Design of Networked Control Systems Using Passivity
Nicholas Kottenstette, Joseph Hall, Xenofon Koutsoukos, Janos Sztipanovits, and Panos Antsaklis

Abstract—Real-life cyber-physical systems, such as automotive vehicles, building automation systems, and groups of unmanned 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. This paper proposes a passive control architecture for designing networked control systems that are insensitive to network uncertainties. We describe the architecture for a system consisting of a robotic manipulator controlled by a digital controller over a wireless network and we show that the system is stable even in the presence of time-varying delays. Both simulation and experimental results demonstrate the advantages of the passivity-based architecture with respect to stability and performance and show that the system is insensitive to network uncertainties.

1 INTRODUCTION
1.1 Emerging Challenges
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, is inspired by the rapidly increasing use of Networked Control System (NCS) architectures in constructing real-world CPSs that integrate computational and physical devices using wireless networks such as medical device networks, groups of unmanned vehicles, and transportation networks. NCS research has been recently a very active area investigating problems at the intersection of control systems, networking, and computer science [1].

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 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 [2], [3]. 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 [4]. 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 [5] that extends MATLAB/Simulink with implementation related modeling concepts (networks, clocks, schedulers) and supports simulation of networked and embedded control systems. 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 over-design. For example, the Timed Triggered Architecture (TTA) orthogonalizes timing, fault tolerance, and functionality [6], 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.

Network delays are of primary concern in NCSs that require the development of novel analysis and synthesis methods. Other significant challenges include bandwidth limitations and packet dropouts. While there have been several methods for characterizing stability and performance (see [7] and the references there in), time-varying delays cause significant challenges. A new approach for decoupling between the control design and implementation layers has been proposed recently in [8]. The approach allows the design of state-feedback controllers that minimize a quadratic performance bound for a given level of timing jitter using linear matrix inequality methods.

1.2 Passivity-Based Design

This paper is motivated by the rapidly increasing use of network control system architectures in constructing real-world CPSs and aims at addressing fundamental problems caused by networks effects, such as time-varying delay, jitter, limited bandwidth, 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 [9]. 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 [10], teleoperators [11], digital filters [12], and networked control systems [13]–[15].

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. Detailed information about the network effects needs not to be considered at the controller 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 primary contributions of this paper are:

- We present a passive control architecture for a system consisting of a robotic manipulator controlled by a digital controller over a wireless network.
- We provide analytical results that prove that our architecture ensures stability of the networked control system
in the presence of time varying delays assuming that the communication protocols does not process duplicate transmissions.

- We implement the passive control architecture using MATLAB/Simulink/TrueTime models and we present simulation results for a typical 6 degree-of-freedom robotic arm controlled by a digital controller over a 802.11b wireless network that demonstrate that the passivity-based design offers significant advantages with respect to stability and performance. Specifically, the proposed solution (i) can use lower sampling rates that reduce the bandwidth requirements, (ii) allows higher gains that improve settling times and overshoot, and (iii) ensures robustness to time-varying network delays.

- We implement the passive control architecture on an experimental networked control system consisting of two computer nodes that realize the robotic manipulator and the digital controller respectively and communicate over an ad hoc 802.11b wireless network, and additional disturbance nodes that create traffic in the wireless network. We present experimental results that demonstrate the stable operation of the system in the presence of severe time-varying delays caused by network traffic generated by the disturbance nodes or by excessive computational load competing with the controller. Our experimental results validate that the passivity-based architecture ensures stability of the networked system and provides robustness to time varying delays.

The work presented in the paper demonstrates that passivity can be exploited to account for the effects of network uncertainties, thus improving orthogonality across the controller design and implementation design layers and empowering model-driven development. Part of this work has been presented in [16]. The main extensions are (1) experimental implementation and evaluation of the passivity-based architecture using a networked control system, (2) detailed design of the digital passive controller, and (2) theoretical analysis that includes the proofs of passivity and stability for the proposed architecture. It should be noted that passive structures offer additional advantages for robustness to finite length representations and saturation [12] but this paper focuses on network effects which is one of the most significant concerns in the development of CPSs.

The remaining of the paper after a brief background on passivity in Section 2, presents the passive control architecture in Section 3 focusing on the technical details required for implementation. Stability of the networked control system is proved in Section 4. Section 5 evaluates the passive control design using simulation results. The experimental evaluation is presented in Section 6. Finally, Section 7 presents the main directions of our future work.

2 Background on Passivity

There are various precise mathematical definitions for passive systems [15]. 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 that in that they are $L_2$-stable. 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 [12].

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 [12]. 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 [13]. More recently it has been shown that discrete wave variables can remain passive in spite of certain classes of time-varying delays and dropouts [14], [17]. In addition, a method which states how to properly handle time-varying discrete wave variables and maintain passivity has been developed in [15] and is used in our passive control architecture.

Before discussing our passive control scheme in Section 3 we recall the following definitions in regards to passivity and $L_2$-stability. These standard definitions which generalize “input-output” properties of many linear and nonlinear systems will be particularly useful when discussing the proof for Theorem 2 and understanding Corollary 1. In so doing, we choose to use the following compact notation.

$$\langle G(u), u \rangle_T \triangleq \int_0^T G(u(t))^T u(t) dt$$
$$\|\langle G(u) \rangle_T \|_2^2 \triangleq \int_0^T G(u(t))^T G(u(t)) dt$$

We also denote $L_2^m(U)$ as the extended $L_2^m$ space for the function $u(t) \in U$ in which $U \subset \mathbb{R}^m$ as all possible functions for a given $T \geq 0$ which satisfy:

$$\|u\|_T^2 < \infty.$$  

In the limit as $T \to \infty$, then $u \in L_2^m(U)$ is any function
which satisfies
\[ \int_0^\infty u^T(t)u(t)dt < \infty \] or more compactly, \( ||u||_2^2 < \infty \).

Note also that \( L^m_2(U) \subset L^n_2(U) \).

**Definition 1:** [18] Let \( G : L^m_2(U) \rightarrow L^n_2(U) \) then for all \( u \in L^m_2(U) \) and all real \( T \geq 0 \):

I. \( G \) is passive if there exist a constant \( \beta \) such that (1) holds.
\[ \langle G(u), u \rangle_T \geq -\beta \] (1)

II. \( G \) is strictly-output passive if there exists some constants \( \beta \) and \( \epsilon > 0 \) such that (2) holds.
\[ \langle G(u), u \rangle_T \geq \epsilon \| (G(u))_T \|_2^2 - \beta \] (2)

**Definition 2:** [18, Definition 1.2.3] Let \( G : L^m_2(U) \rightarrow L^n_2(U) \), it is said to be \( L^m_2 \)-stable if
\[ u \in L^m_2(U) \implies y = G(u) \in L^n_2(U), \]
and \( G \) is said to have finite-\( L^m_2 \)-gain if \( \exists \gamma, \beta \) s.t. for all \( T \geq 0 \)
\[ u \in L^m_2(U) \implies \| y \|_T \leq \gamma \| (u)_T \|_2 + \beta. \]

Any \( G : L^m_2(U) \rightarrow L^n_2(U) \) which has finite-\( L^m_2 \)-gain is \( L^m_2 \)-stable.

The following theorem will allow us to complete the proof of our main result (Theorem 2) in which it is shown that the network control system depicted in Fig. 2 is strictly-output passive for any passive robot (plant).

**Theorem 1:** [18, Theorem 2.2.14] Let \( G : L^m_2(U) \rightarrow L^n_2(U) \) be strictly-output passive. Then \( G \) has finite \( L^m_2 \)-gain.

The definitions chosen for passivity are chosen from the input-output perspective similar to the definition for positive systems given in [19]. Numerous linear and non-linear systems satisfy the above passivity definition such as positive real systems and dissipative passive systems [20]. When a dissipative dynamical system can be described by a Hamiltonian (the sum of kinetic and potential energy, \( H = T + V \)) a passive mapping typically exists in which the Hamiltonian serves as \( -\beta \) [20]. This is illustrated in our discussion of the passive structure of robotic systems. However, there are some limitations with the study of passive systems. For example, systems which consist of cascades of passive systems (such as two integrators in series) are not necessarily a passive system.

## 3 Passive Control Architecture

This section presents a passive control architecture for a 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 the architecture and technical details required for the implementation. The theoretical foundations for control of passive plants over wireless networks can be found in [21].

### 3.1 Robotic System

Our control strategy takes advantage of the passive structure of a robotic system [22]. The robot dynamics which are denoted by \( G_{\text{robot}}(\tau(t)) \) in Figure 2 are described by
\[ \tau = M(\Theta)\ddot{\Theta} + C(\Theta, \dot{\Theta})\dot{\Theta} + g(\Theta). \] (3)

The state variables \( \Theta \) denote the robot joint angles, \( \tau \) 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. The inertia matrix \( M(\Theta) = M(\Theta)^T > 0 \) and the matrix \( C \) and \( M \) are related as follows:
\[ -\Delta \tau = (M - 2C)u \implies x^T(M - 2C)x = 0, \forall x \in \mathbb{R}^n. \] (4)

It is the skew-symmetry property given by (4) which makes it possible for the robot to achieve a passive mapping. It is particularly useful when verifying that gravity compensation, for example maintains a passive mapping. 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 [23]. 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 the following supply rate
\[ s(\tau_u(t), \dot{\Theta}(t)) = \dot{\Theta}^T(t)\tau_u(t) \]
and corresponding storage function
\[ V(\dot{\Theta}(t)) = \frac{1}{2} \dot{\Theta}^T(t)M(\Theta(t))\dot{\Theta}(t) \]
can be used to show 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 [20]. Mathematically, this lossness property is characterized as follows:
\[ \int_0^T \dot{\Theta}(t)^T\tau_u(t)dt = V(x(T)) - V(x(0)) \]
\[ \int_0^T \dot{\Theta}(t)^T\tau_u(t)dt \geq -V(x(0)). \]

\( V(x(0)) \) represents all the available storage energy which can be extracted from the robot at time \( t = 0 \).

Furthermore, the robot can be made to be strictly-output passive by adding negative velocity feedback [15]. Therefore, we select the control command \( \tau_u \) to have the following final form
\[ \tau_u = \tau - g(\Theta) + \epsilon \dot{\Theta}, \epsilon > 0. \]

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. Therefore, the following conditions are satisfied:
3.2 Wireless Control Architecture

Figure 2 depicts the proposed wireless control architecture. The robotic system $G : \tau_u \rightarrow \dot{\Theta}$ is controlled by a passive digital controller $G_{pc} : \dot{\epsilon}_1[i] \rightarrow \tau_{ucd}[i]$ 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 equivalent sampler (IPES) and zero-order-hold (ZOH) blocks at the input of the digital controller are used to ensure that the overall system $G_{net} : \dot{\Theta}_n(t), \tau_{ucd}(t) \rightarrow [\tau_{uc}(t), \dot{\Theta}(t)]^T$ is (strictly output) passive. $\dot{\Theta}_n(t)$ 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.

3.3 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:

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

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:

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

where $I \in \mathbb{R}^{m \times m}$ denotes the identity 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:

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

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:

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

The received wave variables $u_{pd}[i]$, $v_{ucd}[i]$ are delayed versions of the transmitted wave variables $u_p[i]$, $v_{ucd}[i]$ such as...
where
\[ u_{pd}[i] = u_p[i - p(i)] \]
\[ v_{ucd}[i] = v_{uc}[i - c(i)]. \]

### 3.4 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 \(\forall N > 0:\)
\[
\int_0^{NT_s} (u_p^T(t)u_p(t) - v_{ucd}^T(t)v_{ucd}(t))dt - \sum_{i=0}^{N-1} (u_p[i]u_p[i] - v_{ucd}^T[i]v_{ucd}(i)) \geq 0. \tag{8}
\]
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 \(j = \{1, \ldots, m\}\). An implementation of the PS that satisfies condition (8) is given by
\[
u_{ucd}[j] = \sqrt{\frac{1}{s}} v_{ucd}[i-1], \quad t \in [iT_s, (i+1)T_s).
\]

We note that the PS effectively scales the feedback velocity from the robot as follows:
\[
\hat{\Theta}_d[i] \propto \sqrt{T_s} \hat{\Theta}((i-1)T_s - \tau(i - 1)T_s)).
\]
Furthermore, we note that the passive controller \(G_{pc}\) has infinite DC gain, and for a small \(\epsilon_c > 0\) at steady state:
\[
\hat{\Theta}[i] \approx -\hat{\Theta}_{-\tau}[i].
\]
Therefore, \(\hat{\Theta}_{-\tau}[i]\) can be related to a discrete time sampled robot velocity trajectory \(\hat{\Theta}_d[i] = \hat{\Theta}_d(iT_s)\) as follows:
\[
\hat{\Theta}_{-\tau}[i] = -\sqrt{T_s} \hat{\Theta}_d[i].
\]

### 3.5 Passive Controller

Typically a passive continuous-time PD controller is implemented as
\[
\dot{e}_1(t) = (\hat{\Theta}_d(t) + \hat{\Theta}_{-\tau}) \tag{11}
\]
\[
\tau_{uc}[t] = K_p e_1(t) + K_d(\hat{\Theta}_d(t) + \hat{\Theta}_{-\tau}).
\]
A state-space realization of the controller can be described by
\[
x(t) = Ax(t) + Bu(t) \tag{9}
\]
\[
y(t) = Cx(t) + Du(t) \tag{10}
\]
where \(A = 0, B = I, C = K_p = K_p^T > 0, D = K_d = K_d^T > 0\) (all matrices are in \(\mathbb{R}^{m \times m}\)).

To obtain a digital controller, first we design the discrete-time equivalent passive controller \(G_{pc} : \hat{e}_1[i] \mapsto \tau_{uc}[i]\) computed from the state-space realization (9)-(10) with sampling period \(T_s\). The resulting controller is
\[
x[i+1] = \Phi_o x[i] + \Gamma_o u[i]
\]
\[
y[i] = K_c p x[i] + K_d D_p u[i]. \tag{11}
\]
where \(u[i] = (\hat{\Theta}_d[i] + \hat{\Theta}_{-\tau}[i]).\) Note that \(K_s > 0\) is a real diagonal scaling matrix which is meant to match the steady state output for the discrete time implementation to the continuous time counterpart. However since both the continuous time and discrete time versions have an integrator, they each have infinite gain at steady state, therefore \(K_s = I\).

A discrete passive controller can be synthesized using the method presented in [21] and is described by
\[
\Phi_o = e^{A T_s}, \quad A_o = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ K_p & 0 \end{bmatrix}
\]
\[
\Gamma_o = \int_0^{T_s} e^{A \tau} d\tau B_o, \quad B_o = \begin{bmatrix} B \\ D \end{bmatrix} = \begin{bmatrix} I \\ K_d \end{bmatrix}
\]
\[
C_p = C_o (\Phi_o - I) = \begin{bmatrix} 0 & I \\ 0 & T_s K_p \end{bmatrix}
\]
\[
D_p = C_o \Gamma_o = \begin{bmatrix} 0 & I \\ 0 & T_s K_p \end{bmatrix}
\]
\[
T \in \mathbb{R}^{m \times m}, \quad \Gamma_o = T_s I \begin{bmatrix} \sigma^2_k P + T_s K_d \\ T_s K_p \end{bmatrix}
\]
\[
D_p = C_o \Gamma_o = \begin{bmatrix} 0 & I \\ 0 & T_s K_p \end{bmatrix}
\]
\[
H(z) = C_p (zI - \Phi_o)^{-1} \Gamma_o + D_p,
\]

It should be noted that this is not a minimal realization for this controller however, solving for
\[
G(z) = C_p (zI - \Phi_o)^{-1} \Gamma_o + D_p,
\]
it can be shown that the above passive controller \(\Sigma\) has the minimal form:
\[
\Phi_o = I \in \mathbb{R}^{m \times m}, \quad \Gamma_o = T_s I \in \mathbb{R}^{m \times m},
\]
\[
C_p = T_s K_p \in \mathbb{R}^{m \times m}, \quad D_p = \frac{T^2}{2} K_p + T_s K_d \in \mathbb{R}^{m \times m}.
\]

Next, we need to ensure that the mapping \(G_{sp} : \hat{e}_1[i] \mapsto \tau_{uc}[i]\) is strictly-output passive (see Fig. 2). Given \(\epsilon_c > 0\) we note that \(\epsilon_c I = K_x\) and define \(G = (I + \epsilon_c D_p) = (I + D_p K_x)\). Therefore the strictly-output passive controller has a discrete time realization \(\Sigma_sp = \{\Phi_sp, \Gamma_sp, C_sp, D_sp\}\) which is described analogously to (11) as
\[
\Phi_sp = \Phi_o - \epsilon_c \Gamma_o G^{-1} C_p = I - \epsilon_c \sigma^2_k G^{-1} K_p
\]
\[
\Gamma_sp = \Gamma_o (I - \epsilon_c G^{-1} D_p) = T_s (I - \epsilon_c G^{-1} D_p)
\]
\[
C_sp = G^{-1} C_p + T_s G^{-1} K_p
\]
\[
D_sp = G^{-1} D_p = G^{-1} D_p.
\]

Finally, to implement the digital controller we state the relationship between the control output \(\tau_{uc}[i]\) and wave output
datasets 
\[ v_{uc}[i] \] to the reference input \( \dot{\theta}_{-t}[i] \) and wave sensor input \( u_{pd}[i] \) (see Fig. 2).

The strictly-output passive digital controller can be synthesized from a continuous passive PD controller governed by (10) in which \( A = 0 \), \( B = I \), \( C = K_p + K_d > 0 \), \( D = K_d = K_d^T > 0 \) (all matrices are in \( \mathbb{R}^{m \times m} \)). The strictly-output passive digital controller with inputs \( (u_{pd}[i], \dot{\theta}_{-t}[i]) \) and outputs \( (\tau_{uc}[i], v_{uc}[i]) \) is implemented by

\[
\begin{align*}
    x[i + 1] &= \Phi_fe x[i] + \Gamma_fe (\sqrt{2} b u_{pd}[i] + \dot{\theta}_{-t}[i]) \\
    \tau_{uc}[i] &= C_fe x[i] + D_fe (\sqrt{2} b u_{pd}[i] + \dot{\theta}_{-t}[i]) \\
    v_{uc}[i] &= u_{pd}[i] - \frac{2}{b} \tau_{uc}[i]
\end{align*}
\]

in which

\[
\begin{align*}
    G_1 &= I + \frac{1}{b} D_{sp} \\
    C_{fe} &= G_1^{-1} C_{sp} \\
    D_{fe} &= G_1^{-1} D_{sp} \\
    \Phi_{fe} &= \Phi_{sp} - \frac{1}{b} \tau_{sp} C_{fe} \\
    \Gamma_{fe} &= \Gamma_{sp} (I - \frac{1}{b} D_{fe}).
\end{align*}
\]

4 Stability of the NCS

This section presents the main analytical result that proves the stability of the networked control system.

**Theorem 2:** For the wireless control architecture depicted in Fig. 2 consists of the passive robot described by (3) and (4) and the passive digital controller described by (11), if the communication protocol ensures that

\[
\int_0^{N_{T_s}} \dot{\Theta}_{uc}(t) \tau_{uc}(t) dt \geq \sum_{i=0}^{N-1} \tau_{uc}[i] \dot{\Theta}_{d}[i] \tag{12}
\]

always holds then when

\[
\epsilon_c = \epsilon = 0
\]

the system depicted in Figure 2 is passive. Furthermore, if

\[
\epsilon_c > 0, \text{ and } \epsilon > 0
\]

then the system is both strictly-output passive and \( L_2^m \) stable.

Condition (12) can be imposed on the wireless communication protocol by not processing duplicate transmissions of wave variables [15]. The proof is fairly intuitive in noting that if the controller or plant processes duplicated transmitted wave variables the system will generate energy which is a non passive operation. Communication protocols such as TCP are appropriate because they provide an un-duplicated ordered stream of data where as the User Datagram Protocol UDP protocol is not appropriate (without checking for duplicated transmissions) since checking of duplicated datagrams is not required. Note that condition (12) does not require that the data needs to be ordered or for all the data to arrive as is guaranteed by the TCP protocol, so choosing to use the UDP protocol may be a better choice for transmitting data as long as the control application is able to drop duplicated datagrams. Furthermore, the controller can essentially be run as an asynchronous task in which it only needs to compute and send a new control command as new data is received from the plant [24].

In order to discuss the proof we will use the following shorthand notation:

\[
\begin{align*}
    &\int_0^{N_{T_s}} f^r(t) g(t) dt \triangleq \langle f, g \rangle_{N_{T_s}} \text{ CT inner product} \\
    &\sum_{i=0}^{N-1} f^r[i] g[i] \triangleq \langle f, g \rangle_N \text{ DT inner product}
\end{align*}
\]

and wave sensor input \( \dot{\Theta}_{uc} \) and wave sensor input \( \dot{\Theta}_{d} \) respectively.

Note that in order to distinguish continuous time from discrete time the integral is taken to the limit \( N_{T_s} \) while the summation is taken to \( N - 1 \).

**Proof:** The PS and PH satisfy (8) which can be compactly written as

\[
\| (u_{pd})_{N_{T_s}} \|_2^2 - \| (v_{ucd})_{N_{T_s}} \|_2^2 \geq \| (u_{pd})_N \|_2^2 - \| (v_{ucd})_N \|_2^2. \tag{13}
\]

Integrating both sides of (7) and substituting into (13) results in

\[
\| (\dot{\Theta}, \tau_{ucd})_{N_{T_s}} \|_2^2 \geq \| (u_{pd})_N \|_2^2 - \| (v_{ucd})_N \|_2^2.
\]

By not processing duplicate wave variables transmissions we can enforce that

\[
\| (u_{pd})_N \|_2^2 - \| (v_{ucd})_N \|_2^2 \geq \| (u_{pd})_N \|_2^2 - \| (v_{ucd})_N \|_2^2 - \| (\dot{\Theta}_d, \tau_{uc})_N \|
\]

will always hold. Therefore, we are confident that we can satisfy (12) which can be more compactly written as

\[
\langle \dot{\Theta}_d, \tau_{ucd} \rangle_{N_{T_s}} \geq \langle \dot{\Theta}_d, \tau_{uc} \rangle_{N_{T_s}} \tag{14}
\]

The passive gravity compensated robot satisfies (6). Denoting \( V(x(0)) \) as \( \beta_r \) for the robot and \( \beta_r > 0 \) to account for non-zero initial conditions for the passive controller. Then the robot satisfies

\[
\langle \dot{\Theta}_d, \tau_{uc} \rangle_{N_{T_s}} = \epsilon \| (\dot{\Theta}_d)_{N_{T_s}} \|_2^2 - \beta_r, \tag{15}
\]

and the controller satisfies

\[
\langle \tau_{uc}, \dot{\epsilon} \rangle_N \geq \epsilon_c \| (\dot{\epsilon})_N \|_2^2 - \beta_c. \tag{16}
\]

We recall that

\[
\tau_{ucd}(t) = \tau_d(t) - \tau_u(t), \text{ and } \dot{\Theta}_d[i] = \dot{\epsilon}[i] - \dot{\Theta}_{-t}[i]. \tag{17}
\]

Substituting (17) into the left side of (14) and (18) into the right side of (14) results in

\[
\langle \dot{\Theta}_d, \tau_{d} \rangle_{N_{T_s}} + \langle \dot{\Theta}_{-t}, \tau_{ucd} \rangle_{N_{T_s}} \geq \langle \dot{\Theta}_{-t}, \tau_{uc} \rangle_{N_{T_s}}
\]

Substituting (15) and (16) into (19) results in

\[
\langle \dot{\Theta}_d, \tau_{d} \rangle_{N_{T_s}} + \langle \dot{\Theta}_{-t}, \tau_{uc} \rangle_{N_{T_s}} \geq \epsilon \| (\dot{\Theta}_d)_{N_{T_s}} \|_2^2 + \epsilon_c \| (\dot{\epsilon})_N \|_2^2 - (\beta_r + \beta_c). \tag{20}
\]
The passive sampler and hold blocks ensure that
\[ \sum_{i=0}^{N-1} \tau_{uc}[i]^T \dot{\Theta}_{-i}[i] = \int_0^{NT_s} \tau_{uc}(t)^T \dot{\Theta}_{-i}(t) \, dt \] and
\[ T_s \sum_{i=0}^{N-1} \tau_{uc}[i]^T \tau_{uc}[i] = \int_0^{NT_s} \tau_{uc}(t)^T \tau_{uc}(t) \, dt. \] (21, 22)

From (21) and (22), the sampling and hold operation satisfies
\[ \langle \tau_{uc}, \dot{\Theta}_{-i} \rangle_N = \langle \tau_{uc}, \dot{\Theta}_{-i} \rangle_{NT_s} \] and
\[ T_s \| (\tau_{uc})_N \|_2^2 = \| (\tau_{uc})_{NT_s} \|_2^2. \] (23, 24)

Substituting (23) and (24) into (20) results in
\[ \langle y, u \rangle_{NT_s} \geq \epsilon_s \| (y)_{NT_s} \|_2^2 - \beta_s. \] (25)

Therefore, (25) satisfies Definition 1-I for passivity when
\( (\epsilon_c, \epsilon) \geq 0 \) and either \( \epsilon_c = 0 \) or \( \epsilon = 0 \) \( \implies \) \( \epsilon_s = 0. \)

Furthermore (25) satisfies Definition 1-II when
\( \epsilon_c > 0 \) and \( \epsilon > 0 \) \( \implies \) \( \epsilon_s > 0 \)
in order for the system to be strictly-output passive. Furthermore, from Theorem 1 when
\( \epsilon_c > 0 \) and \( \epsilon > 0 \) \( \implies \) \( \epsilon_s > 0 \)
then the system is not only strictly-output passive but also \( L_2^m \)-stable.

**Corollary 1:** For the wireless control architecture depicted in Fig. 2 in which the robot \( G_{robot}(\tau(t)) \) is replaced by any passive system satisfying Definition 1-I (with gravity compensation disabled \( g(\Theta(t)) = 0 \) and the passive digital controller \( G_{pc}(\dot{\epsilon}_1[i]) \) satisfies Definition 1-I, if the communication protocol ensures that
\[ \int_0^{NT_s} \dot{\Theta}(t)^T \tau_{ucd}(t) \, dt \geq \sum_{i=0}^{N-1} \tau_{uc}[i]^T \dot{\Theta}_d[i] \] (26)
always holds then 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 both strictly-output passive and \( L_2^m \)-stable.

## 5 Evaluation Using Simulation

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

### 5.1 Simulation 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](image)

The simulation model includes three main subsystems as shown in Figure 4. The dynamic model of the robotic arm is described by Equation (3) and is derived using the Lagrangian approach for computing the elements of the mass matrix, Coriolis and centrifugal vector, and gravity vector [25]. The model is implemented as a Simulink block using the “Robotics Toolbox for Matlab” [26] and includes gravity compensation and velocity damping as described in Section 3.

![Fig. 4. Simulation model](image)

To evaluate the performance of the control scheme over a wireless network, we use the “TrueTime Toolbox 1.5” [5]. We consider that the controller is interconnected to the robotic arm via a 802.11b 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 [27]. For our simulations, we use the 802.11b wireless block in TrueTime with the throughput is set to 11 Mbps, 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.

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 3. 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.

5.2 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. (9)-(10) using a standard zero-order hold operation [28]. 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.

Since we are using a zero-order hold operation to convert the continuous controller to a digital controller and are applying a zero-order hold to the input of the robotic plant we can take working control gains for the passive framework and scale them using the following set of formulas:

\[ \alpha = 2T^2 \]

\[ k_p = \alpha k_{p\text{-passive}} \]

\[ k_d = \alpha k_{d\text{-passive}}. \]

In spite of our best efforts to scale the gains, the non-passive system requires \( \epsilon > .8 \) in order to add enough damping to stabilize the nominal system. As \( \epsilon \) is increased the system will begin to exhibit steady state error, so we chose to limit \( \epsilon = 1 \) for the non-passive system. Figure 5 compares the passive system (\( k_{p\text{-passive}} = 321, k_{d\text{-passive}} = 82, \epsilon = .5 \)) to the non-passive system (\( k_p = 1.6, k_d = .41, \epsilon = 1.0 \)). System responses are provided for both the nominal case and when subject to moderate time varying delays (disturbance \( = 0.5 \)). Due to the added phase lag from the uncompensated zero-order hold, the overall non-passive system has little flexibility in adjusting its gains. Therefore, only the non-passive response for \( T_s = 0.05 \) seconds could be evaluated.

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 parameter, a packet is sent out over the network. Figure 6 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 6. Figure 7 is a graph of the network delay between the controller node and the robot node caused by the disturbance traffic when the disturbance is 0.5. For comparison, Figure 8 depicts the corresponding network delay when the disturbance \( = 1 \) and sampling period is \( T_s = 0.05 \) seconds. 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 5 shows that the time-varying delays (disturbance \( = 0.5 \)) have significant effect on the stability and performance of the robotic arm in the case of the non-passive control scheme.

5.3 Passive Control Architecture

The second set of experiments involves the proposed passive control architecture. In order to choose an appropriate set of
continuous time gains \( k_p \) and \( k_d \) we focus our attention on joint 1 which is subject to the largest (changes of) inertia \( J \) as can be deduced from Figure 3.

\[
G_{pm}(s) = \frac{1}{Js}
\]  

Similarly we approximate the controller to be of the form

\[
G_c(s) = \frac{k_p + k_ds}{s}.
\]

Next using basic loop shaping techniques we desire the system to have a crossover frequency (\( \omega_c \) s.t. \( 20 \log_{10}(|G_{pm}(j\omega_c)G_c(j\omega_c)|) = 0 \) dB), in which \( \omega_c = \frac{\phi}{2\pi} \). \( \omega_n = \frac{\tau}{2\pi} \) is denoted as the Nyquist frequency. Therefore, the control gains can be computed based on a desired phase margin \( 0 < \phi \leq 90 \) (degrees) as follows:

\[
\tau = \frac{(\phi - 40)}{5\omega_c}
\]

\[
k_p = \frac{\omega_c^2}{J(\tau\omega_c + 1)}
\]

\[
k_d = k_p\tau.
\]

Although the phase margin will never exceed 90 degrees, you can still calculate appropriate gains for \( k_p \) and \( k_d \) for \( \phi > 90 \) using the above straight line approximation. Due to the highly non-linear nature of our system (with \( \epsilon = 1.0e-6 \)) we adjusted \( J \) to closely match the expected rise time given a 1 second trajectory since overshoot was still quite a significant component of the system response. All simulations given are for \( \phi = 80 \) degrees, \( N = 2 \), and \( J = 2.93 \) kg·m².

Next, we chose an appropriate trajectory time (\( \tau_t \)) which minimized overshoot and settling time. Finally, we evaluated the effectiveness of increasing \( \epsilon \) while maintaining tracking. Since, \( \epsilon \) and \( \epsilon_c \) serve primarily to show that the overall system is \( L_2 \)-stable we kept \( \epsilon_c = 1.0e-6 \) for all cases, the remaining system parameters are summarized in Table 1.

Figure 10 shows that as the sampling period is increased the overall system requires a larger trajectory time in order to minimize overshoot. Next, Figures (9, 12, 14) clearly show that by increasing \( \epsilon = 0.5 \) the passive system achieves faster settling times while exhibiting greater insensitivity to time varying delays when compared to Figures (11, 13) in which

<table>
<thead>
<tr>
<th>( T_s )</th>
<th>( \tau_t )</th>
<th>( \epsilon )</th>
<th>( k_p )</th>
<th>( k_d )</th>
<th>Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>.05</td>
<td>2.0</td>
<td>1.0e-6</td>
<td>321.0</td>
<td>81.7</td>
<td>11</td>
</tr>
<tr>
<td>.05</td>
<td>2.0</td>
<td>0.5</td>
<td>321.0</td>
<td>81.7</td>
<td>5, 9, 12, 10</td>
</tr>
<tr>
<td>.10</td>
<td>3.0</td>
<td>0.5</td>
<td>80.2</td>
<td>40.9</td>
<td>10</td>
</tr>
<tr>
<td>.20</td>
<td>4.0</td>
<td>1.0e-6</td>
<td>20.1</td>
<td>20.4</td>
<td>13</td>
</tr>
<tr>
<td>.20</td>
<td>4.0</td>
<td>0.5</td>
<td>20.1</td>
<td>20.4</td>
<td>14, 10</td>
</tr>
</tbody>
</table>
$\epsilon = 1.0e-6$. This robustness to time-varying delays stems from the passivity constraints imposed on all the components of the networked control architecture and the damping effects of $\epsilon$.

6 EXPERIMENTAL EVALUATION

This section presents experimental results for an NCS consisting of an asynchronous passive controller and a soft real-

time simulated passive plant representing the robotic arm using an actual 802.11b wireless network. The controller is implemented in an asynchronous manner so that the reference input $\dot{\theta}_{r,i}$ is buffered and processed as measurements from the plant $u_{pd,i}$ arrive over the wireless network. The controller and robotic plant are similar to the controller and plant in the simulation experiments which are implemented as Simulink models. However, the TrueTime Simulink blocks which model the wireless network are replaced with an actual ad-hoc wireless network. The simulated plant is executed using a variable step solver which prohibits simulating the plant in a hard-real-time manner. However, we are able to pace the Simulink simulation in a soft real-time manner such that the experienced network delays correspond to delays an actual plant would be subjected to.  

6.1 NCS Setup

The experimental setup is shown in figure 15. The network is a wireless 802.11b ad-hoc network with six wireless nodes.

1. We selected to simulate the plant because robotic arms such as the Pioneer 3 are controlled using simple servos and they don’t provide feedback.
One node contains the passive controller written in C, another node contains a Simulink program which simulates the robotic arm. The controller and the plant use the TCP/IP Send and TCP/IP Receive blocks in Simulink to communicate with the controller. The remaining four nodes are used to send disturbance packets onto the network.

In order to evaluate the stability and the robustness to time-varying network delays of the proposed architecture, we record the joint angles of the arm and the round-trip delays observed at the plant. The controller produces a trajectory for the robot to follow. The first stage moves the robotic arm from the zero home position to the position of \([1 \ 0.8 \ 0.6 \ 0.4 \ 0.2 \ 0] \text{ rad}\) within five seconds and with \(T_s = 0.1\) seconds. For the second stage, the robot remains in place for five seconds. In the third stage the robot returns to the home position within five seconds.

During a simulation, the controller waits for a connection from the computer containing the passive robotic model. During this time some or all of the disturbance machines send ping floods to the computer containing the passive controller. When the node containing the passive plant is able to send and receive data successfully, the plant model records the packet round-trip time. Specifically, the round-trip delay (plotted in figures such as Fig. 17) corresponds to the time difference when \(u_p[\text{sent}]\) is sent (\(t_{\text{sent}} = i_{\text{sent}}T_s\)) and when the corresponding control command arrives back to the plant in the form of a wave variable \(v_{\text{ucd}}[\text{arrived}]\) (\(t_{\text{arrived}} = i_{\text{arrived}}T_s\)). In other words \(\Delta_{\text{roundtrip}} = (t_{\text{arrived}} - t_{\text{sent}})T_s\).

**Experiment 1: Nominal Conditions**

In experiment 1, the controller and plant operate and communicate with each other without any communication from the disturbance nodes. This experiment shows how the system behaves under nominal conditions. Figure 16 displays the joint angles of the robotic arm that follow the reference trajectory provided to the controller. The round-trip network delay, as seen in figure 17 is minimal and repeatable, and it has very little effect on the stability of the robot model. The delay is a product of internal processing of both the plant and the controller rather than network delay itself.

**Experiment 2: Network disturbances**

Figures 18, 20, 22, and 24 show how the robotic model behaves in the face of network disturbance. During the experiment, each disturbance node outputs ping flood packets as fast as they come back or one hundred times per second, whichever is more. When one node or two nodes send out ping floods, the robot behavior is very close to the nominal case. However, when three and four disturbance nodes participate on the network, the controller computer has difficulty receiving messages from and sending messages to the plant computer. This is the case that demonstrates the advantages of the passive control architecture. When the plant is unable to communicate with the controller, the robot simply stops and waits for the next packet from the controller. This can be seen in figures 22 and 24. These results show than in the face of crippling network traffic, the robot remains stable.

**Experiment 3: CPU Disturbances**

In experiment 3, the disturbance nodes are silenced. In this experiment, the controller computer executes two programs simultaneously, the passive control program and a disturbance program. The disturbance program uses the Cygwin/Unix low-level copy program “dd” to continuously write random numbers to a file. This process takes overloads the CPU of the controller node. Both programs have the same priority, and both share the same single core processor. Figures 26 and
show how the robotic model behaves when the controlling computer is at 100 percent CPU load. The delay graph shows that the round trip delay is similar to the nominal case in experiment 1, and figure 26 also shows a similar performance to the system in experiment 1. Other programs on the controller computer produce little effect the system performance.

7 Conclusions and Future Work

The paper presents a passive control architecture that 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 system consisting of a robotic manipulator controlled by a digital controller over a wireless network and we have proven
the stability of the networked control system. Finally, we have evaluated the system using simulations and experimental results validating the significant advantages of the passivity-based architecture especially in the presence of time-varying delays. Our current and future work focuses on methods that provide an effective way to interconnect multiple passive systems and controllers as well as an integrated end-to-end tool chain for the model-based design of CPSs based on passivity.

Fig. 25. Packet Round-Trip Delay With Four Disturbance Nodes

![Packet Round-Trip Delay With Four Disturbance Nodes](image)

Fig. 26. Robot Performance With 100 Percent CPU Load

![Robot Performance With 100 Percent CPU Load](image)

Fig. 27. Packet Round-Trip Delay With 100 Percent CPU Load

![Packet Round-Trip Delay With 100 Percent CPU Load](image)

REFERENCES


