negative) desired velocity profile for the robot to follow, $\tau_{uc}(t)$ is the continuous time *passive* control command, and $\tau_d(t)$ is a corresponding "disturbance" torque applied to the robots joints.

C. Wave Variables

The continuous robot input and output *wave variables* $v_{ucd}(t),\ u_p(t)\ \in \mathbb{R}^m$ depicted in Figure 2 are related to the corresponding torque and velocity vectors $\tau_{ucd}(t),\ \dot{\Theta}(t)\ \in \mathbb{R}^m$ as follows:

$$\frac{1}{2}(u_p^{\mathsf{T}}(t)u_p(t) - v_{ucd}^{\mathsf{T}}(t)v_{ucd}(t)) = \dot{\Theta}^{\mathsf{T}}(t)\tau_{ucd}(t).$$

The *wave variable* $v_{ucd}(t)$ and velocity measurement $\dot{\Theta}(t)$ are considered inputs and the *wave variable* $u_p(t)$ and delayed control torque $\tau_{ucd}(t)$ are considered outputs and are computed as follows:

$$\begin{bmatrix} u_p(t) \\ \tau_{ucd}(t) \end{bmatrix} = \begin{bmatrix} -I & \sqrt{2b}I \\ -\sqrt{2b}I & bI \end{bmatrix} \begin{bmatrix} v_{ucd}(t) \\ \dot{\Theta}(t) \end{bmatrix}$$

where $I \in \mathbb{R}^{m\times m}$ denotes the identitiy matrix.

The digital control input and output *wave variables* $u_{pd}[i],\ v_{uc}[i]\ \in \mathbb{R}^m$ depicted in Figure 2 are related to the corresponding discrete torque and velocity vectors $\tau_{uc}[i],\ \dot{\Theta}_d[i]\ \in \mathbb{R}^m$ as follows:

$$\frac{1}{2}(u_{pd}^{\mathsf{T}}[i]u_{pd}[i] - v_{uc}^{\mathsf{T}}[i]v_{uc}[i]) = \tau_{uc}[i]^{\mathsf{T}}\dot{\Theta}_d[i]$$

The *wave variable* $u_{pd}[i]$ and control torque $\tau_{uc}[i]$ are considered inputs and the *wave variable* $v_{uc}[i]$ and delayed velocity $\dot{\Theta}_d[i]$ are considered outputs and are computed as follows:

$$\begin{bmatrix} v_{uc}[i] \\ \dot{\Theta}_d[i] \end{bmatrix} = \begin{bmatrix} I & -\sqrt{\frac{2}{b}}I \\ \sqrt{\frac{2}{b}}I & -\frac{1}{b}I \end{bmatrix} \begin{bmatrix} u_{pd}[i] \\ \tau_{uc}[i] \end{bmatrix}$$

D. Passive Sampler and Passive Hold

The *passive* sampler denoted (PS,T_s) in Figure 2 and the corresponding *passive* hold denoted (PH,T_s) must be designed such that the following inequality is satisfied $\forall N > 0$:

$$\int_0^{NT_s} (u_p^{\mathsf{T}}(t)u_p(t) - v_{ucd}^{\mathsf{T}}(t)v_{ucd}(t))dt - \sum_{i=0}^{N-1}(u_p^{\mathsf{T}}[i]u_p[i] - v_{ucd}^{\mathsf{T}}[i]v_{ucd}[i]) \geq \ 0. \qquad (2)$$

This condition ensures that no energy is generated by the sample and hold devices, and thus, passivity is preserved.

Denote each j^{th} element of the column vectors $u_p(t), u_p[i]$ as $u_{p_j}(t), u_{p_j}[i]$ in which $j = \{1, \dots, m\}$. An implementation of the PS that satisfies condition (2) is given by

$$u_{p_j}[i] = \sqrt{\int_{(i-1)T_s}^{iT_s} u_{p_j}^2(t)dt} \; \mathsf{sign}(\int_{(i-1)T_s}^{iT_s} u_{p_j}(t)dt).$$

in which $j = \{1, \dots, m\}$.

Denote each j^{th} element of the column vectors $v_{ucd}(t), v_{ucd}[i]$ as $v_{ucd_j}(t), v_{ucd_j}[i]$ in which $j = \{1, \dots, m\}$. An implementation of the PH that satisfies condition (2) is

$$v_{ucd_j}(t) = \frac{1}{\sqrt{T_s}} v_{ucd_j}[i-1], \; t \in [iT_s, (i+1)T_s].$$

E. Passive controller

Typically a *passive* continuous-time PD controller is implemented as

$$\begin{aligned} \dot{e}_1(t) &= (\dot{\Theta}_d(t) + \dot{\Theta}_{-t}) \\ \tau_{uc}(t) &= K_p e_1(t) + K_d(\dot{\Theta}_d(t) + \dot{\Theta}_{-t}). \end{aligned}$$

A state-space realization of the controller can be described by

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (3)$$
$$y(t) = Cx(t) + Du(t). \quad (4)$$

where $A = 0, \; B = I, \; C = K_p = K_p^{\mathsf{T}} > 0, \; D = K_d = K_d^{\mathsf{T}} > 0\}$ (all matrices are in $\mathbb{R}^{m \times m}$).

To obtain a digital controller, we implement the discrete-time equivalent *passive* controller $G_{pc} : \dot{e}_1[i] \mapsto \tau_{uc}[i]$ computed from the state-space realization (3,4) with sampling period T_s. The resulting controller is implemented as

$$\begin{aligned} x[k+1] &= \mathbf{\Phi_{sp}} x[k] + \mathbf{\Gamma_{sp}} u[k] \\ y[k] &= \mathbf{K_s C_{sp}} x[k] + \mathbf{K_s D_{sp}} u[k]. \end{aligned} \quad (5)$$

where $\mathbf{K_s} > 0$ is a diagonal scaling matrix and $u[k] = (\dot{\Theta}_d[k] + \dot{\Theta}_{-t}[k])$. Details for computing the digital controller and a theoretical result showing that the controller is *strictly-output passive* can be found in [12, Section 2.3.1].

F. Passivity of the Closed-Loop System

All elements of the wireless control architecture are implemented to ensure passivity. In addition, if the communication protocol ensures that

$$\int_0^{NT_s} \dot{\Theta}^{\mathsf{T}}(t) \tau_{ucd}(t) dt \geq \sum_{i=0}^{(N-1)} \tau_{uc}^{\mathsf{T}}[i] \dot{\Theta}_d[i] \quad (6)$$

always holds, it can be shown that when $\epsilon_c = \epsilon = 0$ the system depicted in Figure 2 is *passive*. Furthermore, if $\epsilon_c > 0$, and $\epsilon > 0$ then the system is *strictly-output passive* and L_2^m stable. The proof is based on the fact that all components of the architecture preserve passivity and is a straightforward extension of the results presented in [12]. Condition (6) can be imposed on the wireless communication protocol by not processing duplicate transmissions of wave variables [7].

V. Evaluation

This section presents preliminary simulation results to evaluate the passive control architecture for controlling a robotic arm over a wireless network.

A. Experimental Setup

We consider the Pioneer 3 (P3) arm which is a robotic manipulator built for the P3-DX and P3-AT Activmedia mobile robots. The P3 Arm has two main segments, the manipulator and the gripper. The manipulator has five degrees of freedom and the gripper has an additional one. Figure 3 shows the home position of the P3 arm including the locations for the centers of gravity using the point mass assumption.

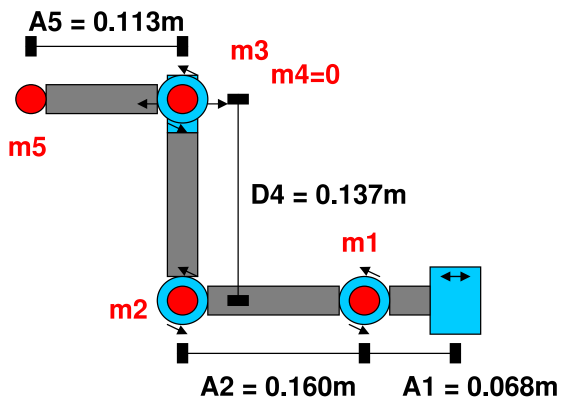

Fig. 3. Pioneer 3 Arm

The simulation model includes three main subsystems as shown in Figure 4. The dynamic model of the robotic arm is described by Equation (1) and is derived using the Langrangian approach for computing the elements of the mass matrix, coriolis and centrifugal vector, and gravity vector [16]. The model is implemented as a Simulink block using the “Robotics Toolbox for Matlab” [17] and includes gravity compensation and velocity damping as described in Section IV.

To evaluate the performance of the control scheme over a wireless network, we use the “Truetime Toolbox 1.5” [9]. We consider that the controller is inteconnected to the robotic arm via a 802.llb wireless network. The network subsystem contains three nodes implemented as TrueTime kernel blocks. The first node (node 1) implements the network interface of the digital controller and the second node (node 2) the interface for the P3 arm. A third network node (node 3) is used as a disturbance node in order to incur time-varying packet delay as described in [18]. For our simulations, we use the 802.11b wireless block in TrueTime with the throughput is set to 11 Mps, which is the theoretical limit of 802.11b, and the remaining parameters set to the default values. The controller wireless node and robot node are 10 meters apart while the disturbance node is 5 meters away from both. The packet size contains a 120 bit header plus preamble and a payload of 384 bits required to fit 6 double precision floating point values.

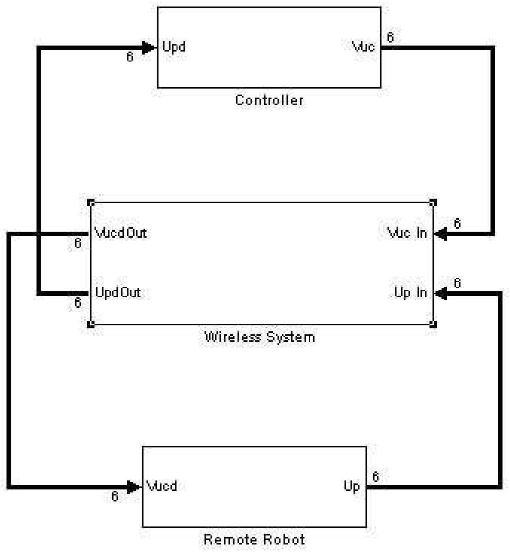

Fig. 4. Simulation model

The controller subsystem contains two components: a block from the robotics toolbox (jtraj) which provides the reference velocity trajectory for the robotic arm to follow and a discrete state-space model of the controller. The controller receives as input the reference trajectory along with the actual robot velocity and computes the torque control command for the robot. To demonstrate the advantages of the passive control architecture, we performed two sets of experiments, one using a non-passive control architecture and one using the passive control scheme presented in Section IV. In all the experiments, the reference provided to the controller commands the robot to go to a position of [1 0.8 0.6 0.4 0.2 0] from the start position of all joints equal to zero over a 4 sec time interval.

B. Non-passive Control Architecture

In the first set of experiments, we consider a non-passive control scheme. To implement the digital controller, we discretize the continuous-time PD controller described by Eqs. (3)-(4) using a standard zero-order hold operation [19]. The digital controller communicates with the robot directly without using wave variables. The gravity compensation and velocity damping are implemented locally as in the passive control scheme.

The non-passive system is highly unstable for large controller gains. To obtain reasonable results, we set the gains to $k_p = k_d = 3$. Figures 5 and 6 show the joint angles of the robotic arm for two different sampling periods. The system is unstable for $T_s = 0.1$ but becomes stable when $T_s = 0.05$ sec.

To simulate the system in the case of time-varying delays, we incorporate the disturbance node. The sampling period is kept constant (0.05 sec), but the amount of disturbance packets on the network varies. The disturbance node samples a uniform distributed random variable in $(0, 1)$ periodically every 0.01 sec. If the value is greater than a disturbance

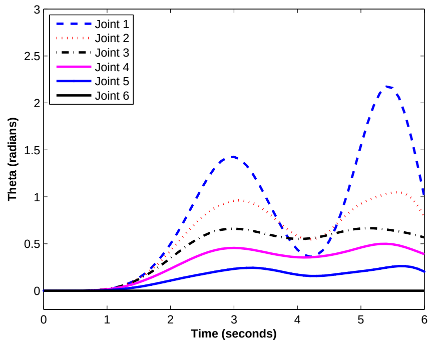

Fig. 5. Non-passive system - $T_s = 0.1$ sec.

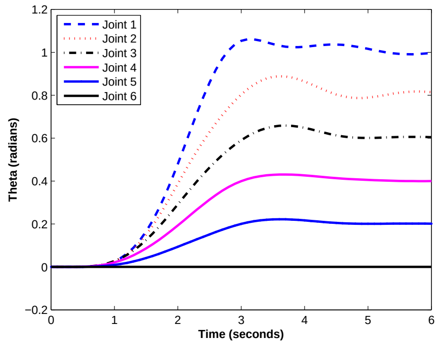

Fig. 6. Non-passive system - $T_s = 0.05$ sec.

parameter, a packet is sent out over the network. Figure 7 shows the network schedule when the disturbance parameter is 0.5. A value of 0 means the node is idle, a value of 0.25 means the node is waiting to send, and a value of 0.5 means the node is sending data. These values are superimposed to the node id and are plotted in Figure 7. Figure 8 is a graph of the network delay between the controller node and the robot node caused by the disturbance traffic. When the disturbance is increased, packet delay is introduced because the controller node and robot node have to wait for the disturbance node to stop sending as illustrated in the network schedule. Figure 9 shows that the time-varying delays have significant effect on the stability and performance of the robotic arm in the case of the non-passive control scheme.

C. Passive Control Architecture

The second set of experiments involves the proposed passive control architecture. In this case the control gains are set to $k_p = 600$ and $k_d = 2.8$. Figures 10, 11, and 12 show the results for sampling periods 0.2, 0.1, and 0.05 sec respectively. When the sampling period is increased, the robot slighly overshoots its destination but it finally settles to the desired location.

The passive control design not only allows larger controller gains and slower sampling but also ensures that the system is insensitive to time-varying delays. Figure 13 shows the network schedule and figure 14 the network delay when the

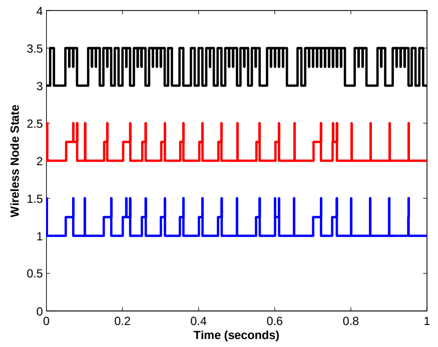

Fig. 7. Network schedule for disturbance = 0.5.

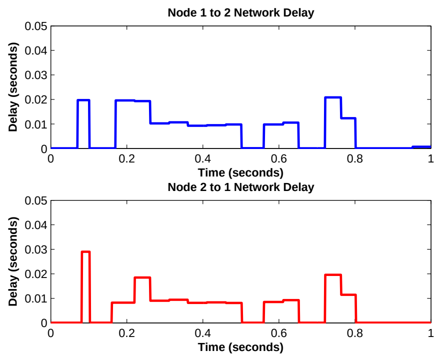

Fig. 8. Network delay for disturbance = 0.5.

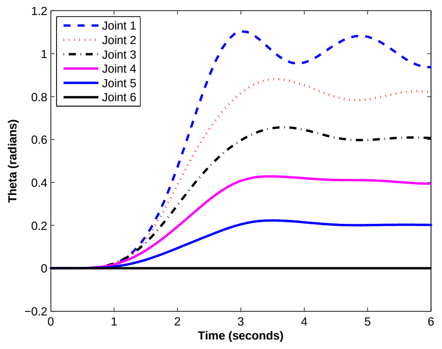

Fig. 9. Non-passive system - $T_s = 0.05$ sec.

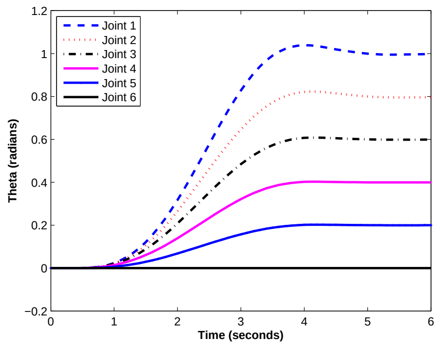

Fig. 10. Passive system - $T_s = 0.2$ sec.

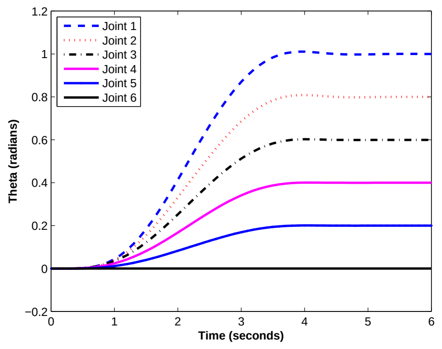

Fig. 11. Passive system - $T_s = 0.1$ sec.

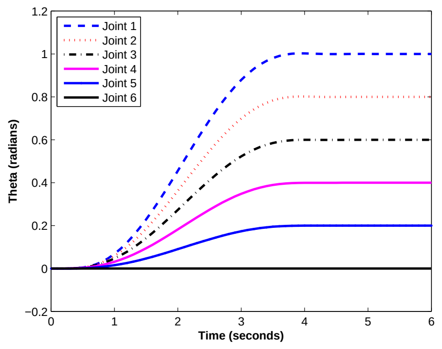

Fig. 12. Passive system - $T_s = 0.05$ sec.

disturbance parameter is 1 and the sampling period of the controller is $T_s = 0.05$. Figure 15 shows that the time-varying delays have little effect on the stability and performance of the robot as ensured by the theoretical analysis. The simulations are similar for the larger sampling periods as well. This robustness to time-varying delays stems from the passivity constraints imposed on all the components of the networked control architecture.

VI. Discussion and Future Work

The overall system dynamics of CPSs emerges from the interaction among physical dynamics, computational dynamics,

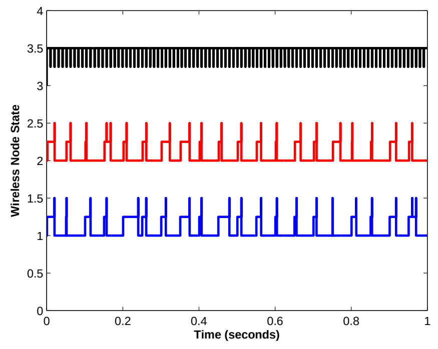

Fig. 13. Network schedule for disturbance = 1.

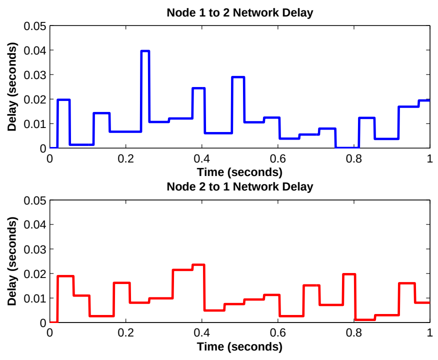

Fig. 14. Network delay for disturbance = 1.

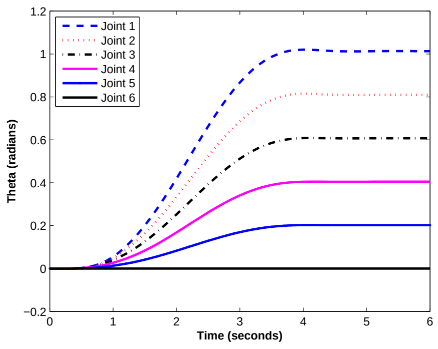

Fig. 15. Passive system - $T_s = 0.05$ sec.

and communication networks. This heterogeneity causes major challenges for compositional design of large-scale systems including fundamental problems caused by network uncertainties, such as time-varying delay, jitter, data rate limitations, packet loss and others. To address these implementation uncertainties, we propose a passive control architecture. The inherent safety of passive systems offers advantages in building CPSs that are insensitive to network uncertainties, thus improving orthogonality across the controller design and implementation design layers and empowering model-driven development. We have presented an architecture for a simplified system consisting of a robotic manipulator controlled by a digital controller over a wireless network and we have evaluated the system based on preliminary simulations results based on a detailed model that contains components for the robotic arm, the wireless network, and the digital controller that show that the system operation is insensitive to time-varying delays.

Our future work in this area aims at three major directions:

- We are investigating theoretical methods that provide an effective way to interconnect multiple passive systems and controllers together and preserve stability in the presence of time-varying delays and data dropouts as well as address important concerns such as distortion and low sampling rates;
- We are developing an integrated end-to-end tool chain for the model-based design of networked control systems that will support modeling, simulation, and model-based code generation of networked control applications; and
- We plan to demonstrate and experimentally evaluate the proposed design technology using a networked teleoperated robotic platform consisting of networked autonomous vehicles equipped with robotic arms and haptic paddles connected via a wireless network.

Acknowledgements

The first author would like to acknowledge the partial support of the National Science Foundation CAREER grant CNS-0347440.

References

[1] C. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*. Academic Press, Inc., 1975.

[2] P. Li and R. Horowitz, “Control of smart machines, part 1: Problem formulation and non-adaptive control,” *IEEE/ASME Trans. Mechatron.*, vol. 2, no. 4, pp. 237–247, 1997.

[3] D. Lee and P. Li, “Passive coordination control of nonlinear mechanical teleoperator,” *IEEE Trans. Rob. Autom.*, vol. 21, no. 5, pp. 935–951, 2005.

[4] A. Fettweis, “Wave digital filters: theory and practice,” *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270 – 327, 1986.

[5] G. Niemeyer and J.-J. E. Slotine, “Towards force-reflecting teleoperation over the internet,” in *IEEE International Conference on Robotics and Automation*, 1998, pp. 1909 – 1915.

[6] P. Berestesky, N. Chopra, and M. W. Spong, “Discrete time passivity in bilateral teleoperation over the internet,” in *IEEE International Conference on Robotics and Automation*, 2004, pp. 4557 – 4564.

[7] N. Kottenstette and P. J. Antsaklis, “Stable digital control networks for continuous passive plants subject to delays and data dropouts,” in *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007, pp. 4433 – 4440.

[8] A. L. Sangiovanni-Vincentelli, “Quo vadis sld: Reasoning about trends and challenges of system-level design,” *Proceedings of the IEEE*, vol. 95, no. 3, pp. 467 – 506, 2007.

[9] A. C. M. Ohlin, D. Henriksson, “TrueTime 1.5 - reference manual,” Department of Automatic Control, Lund University, Tech. Rep.

[10] H. Kopetz, “The time-triggered architecture,” *Proceedings of the IEEE*, vol. 91, no. 1, pp. 92 – 126, 2003.

[11] C. Secchi, S. Stramigioli, and C. Fantuzzi, “Digital passive geometric telemanipulation,” in *IEEE International Conference on Robotics and Automation*, 2003, pp. 3290 – 3295.

[12] N. Kottenstette, “Control of passive plants with memoryless nonlinearitites over wireless networks,” Ph.D. dissertation, University of Notre Dame, 2007.

[13] R. Ortega and M. Spong, “Adaptive motion control of rigid robots: A tutorial,” in *27th IEEE Conference on Decision and Control*, 1988, pp. 1575 – 84.

[14] W. S. Levine, *Control System Applications*. CRC Press, 2000.

[15] W. M. Haddad and V. S. Chellaboina, *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton University Press, 2008.

[16] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1989.

[17] P. I. Corke, “Robotic toolbox for matlab, release 7.1,” CSIRO, Tech. Rep.

[18] A. Cervin, D. Hendriksson, B. Lincoln, and K. Arzen, “Jitterbug and truetime analysis tools for real-time control systems,” in *2nd Workshop on Real-Time Tools*, 2002.

[19] P. Antsaklis and A. Michel, *Linear Systems*. McGraw-Hill, 1997.