CONTROL OF PASSIVE PLANTS WITH MEMORYLESS NONLINEARITIES OVER WIRELESS NETWORKS

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Nicholas E. Kottenstette, B.S.M.E., M.S., M.S.E.E.

Panos J. Antsaklis, Director

Graduate Program in Electrical Engineering

Notre Dame, Indiana

August 2007

CONTROL OF PASSIVE PLANTS WITH MEMORYLESS NONLINEARITIES OVER WIRELESS NETWORKS

Abstract

by

Nicholas E. Kottenstette

This dissertation shows how to develop wireless networked embedded control systems (*wnecs*) in which the controller and the plant are isolated and can only interact over a wireless network. Many of the new results presented are based on *passivity* and *scattering* theory. In particular we show how to:

1. synthesize discrete time *passive*, *strictly-input passive*, and *strictly-output passive* systems from their continuous counterparts using a *inner-product equivalent sample and hold* (*IPESH*) block (with an optional *passive* observer),

2. create a data-drop out, and delay tolerant l^2-*stable* digital control network for a continuous *passive* plant in which:

   (a) the continuous *passive* plant can also be subject to various memoryless nonlinearities such as actuator saturation,

   (b) the digital controller only needs to be run when *passive* data is received over the wireless network,

   (c) the entire control network has been simulated on a theoretically validated wireless ring token network,

   (d) a new "power junction" is introduced in which multiple plants and controllers can interact while preserving *passivity*,

   (e) a new distortion measure is used to evaluate these control networks,

3. determine the capacity, and mean delays of a wireless ring token network.

We conclude with a presentation of *neclab*, a set of python and C based tools used to help an engineer simulate and develop *wnecs*.

To Maryellen, Marie, Therese, and Claire.

CONTENTS

FIGURES

TABLES

ACKNOWLEDGMENTS

I would like acknowledge my advisor, Dr. Panos Antsaklis, for all of his help and time. He has been there for guidance when I needed it. He has given me helpful advice, and to that, I am forever thankful. I would also like to thank the readers of this dissertation, Dr. Michael Sain, Dr. Peter Bauer, and Dr. Martin Haenggi. I appreciate your time and effort that you have put in, as well as the helpful suggestions and comments that have resulted in this process.

I would also like to acknowledge the donors of the Burns and Raymond Schubmehl Fellowships: thank you for your generosity.

To my parents Gene and Mary Kottenstette Jr.: thank you for all of your encouragement and support. To Joe and Lynn Finn Jr.: thank you for all of your encouragement and support. To my beautiful wife Maryellen: thank you for your love, encouragement and support, without you, this would not have been possible. To my wonderful daughters Marie, Therese, and Claire: you are all a father could ask for.

CHAPTER 1

INTRODUCTION

A wireless networked embedded control systems (*wnecs*) is a special class of *networked* control systems [2, 3], in which a collection of interconnected plant sensors, digital controllers, and plant actuators communicate with each other over wireless channels [11, 45, 54, 66, 68, 84, 108]. These wireless channels and interconnection of the network are typically maintained by a medium access control mechanism (*MAC*), and a specific routing protocol. The *MAC*, routing protocol, transceivers, and desired data rates, all affect the delay characteristics of the data being transmitted over a network. In existing *wnecs* literature, how a specific *MAC* affects control performance is typically not treated; also assumptions requiring some bounded limits on delay are typically made (with the exception of a few references [66, 68] in which the former investigates data dropout policies and the latter investigates specific *MAC*s in order to improve performance).

This dissertation takes a two pronged approach in developing *wnecs*. The first develops a control theory which allows us to tolerate long, unknown time varying delays and data dropouts. The second evaluates our control system with an accurate model for a wireless *MAC* and network routing protocol. The first approach is accomplished and discussed in detail in Chapter 2 and Chapter 3. Chapter 2 addresses the control of *passive* systems in which the effects of memoryless input nonlinearities such as actuator saturation are ignored. Chapter 3 addresses the control of *passive* systems subject to memoryless input nonlinearities and provides a viable solution to compensate for these memoryless input nonlinearities. Chapter 4 specifically studies the control systems developed in Chapter 2 which are implemented over a wireless token ring network in which the transceivers and data packets comply with the 802.15.4 standard. Chapter 5 concludes with a presentation of *neclab*, a set of tools used to help an engineer simulate and develop *wnecs*.

Chapter 2 presents a general framework to synthesize l^2*-stable* digital control networks for *passive* plants in which the control command and plant sensor data can be subject to delays and dropouts. Chapter 2 forms a basis for much of Chapter 3 and the latter half of Chapter 4. The *passive* plants and controllers are interconnected using *wave variables* which are created using a scattering transform (see Section 2.1.1 for additional details). A Linear Time Invariant (*LTI*) system is *passive* if and only if it is positive real (*PR*). Furthermore, a *LTI* system is *strictly-input passive* with *finite* l^2/L^2-*gain* if and only if it is *strictly positive real* (S*PR*), see Section A.1 and [25, 60, 133]. Finally a *strictly-input passive* (S*PR*) *LTI* system is also *strictly-output passive*, however the converse is not true. For example a *strictly-output passive LTI* system can have zeros on either the imaginary axis for continuous time systems or on the unit circle for discrete time systems. Many stability results can be determined by simply studying the input-output mappings of *passive* systems. However, Lyapunov [4, 73, 112] storage functions based on the internal states of a *passive* system do exist such that we can determine if a *LTI* system is (S)*PR* (Lemma 4). These storage functions can also be derived from the study of *Euler-Lagrange Systems* systems [94] and *Hamiltonian* systems [127].

Chapter 3 addresses how actuator constraints, such as saturation and other memoryless nonlinearities can adversely affect stability. Section 3.1 opens with a motivating example showing how a discrete mean square stable control system of a continuous first order plant in which the pole is strictly in the right half plane and subject to Bernoulli data drop outs will be destabilized when subjected to input actuator saturation. The l^2-*stable* networks presented in Chapter 2 can tolerate both time varying delays and data drop outs. However, we show that memoryless input nonlinearities such as actuator saturation can eliminate our desired *passivity* properties of a given plant. In Section 3.2.4, using an *inner-product recovery block* (*IPRB*) we prove that we can synthesize a l^2-*stable* digital control network which is subject to memoryless input nonlinearities.

Chapter 4 is concerned with a detailed study of wireless digital control of *passive* plants over token ring networks. In this chapter we derive network capacity (see Definition 9), and characterize the delays for our ring network when carrying correlated data between the controller and plant in our l^2-*stable* networks (Section 4.2). In Section 4.3 we study further the resulting *distortion* (Definition 10) between a desired position set point $\theta_{set}(i)$ and the resulting position output from the plant $\theta_{act}(i)$ and examine how a *passive* discrete time varying lossy data reduction *LDR* algorithm affects this distortion. In Section 4.3.2.1, we provide a proof showing that a *novel* asynchronous controller can create a l^2-*stable* network. In simulations this new controller obtained lower distortion levels than our controller which relied on adaptive data compression and required significantly less computations to implement.

In Chapter 5, a software environment is introduced called *neclab*, that is a software environment designed to allow easy deployment of networked embedded control systems, in particular wireless networked embedded control systems called *wnecs*. The components of *neclab* are presented in the following and described in terms of a classical control experiment, the ball and beam.

1.1 Motivating Examples of Wireless Networked Embedded Control Systems

As technology continues to lower costs and ease development of *wnecs*, applications which require *wnecs* are in development and deployment stages. Applications range from static wireless networks which monitor water levels and control gates in irrigation canals to mobile networks which attempt to provide real-time situation awareness to soldiers on the ground. These and other applications will be discussed in the following subsections.

1.1.1 Irrigation Control and Storm Water Drainage Control systems

In Australia, a decentralized control system has been implemented for the flow control of water in irrigation channels [18] which has shown impressive results in performance using a feed-forward compensation algorithm. Furthermore [18] shows promising simulations for a distributed control system which is solved using recently established convex programming techniques [24, 97]. Their simulations indicated *that their distributed controller performed nearly as well as a centralized H_∞controller*.

An emerging application of *wnecs* is related to reducing the number of combined sewer overflow (*CSO*) events by regulating the flow of storm water into combined sewer wastewater systems (*CSS*) in order to prevent mixing of sewage and storm water [92, 101]. When a large rainfall occurs the capacity of the *CSS* can be exceeded and sewage and rainwater are combined, resulting to the discharge of polluted storm water into nearby lakes and rivers which leads to environmental pollution and triggers large fines from the *EPA*. This is an extremely diverse and challenging problem in which wireless sensing of storm water holding basins, *CSS* water and sewage levels, and weather forecasting all can provide feed-back in order to make distributed control decisions to actuate valves which can regulate and divert flows in order to prevent *CSO* events.

1.1.2 Mobile Control Applications

Moving to applications involving mobile units, the following paper determines ways to achieve optimal sampling of moving currents in an ocean [61]. In [61] algoritms are given to determine optimal elliptical trajectories for a fleet of Slocum Gliders used to explore the ocean. These algorithms have to contend with very low data rate, asynchronous sampling, and large disturbances (due to the underwater currents) in order to coordinate their computationally and energy limited gliders. In spite of these limitations, their coordinated feedback control algorithm significantly improves sampling performance.

The military is particularly concerned with coordination and control of multiple unmanned aerial vehicles (*UAV*'s) [104]. These *UAV*'s are used for urban surveillance and reconnaissance in order to improve situation awareness (*SA*). Soldier have to contend with planning and implementing urban missions for a highly nonlinear and dynamic environment. Due to the close proximity of buildings, the enemy is provided with a three dimensional landscape to perform attacks. This three dimensional landscape results in a highly nonlinear environment to interact in. The structure of the city is constantly changing with new buildings being built, cars moving, etc. The combination of a nonlinear environment with the dynamically changing environment, allows us to classify the *SA* problem as a non-autonomous nonlinear control problem [73, 112]. Furthermore, this means the soldier will enter a highly uncertain environment. It is hoped that *UAV*'s will help reduce the uncertainty of the environment while also being robust enough to navigate in such an environment. An indirect way in which *UAV*'s may improve *SA* is by simply creating spatial diversity for ad-hoc communication networks which will be less suspect to multi-path fading effects in an urban environment [9]. In summary, it is hoped that the use of *UAV*'s for both surveillance and improved communication networks will improved *SA* for the soldier.

1.1.3 Heating, Lighting, and Power Control Applications

Over two-thirds of electricity generated in the US is for commercial buildings in which 40% is consumed by lighting. Research has shown that intelligent lighting systems (*ILS*) [26] can reduce electricity consumption by as much as 45% with little environmental impact due to the addition of wireless sensing and actuation components. These lighting systems use remote ambient light, temperature, and motion sensors to send information to a controller to vary the light levels in the room. The controller will vary the light levels due to the presence/absence of a user in the room and their own demands [88, 105].

Spatially distributed power electronic systems, which are used in telecommunication, naval, and micro grid power systems are attempting to meet increased demands for reliability, modularity and reconfigurability. A recent article was published to address these demands by showing wireless control of distributed voltage converters [78]. Two different types of voltage converters were tested, a two-module DC-DC buck converter and a two-module voltage inverter. In each unit a master-slave architecture was chosen in which the master controller would transmit an appropriate reference signal to the slave. Although the delays were minimal, they created significant problems for the two-module voltage inverter to compensate for.

1.1.4 Other Control Applications With Wireless Networks.

Many classic control applications attempt to stabilize unstable systems such as an inverted pendulum. In [13] a dynamic delay compensated controller to balance an inverted pendulum using Bluetooth for wireless feedback was implemented. It was not mentioned how long the controller could operate before failing. However, a similar experiment (*not using dynamic delay compensation*) could only successfully balance an inverted pendulum for about a minute using Bluetooth [120]. Another paper simulated and tested the flow of fluid in tanks [39]. Most other papers in this field present *simulated results* for idealized systems: [30, 110] simulate vehicle following problems, [29, 93] simulate the information consensus problem, while [63] simulate a binary controller for a first order plant which is subject to a minimum data rate transfer, [83] simulates model-based control of nonlinear plants such as an inverted pendulum and positioning of a satellite subject to fixed update rates, [66] performed simulation and tested a helicopter simulator, comparing Bernoulli dropouts and using a deterministic scheduling algorithm (it is interesting to note that a *nonlinear limit-cycle* in the experiment resulted in *unpredicted results* especially when the system was subjected to *Bernoulli dropouts*).

1.2 A Brief Review of Quantization Results in Networked Systems

Although we did not research the effects of quantization, the research in this field is significant [87]. We highlight a few additional papers which have addressed quantization issues. Many papers in this area focus on designing an optimal source encoder of the sensor data to be transmitted over a wireless channel and to be optimally decoded and sent to the controller. A recent paper looked at the problem slightly differently in which a remote set point would be sent to remote control system (in which the sensor and actuator were collocated) [106]. The approach of this paper looked at compressing the set point in order to transmit a more precise set-point with a limited channel data rate. The paper formulates and shows how to solve the following optimization problem (1.1)

$$\min_{K(M,N)} \sqrt{2}\|[\ (H_\alpha - T)\bar{r} \quad H_\alpha\eta\ ]\|_{\mathcal{H}_2} \text{ s.t. } H \text{ is stable.} \tag{1.1}$$

In which $\bar{r}$ is the largest p-normed refrence signal from the finite set of reference signals $\mathcal{C}_r$, and $\eta = \{\beta_{min} + (\beta_{max} - \beta_{min})\bar{\omega}\}$. β_{max} is the largest p-normed distance between any pair of signals $\in \mathcal{C}_r$ while β_{min} can be thought of as the largest p-normed distance of any pair of signals which are constrained to be within a given covering set of $\mathcal{C}_r$. The symbol $\bar{\omega}$ represents an upper bound on the probability of a decoding error. H is the closed loop transfer function (T is the desired closed loop transfer function) of a discrete single input single output plant with control gain K. H_α is the delayed system which is dependent on the number of source encoder symbols (M) and block channel encoder length bits N such that $H_\alpha = z^{-\alpha}H$ and $\alpha = \log_2(M) + N$. The results of the synthesis for a first order discrete time system show that compression is most useful for low noise channels, when the plant pole is more unstable and when the desired response time of T is long (the pole λ is near the unit circle).

Two papers have been presented relating optimal packet drop out policies $P_{drop}(k) =$

$G(\Upsilon[k])$ for linear discrete time systems which can be described by scalar difference equations [85, 86]. The latter paper is more general. The results are concerned with addressing observer stability and minimizing the steady state ($k \to \infty$) error variance of an observer $e[k] = E\{(x(k) - \hat{x}(k))^2\}_{\Upsilon[k-1],\ldots,\Upsilon[0]}$, in which $\hat{x}[k]$ is the state estimate of the state $x(k)$ from a Kalman filter. The results show that for point-to-point wireless communication over mobile channels that regardless of the instantaneous received Signal to Noise Ratio $\Upsilon[k]$, no data packets should be dropped in order to maximize the observer stability range. The articles also show that by providing a measure of the noise for the received packet, a form of *cross-layer feedback*, the average steady state error variance $e[\infty]$ will be minimized. Finally the article showed that for systems which did not utilize *cross-layer feedback* that an optimal dropping threshold Υ_T could be determined in order to minimize $e[\infty]$ at the cost of reduced stability. The dropping policy is if $\Upsilon[k] < \Upsilon_T$ the package should be dropped. The assumption for the noise process $\Upsilon[k]$ is stationary and independent from one transmission to the next (no other assumptions are made such as the distribution of Υ). Stability of the filter in general can be described in terms of the scalar state coefficient A and probability distribution function (pdf) of Υ (1.2).

$$E(P_{drop}) = \int_{0}^{\Upsilon_T} pdf(\Upsilon) d\Upsilon < A^{-2} \tag{1.2}$$

Another article which has examined encoder and decoder design for control of plants over networks is [123]. These additional texts provide results related to control and quantization [39, 63].

CHAPTER 2

DIGITAL CONTROL NETWORKS FOR CONTINUOUS PASSIVE PLANTS WHICH MAINTAIN STABILITY WHEN SUBJECT TO FIXED DELAYS, TIME VARYING DELAYS AND DATA DROPOUTS

This chapter provides procedures to synthesize l^2-*stable* networks in which the controller and plant can be subject to delays and data dropouts. This approach can be applied to control systems which use "soft-real-time" cooperative schedulers as well as those which use wired and wireless network feedback. The approach applies to plants and controllers which are *passive*, and allows for these *passive* systems to be either linear, nonlinear, and (or) time-varying. This framework arises from fundamental results related to *passive* control, and scattering theory which are used to design *passive* force-feedback telemanipulation systems, in which we provide a short review. Theorem 3 states how a (non)linear (*strictly input* or *strictly output*) passive plant can be transformed to a discrete (*strictly input*) passive plant using a particular digital sampling and hold scheme. Furthermore, Theorem 4(5) provide new sufficient conditions for l^2 (and L^2)-*stability* in which a *strictly-output passive* controller and plant are interconnected with only *wave-variables*. Lemma 2 shows it is sufficient to use discrete *wave-variables* when data is subject to fixed time delays and dropouts in order to maintain *passivity*. Lemma 3 shows how to safely handle time varying discrete *wave-variable* data in order to maintain *passivity*. Based on these new theories, we provide an extensive set of new results as they relate to *LTI* systems. For example, Proposition 2 shows how a *LTI* *strictly-output passive* observer can be implemented. We then present a new cooperative scheduler algorithm to implement an l^2-*stable* control network. We also provide an illustrative simulated example which uses a *passive* observer followed with a discussion for future research.

2.1 Introduction

The primary goal of this research is to develop reliable wireless control networks. These networks typically consist of distributed-wireless sensors, actuators and controllers which communicate with low cost devices such as the MICA2 and MICAz motes [54]. The operating systems for these devices, typically consist of a very simple scheduler, known as a cooperative scheduler [44]. The cooperative scheduler provides a common time-base to schedule tasks to be executed, however, it does not provide a context-switch mechanism to interrupt tasks. Thus, tasks have to cooperate in order not to delay pending tasks, but this cooperative condition is rarely satisfied. As a result, a controller needs to be designed to tolerate time-varying delays which can incur from disruptive tasks which share the cooperative scheduler. Although, other operating systems can be designed to provide a more hard real-time scheduling performance, the time varying delays which will ultimately be encountered with wireless sensing and actuation will be comparable if not more significant. Hence, the primary aim of this chapter is to provide the theoretical framework to build l^2-*stable* controllers which can be subject to time-varying scheduling delays. Such results are also of importance as they will eventually allow the plant-controller network depicted in Fig. 2.6 to run entirely isolated from the plant as is done with telemanipulation systems. Telemanipulation systems have had to address wireless control problems [1] years before the MICA2 mote existed and the corresponding literature provides results to address how to design stable control systems subject to transmission delays in such systems. Much of the theory presented in this chapter is inspired and related to work related to telemanipulation systems. Thus, our introduction will conclude (Section 2.1.1) with a brief review of telemanipulation, and how it relates to *passive* control and scattering theory in order too provide the reader some physical insight related to the framework presented in Section 2.2.

Telemanipulation systems are distributed control systems in which a human operator controls a local manipulator which commands a remotely located robot in order to modify a remote environment. The position tracking between the human operator and the robot is typically maintained by a passive proportional-derivative controller. In fact, a telemanipulation system typically consists of a series network of interconnected two-port passive systems in which the human operator and environment terminate at each end of the network [89]. These passive networks can remain stable in-spite of system uncertainty; however, delays as small as a few milliseconds would cause force feedback telemanipulation systems to become unstable. The instabilities occur because delayed power variables, force (effort) and velocity (flow), make the communication channel non *passive*. In [1] it was shown that by using a scattering transformation of the power variables into power *wave variables* [90] the communication channel would remain passive in spite of arbitrary fixed delays. For continuous systems, 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 [91]. However, only recently has it been shown how discrete *wave variables* can remain passive in spite of time varying delays and dropouts [11, 107]. We verified this to be true for fixed time delays and data dropouts (Lemma 2). However, we provide a simple counter example that shows this is not the case for all time-varying delays and provide a lemma which states how to properly handle time varying discrete wave variable data and maintain *passivity* (Lemma 3). The initial results from [107] build upon a novel digital sample and hold scheme which allows the discrete inner-product space and continuous inner-product space to be equivalent [103, 118].

We build on the results in [118] to show in general how to transform a (non)linear (*strictly input* or *strictly output*) *passive* system into a discrete (*strictly input*) *passive* system (Theorem 3). We then formally show some new l^2-*stability* results related to *strictly-output passive* networks. In particular Theorem 2 shows how to make a discrete *passive* plant *strictly-output passive* and l^2-*stable*. Theorem 2 also makes it possible to synthesize discrete *strictly-output passive* systems from discrete *passive LTI* systems such as those consisting of passive wave digital filters [32]. We will then use the scattering transform to interconnect the controller to the plant with *wave variables*. We use Lemma 3 to show that the cooperative scheduler can allow time varying data transmission delays and maintain passivity between the plant and controller. As a result our digital control system implemented with a cooperative scheduler will remain l^2-*stable*. We conclude this introduction with a brief discussion of telemanipulation systems, *passivity* and scattering theory from continuous time and classic control framework. Section 2.2 provides the necessary definitions and theorems necessary to present our main results. Section 2.3 shows our main results and outlines how to design a driver which allows the digital controller to be implemented as a cooperative task managed by a cooperative scheduler, such as the one provided by *SOS*. Section 2.4 concludes with a simulation implementing the cooperative scheduler to control a passive system. Section 2.6 summarizes our key findings and discusses future research directions.

2.1.1 Passive Systems and Telemanipulation.

Passive systems are an important class of systems for which Lyapunov like functions exist [25, 73, 112, 127]. The Lyapunov like function arises from the definition of passivity (2.1). In passive systems (2.1), the rate of change in stored energy E_{store} is equal to the amount of power put in to the system P_{in} minus the amount of power dissipated P_{diss} which is greater than or equal to zero.

$$\dot{E}_{store} = P_{in} - P_{diss} \tag{2.1}$$

As long as all internal states x of the system are associated with stored energy in the system, we can show that a passive system is stable when no input power is present simply by setting $P_{in} = 0$. $P_{diss} \geq 0$ implies that $\dot{E}_{store} \leq 0$ which shows the system is Lyapunov stable. By using either the invariant set theorem or Barbalat's Lemma [112] we can prove asymptotic stability [89]. These passive systems can be interconnected in parallel and feed-back configurations and are fundamental components in telemanipulation systems [89]. When a telemanipulation system incurs millisecond communication delays between the master controller and slave manipulator instabilities can occur. These delays primarily destabilize the system because the communication channel is no longer a *passive* element in the telemanipulation system [90]. *Wave variables* are used here to communicate commands and provide feed-back in telemanipulation systems, because they allow the communication channel to remain passive for arbitrarily fixed delays. The variables which traditionally in the past were communicated over a telemanipulation channel were *power variables* such as force and velocity (F,$\dot{x}$). *Power variables*, generally denoted with an *effort* and *flow* pair (e_*,f_*) whose product is power, are typically used to show the exchange of energy between two systems using *bond graphs* [17, 38]. Some other examples of *effort* and *flow* pairs of *power variables*

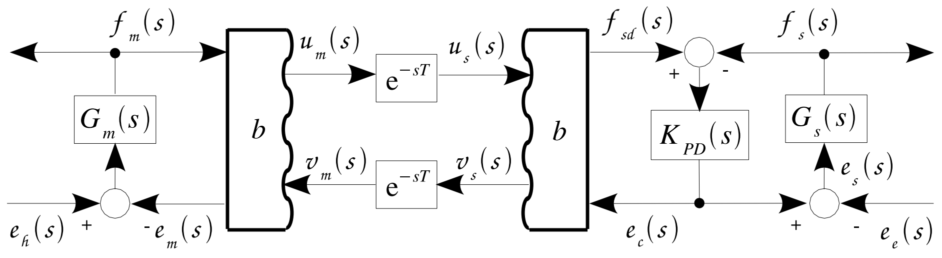

Figure 2.1. Telemanipulation system depicted in the s-Domain, subject to communication delays.

are voltage and current (V,$\dot{q}$), and magnetomotive force and flux rate ($\mathcal{F}$,$\dot{\varphi}$). *Wave variables* are denoted by the following pair of variables (u_*,v_*), the transmission wave impedance $b > 0$ and the channel communication time delay T [90]. The transmission between the master and slave controller (as depicted in Fig. 2.1 in the s-Domain) are governed by the following delayed equations:

$$u_s(t) = u_m(t - T) \tag{2.2}$$

$$v_m(t) = v_s(t - T) \tag{2.3}$$

in which the input waves are computed using

$$u_m(t) = \frac{1}{\sqrt{2b}}(bf_m(t) + e_m(t)) \tag{2.4}$$

$$v_s(t) = \frac{1}{\sqrt{2b}}(bf_{sd}(t) - e_c(t)) \tag{2.5}$$

These simple wave variable transformations, which can be applied to vectors, allow us to show that the wave communication channel is both passive and lossless assuming zero initial conditions.

$$E_{store}(t) = \int\limits_0^t P_{in} d\tau = \int_{t-T}^{t} \frac{1}{2} u_m^{\mathsf{T}} u_m + \frac{1}{2} v_s^{\mathsf{T}} v_s \geq 0 \tag{2.6}$$

In Fig. 2.1, the transfer function associated with the master manipulator is denoted by $G_m(s)$ and is typically a *passive* mass. Furthermore, the slave manipulator is denoted by the transfer function, $G_s(s)$ and is typically a *passive* mass. The *passive* "proportional-derivative" plant controller $K_{PD}(s)$ has the following form:

$$K_{PD}(s) = \frac{Bs + K}{s} \tag{2.7}$$

The plant controller is "proportional-derivative" in the sense that the integral of the flow variable f_* yields a displacement variable q_* which is then multiplied by a proportional gain K and derivative term B. Note that the final position of the plant is dependent on the systems initial condition. *Passivity* will be preserved in this configuration.

Figure **??** depicts a more fundamental system which we desire to study in which the controller $G_p(s)$ is interconnected to the plant $K_{PD}(s)$ with wave variables. If $e_d(s) = 0$ then it can be shown that $K(s)$ is positive real for $\forall b > 0$ (see [75] for explanation of $K(s)$) which implies the system is *passive* and stable [75, Section B]. We may be able to show that the system is L^2 stable when $e_d(s) = 0$ using Theorem 2 in [76]. However, we will show that it is sufficient for $K_{PD}(s)$ and $G_P(s)$ to be modified to be *strictly-output passive* in order to satisfy L^2 stability $\forall b > 0$ in which both $e_d(s)$ and $r_s(s)$ can be signals in L^2. The proof for both L^2 and l^2 stability is given in Section 2.3. Although the wave variables (u_*,v_*) do not need to be associated with a particular direction as do the power variables, when interconnected with a pair of effort and flow variables an effective direction is implied. Fig. 2.2 shows how to implement the wave transform for

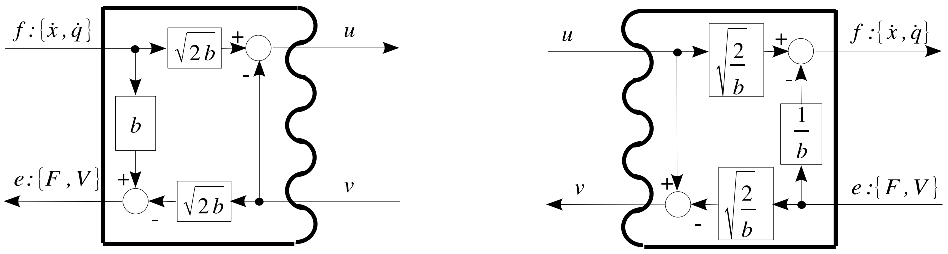

Figure 2.2. Block diagrams depicting the wave variable transformation (simplified version of Fig. 3 in [121]).

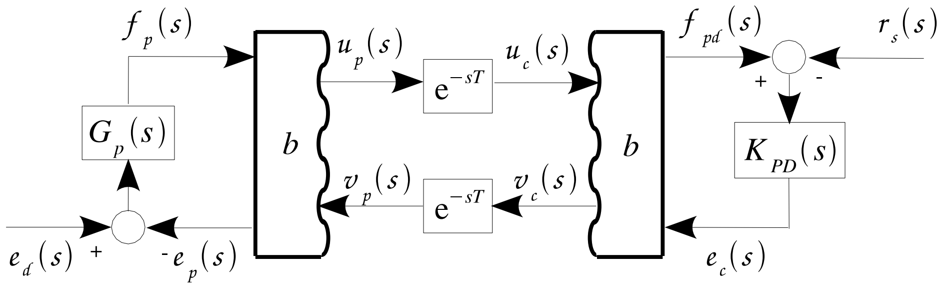

Figure 2.3. A fixed delay-insensitive system in which a passive controller commands a passive plant.

both cases. Fig. 2.1 can be modified to yield the following system in which a passive controller $K_{PD}(s)$ is able to command a passive plant $G_p(s)$. The plant will follow the negative flow set-point $\mathbf{r}_s(s)$. If we precede the flow set-point with a causal derivative filter $G_d(s) = \frac{s}{\tau s+1}$ then the plant will track a desired displacement set-point $\mathbf{q}_s(s)$.

The following observations, have been made by simulating this system: If the plant is a *passive* mass, then the plant displacement will equal the negative displacement set-point at steady state. If the plant is *passive* and stable such as a mass-spring-damper, then steady state error will occur. So far the discussion has taken place with respect to the continuous time domain in which it has been shown that delayed data to and from the controller $K_{PD}(s)$ can occur in an isolated manner such that a passive control system can be designed.

2.2 Passive Control Theory

Passive control theory is extremely general and broad in that it applies to a large class of controllers for linear, nonlinear, continuous and discrete control systems. In [25] control theory for continuous and discrete passive systems is presented. In particular, passive control theory has been used in digital *adaptive control* theory to show stability of various *parameter adaptation algorithms* [57]. Additional texts which discuss nonlinear continuous passive control theory are [73, 112, 127]. In [94] a comprehensive treatment is dedicated to the passive control of a class of nonlinear systems, known as *Euler-Lagrange Systems*. *Euler-Lagrange Systems* can be represented by a *Hamiltonian* which possess a Dirac structure that allows dissipative and energy storage elements to be interconnected to ports without causal specification [117, p. 124]. Thus, in [117] an extensive treatment on intrinsically passive control using Generalized Port-Controlled Hamiltonian Systems is presented, in particular as it relates to telemanipulation and scattering theory. Our presentation of passive control theory focuses on laying the groundwork for discrete passive control theorems, mirrors the continuous counterpart results presented in [127], and is based off of the continuous and discrete time framework presented in [25].

2.2.1 l^2 Stability Theory for Passive Networks

Definition 1 *The l^2 space is the real space of all bounded, infinitely summable functions $f(i) \in \mathbb{R}^n$. We assume $f(i) = 0$ for all $i < 0$ and note that $\mathbb{R}^n$ could be replaced with $\mathbb{C}^n$ in (2.8) without loss of generality. Denoting $\langle \cdot, \cdot \rangle$ as an inner product [7], the*

l^2 space is the set of all functions $f(i)$ which satisfy the following inequality (2.8).

$$\sum_{i=0}^{\infty}\langle f^*(i), f(i)\rangle < \infty \tag{2.8}$$

A truncation operator will be defined as follows:

$$f_N(i) = \begin{cases} f(i), & \textit{if}\; 0 \leq i < N \\ 0, & \textit{otherwise} \end{cases} \tag{2.9}$$

Likewise the extended l^2 space, l_e^2, is the set of all functions $f(i)$ which satisfy the following inequality (2.10).

$$\sum_{i=0}^{N-1}\langle f^*(i), f(i)\rangle < \infty, N \geq 1 \tag{2.10}$$

Note that $l^2 \subset l_e^2$. Typically l_e^2 is defined with the summation to N and the truncation operator includes N [57, p. 75] [25, p. 172], however, these definitions are equivalent. Finally we can define our l^2 norms (2.11) and truncation of the l^2 norm (2.12) as follows:

$$\|f(i)\|_2 \triangleq (\sum_{i=0}^{\infty}\langle f(i), f(i)\rangle)^{\frac{1}{2}} \tag{2.11}$$

$$\|f(i)_N\|_2^2 \triangleq \langle f(i), f(i)\rangle_N \triangleq \sum_{i=0}^{N-1}\langle f(i), f(i)\rangle \tag{2.12}$$

The following definition for l^2-stability is similar to the one given in [19] which refers to [127] in regards to stating that *finite l^2-gain* is sufficient for l^2-stability, however, in [127] this is only stated for the continuous time case. We provide a short proof for the discrete time case and we note for completeness where the development parallels [127].

Definition 2 *Let the set of all functions $u(i) \in \mathbb{R}^n$, $y(i) \in \mathbb{R}^p$ which are either in the l^2*

space, or l_e^2 space be denoted as $l^2(U)$/$l_e^2(U)$ and $l^2(Y)$/$l_e^2(Y)$ respectively. Then define G as an input-output mapping $G : l_e^2(U) \to l_e^2(Y)$, such that it is l^2-stable if

$$u \in l^2(U) \Rightarrow G(u) \in l^2(Y) \tag{2.13}$$

The map G has finite l^2-gain if there exist finite constants γ and b such that for all $N \geq 1$

$$\|(G(u))_N\|_2 \leq \gamma \|u_N\|_2 + b, \forall u \in l_e^2(U) \tag{2.14}$$

holds. Equivalently G has finite l^2-gain if there exist finite constants $\hat{\gamma} > \gamma$ and $\hat{b}$ such that for all $N \geq 1$ [127, (2.21)]

$$\|(G(u))_N\|_2^2 \leq \hat{\gamma}^2 \|u_N\|_2^2 + \hat{b}, \forall u \in l_e^2(U) \tag{2.15}$$

holds.

Note 1 *If G has finite l^2-gain then it is sufficient for l^2-stability. Let $u \in l^2(U)$ and $N \to \infty$ which leads (2.14) to*

$$\|(G(u))\|_2 \leq \gamma \|u\|_2 + b, \forall u \in l^2(U) \tag{2.16}$$

which implies (2.13) (see [127, p. 4] for continuous time case).

Lemma 1 *[127, Lemma 2.2.13] The l^2-gain $\gamma(G)$ is given as*

$$\gamma(G) = \inf\{\hat{\gamma} \mid \exists\, \hat{b} \textit{ s.t. (2.15) holds}\} \tag{2.17}$$

Next we will present definitions for various types of passivity for discrete time systems.

Definition 3 *[25, 127] Let $G : l_e^2(U) \to l_e^2(U)$ then for all $u \in l_e^2(U)$ and all $N \geq 1$:*

I. *G is passive if there exists some constant β such that (2.18) holds.*

$$\langle G(u), u\rangle_N \geq -\beta \tag{2.18}$$

II. *G is strictly-output passive if there exists some constants β and $\epsilon > 0$ such that (2.19) holds.*

$$\langle G(u), u\rangle_N \geq \epsilon\|(G(u))_N\|_2^2 - \beta \tag{2.19}$$

III. *G is strictly-input passive if there exists some constants β and $\delta > 0$ such that (2.20) holds.*

$$\langle G(u), u\rangle_N \geq \delta\|u_N\|_2^2 - \beta \tag{2.20}$$

Theorem 1 *Let $G : l_e^2(U) \rightarrow l_e^2(U)$ be strictly-output passive. Then G has finite l^2-gain.*

Proof 1 *The proof for the discrete case is practically the same as for the continuous case given in [127, Theorem 2.2.14], for completeness we denote $y = G(u)$, and rewrite (2.19)*

$$\begin{aligned}
\epsilon\|y_N\|_2^2 &\leq \langle y, u\rangle_N + \beta \\
&\leq \langle y, u\rangle_N + \beta + \frac{1}{2}\|\frac{1}{\sqrt{\epsilon}}u_N - \sqrt{\epsilon}y_N\|_2^2 \\
&\leq \beta + \frac{1}{2\epsilon}\|u_N\|_2^2 + \frac{\epsilon}{2}\|y_N\|_2^2
\end{aligned} \tag{2.21}$$

thus moving all terms of y to the left, (2.21), has the final form of (2.15) with l^2-gain $\hat{\gamma} = \frac{1}{\epsilon}$ and $\hat{b} = \frac{2\beta}{\epsilon}$.

The requirement for *strictly-output passive* is a relatively easy requirement to obtain for a *passive* plant with map G and input u and output y. This is accomplished by closing the loop relative to a reference vector r with a positive definite feedback gain matrix $K > 0$ such that $u = r - Ky$.

Theorem 2 *Given a passive system with input u, output $G(u) = y$, a positive definite matrix $K > 0$, and reference vector r. If the input $u = r - Ky$, then the mapping $G_{cl} : r \to y$ is strictly-output passivity which implies l^2-stability.*

Proof 2 *First we use the definition of passivity for G and substitute the feedback formula for u.*

$$\langle y, u \rangle_N = \langle y, r - Ky \rangle_N \geq -\beta \tag{2.22}$$

Then we can obtain the following inequality

$$\langle y, r \rangle_N \geq \lambda_m(K) \|y\|_2^2 - \beta \tag{2.23}$$

in which $\lambda_m(K) > 0$ is the minimum eigenvalue for K. Hence, (2.23) has the form of (2.19) which shows strictly-output passive and implies l^2-stability.

It is important to note that for very small maximum eigenvalues, the system is essentially the nominal passive system we started with. This is important, for we can design more general passive digital controllers and modify them with this simple transform to make them *strictly-output passive*.

2.2.2 Inner-product Equivalent Sample and Hold

In this section we prove Theorem 3 which shows how a (non)linear (*strictly input* or *strictly output*) passive plant can be transformed to a discrete (*strictly input*) passive plant using a particular digital sampling and hold scheme. This novel zero-order digital to analog hold, and sampling scheme introduced by [118] creates a combined system

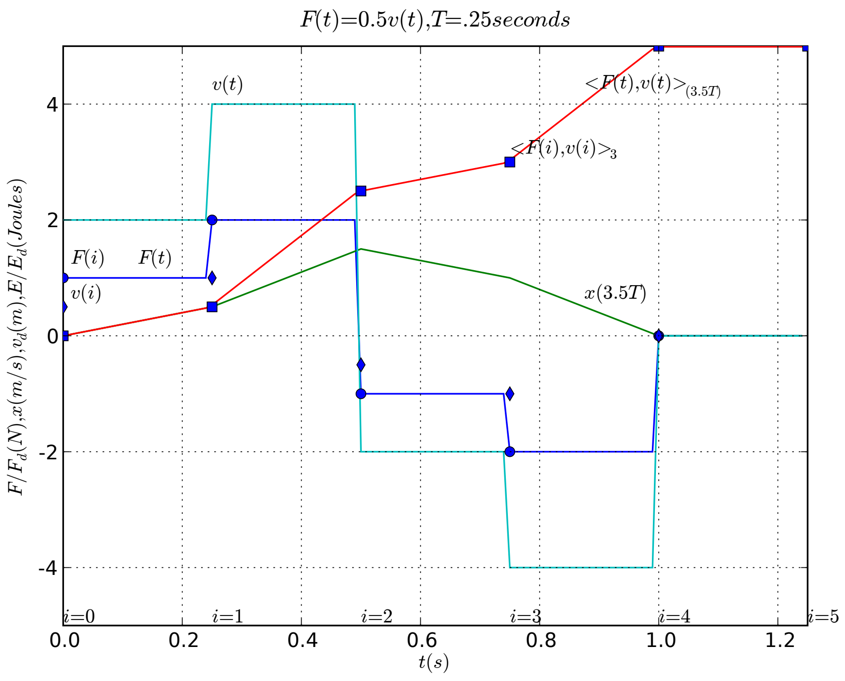

Figure 2.4. Illustration showing $\langle v(i), F(i)\rangle_N = \langle v(t), F(t)\rangle_{NT}$

such that the energy exchange between the analog and digital port is equal. This equivalence allows one to interconnect an analog to a digital Port-Controlled Hamiltonian (*PCH*) system which yields an overall passive system. In [103], a correction was made to the original scheme proposed in [118]. In order to prove Theorem 3, we will restate the sample and hold algorithm with a slightly modified nomenclature. Fig. 2.4 shows a simple example of a continuous force, $F(t)$ (solid blue line), being applied to a damper with damping ratio 0.5 (kg/s-m). The force is updated at a rate of T seconds, such that at $t = iT$ the corresponding discrete force, $F(i)$ (circles), updates $F(t)$ and is held for an additional T seconds. The discrete "velocity", $v(i)$ (diamonds), is defined as $v(i) = (x(i+1) - x(i))$. The discrete "position", $x(i)$, is the sampled integral of the continuous velocity, $v(t)$ (solid magenta line), up to time $t = iT$. Likewise $x(i+1)$ is the sampled integral of the *predicted* continuous velocity up to time $t + T$. Note that the solid green line, $x(t)$ denotes the integral of the continuous velocity. Finally, the continuous inner-product integral, $\langle F(t), v(t)\rangle_{NT} \triangleq \int_0^{NT} \langle F(t), v(t)\rangle$, is denoted by the solid red line. The discrete inner-product summation, $\langle v(i), F(i)\rangle_N$, is indicated at each index i with a blue square, thus showing equivalence to $\langle F(t), v(t)\rangle_{NT}$.

Definition 4 *[103, 118] Let a continuous one-port plant be denoted by the input-output mapping $G_{ct} : L_e^2(U) \to L_e^2(U)$. Denote continuous time as t, the discrete time index as i, the continuous input as $u(t) \in L_e^2(U)$, the continuous output as $y(t) \in L_e^2(U)$, the transformed discrete input as $u(i) \in l_e^2(U)$, and the transformed discrete output as $y(i) \in l_e^2(U)$. The inner-product equivalent sample and hold (IPESH) is implemented as follows:*

I. $x(t) = \int_0^t y(\tau)d\tau$

II. $y(i) = x((i+1)T) - x(iT)$

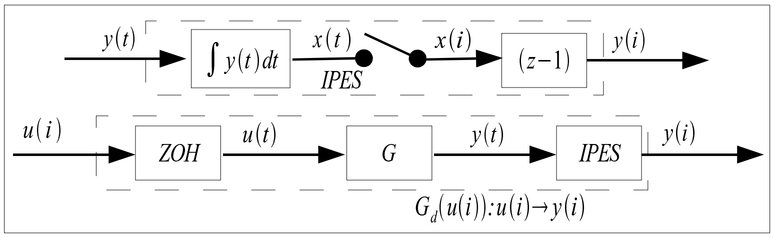

Figure 2.5. An implementation of the *IPESH* for *LTI* systems.

III. $u(t) = u(i), \forall t \in [iT, i(T+1))$

As a result

$$\langle y(i), u(i)\rangle_N = \langle y(t), u(t)\rangle_{NT}, \forall N \geq 1 \tag{2.24}$$

holds.

Theorem 3 *Using the IPESH given in Definition 4, the following relationships can be stated between the continuous one-port plant, G_{ct}, and the discrete transformed one-port plant, $G_d : l_e^2(U) \rightarrow l_e^2(U)$:*

I. *If G_{ct} is passive then G_d is passive.*

II. *If G_{ct} is strictly-input passive then G_d is strictly-input passive.*

III. *If G_{ct} is strictly-output passive then G_d is strictly-input passive.*

This is a general result, in which Theorem 3-I includes the special case where the input is a force and the output is a velocity ([103, Definition 2]) and it includes the special case for interconnecting *PCH* systems ([118, 119, Theorem 1]).

Proof 3 *I. Since the continuous passive system G_{ct} satisfies*

$$\langle y(t), u(t)\rangle_\tau \geq -\beta, \forall \tau \geq 0 \tag{2.25}$$

then by substituting (2.24) into (2.25) results in

$$\langle y(i), u(i) \rangle_N \geq -\beta, \forall N \geq 1 \tag{2.26}$$

which satisfies (2.18) and completes the proof of Theorem 3-I.

II. *Let $\tau = NT$, then since the continuous strictly-input passive system G_{ct} satisfies*

$$\langle y(t), u(t) \rangle_\tau \geq \delta \|u(t)_\tau\|_2^2 - \beta, \forall \tau \geq 0 \tag{2.27}$$

and Definition 4-III implies

$$\|u(t)_\tau\|_2^2 = T \|u(i)_N\|_2^2 \tag{2.28}$$

substituting (2.28) and (2.24) into (2.27) results in

$$\langle y(i), u(i) \rangle_N \geq T\delta \|u(i)_N\|_2^2 - \beta, \forall N \geq 1 \tag{2.29}$$

therefore, the transformed discrete system G_d satisfies (2.20) and completes the proof of Theorem 3-II.

III. *Let $\tau = NT$, then since the continuous strictly-output passive system G_{ct} satisfies*

$$\langle y(t), u(t) \rangle_\tau \geq \epsilon \|y(t)_\tau\|_2^2 - \beta, \forall \tau \geq 0 \tag{2.30}$$

however, no direct relationship can be made between $\|y(t)_\tau\|_2^2$ and $\|y(i)_N\|_2^2$. But Definition 4-III still implies (2.28), and since G_{ct} is strictly-output passive, which *implies finite l^2-gain such that*

$$\begin{aligned}\|y(t)_\tau\|_2^2 &\leq \frac{1}{\epsilon^2}\|u(t)_\tau\|_2^2 + \frac{2\beta}{\epsilon} \\ &\leq \frac{T}{\epsilon^2}\|u(i)_N\|_2^2 + \frac{2\beta}{\epsilon}\end{aligned} \tag{2.31}$$

holds. Substituting (2.31) into (2.30) results in

$$\langle y(i), u(i)\rangle_N \geq \frac{T}{\epsilon}\|u(i)_N\|_2^2 - (-\beta), \forall N \geq 1 \tag{2.32}$$

therefore, the transformed discrete system G_d satisfies (2.20) and completes the proof of Theorem 3-III.

Continuous and discrete linear time invariant systems have an important property, namely that if they are *strictly-input passive* they have *finite L^2/l^2-gain* and are *strictly-output passive* (See Corollary 10 in Appendix A.1).

Corollary 1 *Using the IPESH given in Definition 4, the following relationships can be stated between the continuous LTI one-port plant, G_{ct}, and the discrete transformed LTI one-port plant, $G_d : l_e^2(U) \to l_e^2(U)$: If G_{ct} is either strictly-input passive or strictly-output passive then G_d is both strictly-input passive with finite l^2-gain and strictly-output passive.*

2.3 Main Results

Fig. 2.6 depicts our proposed control scheme in order to guarantee l^2 stability in which the feedback and control data can be subject to variable delays between the controller and the plant. Depicted is a continuous passive plant $G_p(e_p(t)) = f_p(t)$ which is actuated by a zero-order hold and sampled by an *IPES*. Thus G_p is transformed

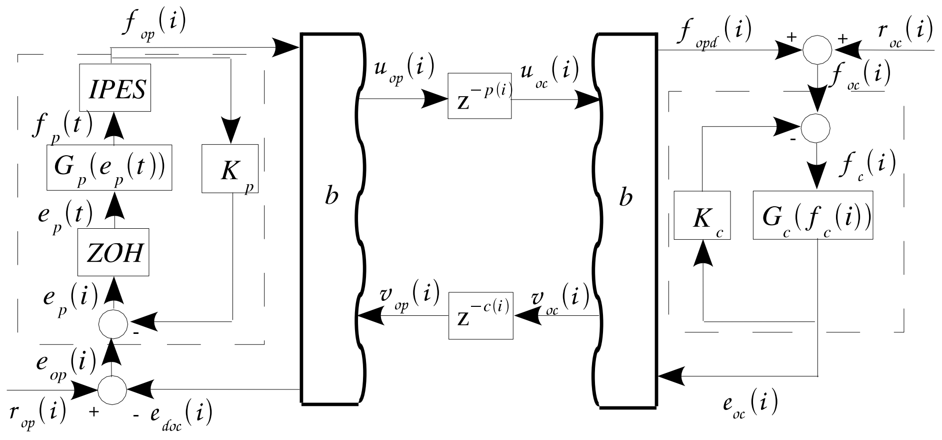

Figure 2.6. l^2-*stable* digital control network for cooperative scheduler

into a discrete passive plant $G_{dp}(e_p(i)) = f_{op}(i)$. Next, a positive definite matrix K_p is used to create a discrete *strictly-output passive* plant $G_{op}(e_{op}(i)) = f_{op}(i)$ outlined by the dashed line. Next G_{op} is interconnected in the following feed-back configuration such that

$$\langle f_{op}, e_{doc}\rangle_N = \frac{1}{2}(\|(u_{op})_N\|_2^2 - \|(v_{op})_N\|_2^2) \tag{2.33}$$

holds due to the wave transform. Moving left to right towards the *strictly-output passive* digital controller $G_{oc}(f_{oc}) = e_{oc}$ we first note that

$$\langle f_{opd}, e_{oc}\rangle_N = \frac{1}{2}(\|(u_{oc})_N\|_2^2 - \|(v_{oc})_N\|_2^2) \tag{2.34}$$

holds due to the wave transform. The wave variables $u_{oc}(i), v_{op}(i)$ are related to the corresponding wave variables $u_{op}(i), v_{oc}(i)$ and by the discrete time varying delays

$p(i), c(i)$ such that

$$u_{oc}(i) = u_{op}(i - p(i)) \tag{2.35}$$

$$v_{op}(i) = v_{oc}(i - c(i)) \tag{2.36}$$

(2.35) and (2.36) hold. Finally the positive definite matrix K_c is used to make the *passive* digital controller $G_c(f_c(i)) = e_{oc}(i)$ *strictly-output passive*. Typically, r_{oc} can be considered the set-point in which $f_{opd}(i) \approx -r_{oc}(i)$ at steady state, while $r_{op}(i)$ can be thought as a discrete disturbance. Which leads us to the following theorem.

Theorem 4 *The system depicted in Fig. 2.6 is l^2-stable if*

$$\langle f_{op}, e_{doc} \rangle_N \geq \langle e_{oc}, f_{opd} \rangle_N \tag{2.37}$$

holds for all $N \geq 1$.

Proof 4 *First, by theorem 3-I, G_p is transformed to a discrete passive plant. Next, by theorem 2 both the discrete plant and controller are transformed into a strictly-output passive systems. The strictly-output passive plant satisfies*

$$\langle f_{op}, e_{op} \rangle_N \geq \epsilon_{op} \|(f_{op})_N\|_2^2 - \beta_{op} \tag{2.38}$$

while the strictly-output passive controller satisfies (2.39).

$$\langle e_{oc}, f_{oc} \rangle_N \geq \epsilon_{oc} \|(e_{oc})_N\|_2^2 - \beta_{oc} \tag{2.39}$$

Substituting, $e_{doc} = r_{op} - e_{op}$ and $f_{opd} = f_{oc} - r_{oc}$ into (2.37) yields

$$\langle f_{op}, r_{op} - e_{op} \rangle_N \geq \langle e_{oc}, f_{oc} - r_{oc} \rangle_N$$

which can be rewritten as

$$\langle f_{op}, r_{op}\rangle_N + \langle e_{oc}, r_{oc}\rangle_N \geq \langle f_{op}, e_{op}\rangle_N + \langle e_{oc}, f_{oc}\rangle_N \tag{2.40}$$

so that we can then substitute (2.38) and (2.39) to yield

$$\langle f_{op}, r_{op}\rangle_N + \langle e_{oc}, r_{oc}\rangle_N \geq \epsilon(\|(f_{op})_N\|_2^2 + \|(e_{oc})_N\|_2^2) - \beta \tag{2.41}$$

in which $\epsilon = \min(\epsilon_{op}, \epsilon_{oc})$ and $\beta = \beta_{op} + \beta_{oc}$. Thus (2.41) satisfies (2.19) in which the input is the row vector of $[r_{op}, r_{oc}]$, and the output is the row vector $[f_{op}, e_{oc}]$ and completes the proof.

Theorem 5 *The system depicted in Fig. 2.6 without the IPESH in which i and t denote continuous time is L^2-stable if*

$$\langle f_{op}, e_{doc}\rangle_\tau \geq \langle e_{oc}, f_{opd}\rangle_\tau \tag{2.42}$$

holds for all $\tau \geq 0$.

Proof 5 *The proof is completely analogous to the proof given for Theorem 4, the differences being that the IPESH is no longer involved and the discrete time delays are replaced with continuous time delays.*

In order for (2.37) to hold, the communication channel/ data-buffer needs to remain *passive*. It has been proved in [119] that the discrete communication channel is passive for both fixed delays [119, Proposition 1] and variable time delays including loss of packets [119, Proposition 2], as we will show with a different and straight forward proof.

Lemma 2 *If the discrete time varying delays are fixed $p(i) = p, c(i) = c$ in which $0 \leq p, c < N$ and/or data packets are dropped then (2.37) holds.*

Before we begin the proof, we denote the partial sum from M to N of an extended norm as follows

$$\|x_{(M,N)}\|_2^2 \triangleq \langle x^*, x\rangle_{(M,N)} = \sum_{i=M}^{N-1} \langle x^*, x\rangle \tag{2.43}$$

Proof 6 *In order to satisfy (2.37), (2.33) minus (2.34) must be greater than zero, or*

$$\begin{aligned}(\|(u_{op})_N\|_2^2 - \|(v_{op})_N\|_2^2) - (\|(u_{oc})_N\|_2^2 - \|(v_{oc})_N\|_2^2) \geq 0 \\ (\|(u_{op})_N\|_2^2 - \|(u_{oc})_N\|_2^2) + (\|(v_{oc})_N\|_2^2 - \|(v_{op})_N\|_2^2) \geq 0\end{aligned} \tag{2.44}$$

$$\begin{aligned}(\|(u_{op})_N\|_2^2 - \|(u_{op}(i-p(i))_N\|_2^2) + \\ (\|(v_{oc})_N\|_2^2 - \|(v_{oc}(i-c(i))_N\|_2^2) \geq 0\end{aligned} \tag{2.45}$$

holds. Clearly (2.45) holds when the delays are fixed, as (2.45) can be written to show

$$(\|(u_{op})_{((N-p),N)}\|_2^2 + \|(v_{oc})_{((N-c),N)}\|_2^2) \geq 0 \tag{2.46}$$

the inequality always holds for all $0 \leq p, c < N$. Note if p and c equal zero, then inequality in (2.46) becomes an equality. If all the data packets were dropped then, $\|(u_{oc})_N\|_2^2) = 0$ and $\|(v_{op})_N\|_2^2) = 0$, such that (2.37) holds and all the energy is dissipated. If only part of the data packets are dropped, the effective inequality described by (2.45) serves as a lower bound ≥ 0; hence dropped data packets do not violate (2.37).

[119, Proposition 2] is too broad in stating that the communication channel is passive in spite of variable time delays when only the transmission of one data packet per sample period occurs. For instance, a simple counter example is to assume $p(i) = i$, then (2.45) will not hold if $N\|(u_{op})_1\|_2^2 > (\|(u_{op})_N\|_2^2 + \|(v_{oc})_N\|_2^2)$. Clearly other variations can be given such that $p(i)$ eventually becomes fixed and never changes after sending old *duplicate samples*, and still (2.37) will not hold. Therefore, we state the following lemma:

Lemma 3 *The discrete time varying delays $p(i), c(i)$ can vary arbitrarily as long as (2.45) holds. Thus, the main assumption (2.37) for Theorem 4 will hold if:*

1. *Duplicate transmissions are dropped at the receivers. This can be accomplished by transmitting the tuple (i,$u_{op}(i)$), if $i \in \{$ the set of received indexes $\}$ then set $u_{oc}(i) = 0$.*

2. *we drop received data in order that (2.45) holds. This requires us to track the current energy storage in the communication channel.*

Note 2 *Examples of similar energy-storage audits as stated in Lemma 3-2 are given in [102, Section IV] which does not use wave-variables, and in [91] which is for the continuous time case.*

2.3.1 Passive Discrete *LTI* System Synthesis

In [20], using dissipative theory and a longer proof than we will provide, it was shown how to synthesize a discrete passive plant from a linear time invariant (*LTI*) plant. The advantage of the observer described in [20] is that it does not require a measurement of the integrated output of the passive plant. However, if one is concerned with controlling the integrated output such as position, one will probably have this measurement as well as the corresponding passive output such as velocity. We will also show how an observer, based on the integrated output measurement can still be used. Such an observer maintains passivity and eliminates the need to directly measure the actual passive output such as the velocity. The proof for the observer will follow a similar proof in [21].

A passive continuous time *LTI* system, $H(s)$, which is described by the following state space representation $\{\mathbf{A} \in \mathbb{R}^{\mathbf{n}\times\mathbf{n}}, \mathbf{B} \in \mathbb{R}^{\mathbf{n}\times\mathbf{p}}, \mathbf{C} \in \mathbb{R}^{\mathbf{p}\times\mathbf{n}}, \mathbf{D} \in \mathbb{R}^{\mathbf{p}\times\mathbf{p}}\}$ is cascaded in series with a diagonal matrix of integrators, $H_I(s)$, described by
$\{\mathbf{A_I} = \mathbf{0}, \mathbf{B_I} = \mathbf{I}, \mathbf{C_I} = \mathbf{I}, \mathbf{D_I} = \mathbf{0}\}$. The combined system, $H_o(s) = H(s)H_I(s)$, is described by $\{\mathbf{A_o}, \mathbf{B_o}, \mathbf{C_o}\}$. Where

$$\mathbf{A_o} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(n+p)\times(n+p)} \tag{2.47}$$

$$\mathbf{B_o} = \begin{bmatrix} \mathbf{B} \\ \mathbf{D} \end{bmatrix} \in \mathbb{R}^{(n+p)\times p} \tag{2.48}$$

$$\mathbf{C_o} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{p\times(n+p)} \tag{2.49}$$

Applying a zero-order-hold and an ideal sampler, the system is described by [34]

$$\begin{aligned} x(k+1) &= \mathbf{\Phi_o} x(k) + \mathbf{\Gamma_o} u(k) \\ p(k) &= \mathbf{C_o} x(k) \end{aligned} \tag{2.50}$$

in which

$$\begin{aligned} \mathbf{\Phi_o} &= e^{\mathbf{A_o}T} \\ \mathbf{\Gamma_o} &= \int_0^T e^{\mathbf{A_o}\eta} d\eta \mathbf{B_o} \end{aligned} \tag{2.51}$$

.

Proposition 1 *A passive continuous time LTI system, $H(s)$, can be converted to a discrete passive LTI system, $G_p(z)$ at a sample rate T in which the discrete state equations are*

$$\begin{aligned} x(k+1) &= \mathbf{\Phi_o} x(k) + \mathbf{\Gamma_o} u(k) \\ y(k) &= \mathbf{C_p} x(k) + \mathbf{D_p} u(k) \end{aligned} \tag{2.52}$$

in which $\mathbf{C_p} = \mathbf{C_o}(\mathbf{\Phi_o} - \mathbf{I})$, and $\mathbf{D_p} = \mathbf{C_o}\mathbf{\Gamma_o}$.

Proof 7 *From Definition 4 it is a simple exercise to compute the passive output $y(k) = p(k+1) - p(k)$ as follows*

$$\begin{aligned} x(k+1) &= \mathbf{\Phi_o} x(k) + \mathbf{\Gamma_o} u(k) \\ y(k) &= \mathbf{C_o}(\mathbf{\Phi_o} - \mathbf{I}) x(k) + \mathbf{C_o}\mathbf{\Gamma_o} u(k) \end{aligned} \tag{2.53}$$

hence $\mathbf{C_p} = \mathbf{C_o}(\mathbf{\Phi_o} - \mathbf{I})$, and $\mathbf{D_p} = \mathbf{C_o}\mathbf{\Gamma_o}$ which completes the proof.

Using Proposition 1 and Theorem 2 the following corollary can be shown:

Corollary 2 *Given a positive definite matrix $\mathbf{K_x} > 0$ and discrete passive system described by (2.52), the system*

$$\begin{aligned} x(k+1) &= \mathbf{\Phi_{sp}} x(k) + \mathbf{\Gamma_{sp}} u(k) \\ y(k) &= \mathbf{C_{sp}} x(k) + \mathbf{D_{sp}} u(k) \end{aligned} \tag{2.54}$$

is strictly-output passive. Here

$$\begin{aligned}
\mathbf{\Phi_{sp}} &= \mathbf{\Phi_o} - \mathbf{\Gamma_o}\mathbf{K_x}(\mathbf{I} + \mathbf{D_p}\mathbf{K_x})^{-1}\mathbf{C_p} \\
\mathbf{\Gamma_{sp}} &= \mathbf{\Gamma_o}(\mathbf{I} - \mathbf{K_x}(\mathbf{I} + \mathbf{D_p}\mathbf{K_x})^{-1}\mathbf{D_p}) \\
\mathbf{C_{sp}} &= (\mathbf{I} + \mathbf{D_p}\mathbf{K_x})^{-1}\mathbf{C_p} \\
\mathbf{D_{sp}} &= (\mathbf{I} + \mathbf{D_p}\mathbf{K_x})^{-1}\mathbf{D_p}
\end{aligned} \tag{2.55}$$

With our discrete *strictly-output passive* system we can scale the gain so that its steady state gain matches the *strictly-output passive* continuous systems steady state gain.

Corollary 3 *Given a diagonal matrix $\mathbf{K_s} > 0$ and discrete strictly-output passive system described by (2.54), the following system is strictly-output passive*

$$\begin{aligned}
x(k+1) &= \mathbf{\Phi_{sp}}x(k) + \mathbf{\Gamma_{sp}}u(k) \\
y(k) &= \mathbf{K_s}\mathbf{C_{sp}}x(k) + \mathbf{K_s}\mathbf{D_{sp}}u(k)
\end{aligned} \tag{2.56}$$

in which each diagonal element

$$k_s(i) = \begin{cases} y_c(i)/y_d(i) \forall i \in \{1, \ldots, p\} & \textit{if } y_c(i) \textit{ and } y_d(i) \neq 0; \\ \frac{1}{T} & \textit{otherwise} \end{cases} \tag{2.57}$$

The vectors y_c/y_d correspond to the respective steady state continuous/discrete output of a strictly-output passive plant given a unit step input. These vectors can be computed as follows:

$$\begin{aligned}
y_c &= (-\mathbf{C_c}\mathbf{A_c}^{-1}\mathbf{B_c} + \mathbf{D_c})\mathbf{1} \\
y_d &= H_{sp}(z=1)\mathbf{1},\ H_{sp}(z) = \mathbf{C_{sp}}(z\mathbf{I} - \mathbf{\Phi_{sp}})^{-1}\mathbf{\Gamma_{sp}} + \mathbf{D_{sp}}
\end{aligned} \tag{2.58}$$

where

$$\begin{aligned}
\mathbf{G_x} &= \mathbf{I} + \mathbf{D}\mathbf{K_x} \\
\mathbf{C_c} &= \mathbf{G_x}^{-1}\mathbf{C} \\
\mathbf{D_c} &= \mathbf{G_x}^{-1}\mathbf{D} \\
\mathbf{A_c} &= \mathbf{A} - \mathbf{B}\mathbf{K_x}\mathbf{C_c} \\
\mathbf{B_c} &= \mathbf{B}(\mathbf{I} - \mathbf{K_x}\mathbf{D_c})
\end{aligned} \tag{2.59}$$

Next, the following corollary provides a method to compute $u_{op}(k), f_{op}(k)$ given r_{op}, v_{op}, b. We can also synthesize the digital controller from a continuous model using the *IPES* with *ZOH* as well, so an additional corollary will show how to compute $v_{oc}(k), e_{oc}(k)$ given $u_{oc}(k), r_{oc}(k)$.

Corollary 4 *The following state equation describes the relationship between the inputs r_{op}, v_{op} and scattering gain b to the outputs $u_{op}(k), f_{op}(k)$.*

$$\begin{aligned}
x(k+1) &= \mathbf{\Phi_{ef}}x(k) + \mathbf{\Gamma_{ef}}(\sqrt{2b}v_{op}(k) + r_{op}(k)) \\
f_{op}(k) &= \mathbf{C_{ef}}x(k) + \mathbf{D_{ef}}(\sqrt{2b}v_{op}(k) + r_{op}(k)) \\
u_{op}(k) &= \sqrt{2b}f_{op}(k) - v_{op}(k)
\end{aligned} \tag{2.60}$$

Here

$$\begin{aligned}\mathbf{G} &= \mathbf{I} + b\mathbf{K_s D_{sp}} \\ \mathbf{C_{ef}} &= \mathbf{G}^{-1}\mathbf{K_s C_{sp}} \\ \mathbf{D_{ef}} &= \mathbf{G}^{-1}\mathbf{K_s D_{sp}} \\ \mathbf{\Phi_{ef}} &= \mathbf{\Phi_{sp}} - b\mathbf{\Gamma_{sp} C_{ef}} \\ \mathbf{\Gamma_{ef}} &= \mathbf{\Gamma_{sp}}(\mathbf{I} - b\mathbf{D_{ef}}) \end{aligned} \tag{2.61}$$

Corollary 5 *The following state equation describes the relationship between the inputs* r_{oc}, u_{oc} *and scattering gain* b *to the outputs* $v_{oc}(k), e_{oc}(k)$.

$$\begin{aligned} x(k+1) &= \mathbf{\Phi_{fe}}x(k) + \mathbf{\Gamma_{fe}}(\sqrt{\frac{2}{b}}u_{oc}(k) + r_{oc}(k)) \\ e_{oc}(k) &= \mathbf{C_{fe}}x(k) + \mathbf{D_{fe}}(\sqrt{\frac{2}{b}}u_{oc}(k) + r_{oc}(k)) \\ v_{oc}(k) &= u_{oc}(k) - \sqrt{\frac{2}{b}}e_{oc}(k) \end{aligned} \tag{2.62}$$

Where

$$\begin{aligned} \mathbf{G_1} &= \mathbf{I} + \frac{1}{b}\mathbf{K_s D_{sp}} \\ \mathbf{C_{fe}} &= \mathbf{G_1}^{-1}\mathbf{K_s C_{sp}} \\ \mathbf{D_{fe}} &= \mathbf{G_1}^{-1}\mathbf{K_s D_{sp}} \\ \mathbf{\Phi_{fe}} &= \mathbf{\Phi_{sp}} - \frac{1}{b}\mathbf{\Gamma_{sp} C_{fe}} \\ \mathbf{\Gamma_{fe}} &= \mathbf{\Gamma_{sp}}(\mathbf{I} - \frac{1}{b}\mathbf{D_{fe}}) \end{aligned} \tag{2.63}$$

In order to prove that a state observer can be used in a *strictly-input passive* manner, we require the following lemma.

Lemma 4 *[135] The discrete LTI system (2.52) is strictly-input passive (strictly-positive real(SPR)) if and only if there exists a symmetric positive definite matrix* $\mathbf{P}$ *that satisfies the following LMI:*

$$\begin{bmatrix} \mathbf{\Phi_o}^{\mathsf{T}}\mathbf{P}\mathbf{\Phi_o} - \mathbf{P} & (\mathbf{\Gamma_o}^{\mathsf{T}}\mathbf{P}\mathbf{\Phi_o} - \mathbf{K_s}\mathbf{C_p})^{\mathsf{T}} \\ \mathbf{\Gamma_o}^{\mathsf{T}}\mathbf{P}\mathbf{\Phi_o} - \mathbf{K_s}\mathbf{C_p} & -(\mathbf{K_s}\mathbf{D_p} + \mathbf{D_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}} - \mathbf{\Gamma_o}^{\mathsf{T}}\mathbf{P}\mathbf{\Gamma_o}) \end{bmatrix} < 0 \tag{2.64}$$

Note 3 *Therefore by Theorem 3-(II,III) any continuous strictly-input passive or strictly-output passive LTI system which is sampled and actuated by an IPESH will satisfy (2.64).*

Note 4 *We also added* K_s *in order to show that any positive diagonal matrix can be used to scale the output* $y(k)$ *as is done with our observer described by (2.65).*

We now propose the following state observer, based on the sampled integrated output of the *strictly-input passive* or *strictly-output passive* plant and the corresponding output estimate $\hat{y}(k)$:

$$\begin{aligned} \hat{x}(k+1) &= \mathbf{\Phi_o}\hat{x}(k) + \mathbf{\Gamma_o}u(k) - \mathbf{K_e}(\hat{p}(k) - p(k)) \\ \hat{p}(k) &= \mathbf{C_o}\hat{x}(k) \\ \hat{y}(k) &= \mathbf{K_s}\mathbf{C_p}\hat{x}(k) + \mathbf{K_s}\mathbf{D_p}u(k) \end{aligned} \tag{2.65}$$

This observer is similar to the observer proposed in [21] except that it is based on the sampled integrated output, and our focus is on how the observer applies to *strictly-input passive* plants. Defining the error in the state estimate as $e(k) \overset{\triangle}{=} \hat{x}(k) - x(k)$ and the augmented observer state vector as $x_{ob}(k) \overset{\triangle}{=} [x(k), e(k)]$ the system dynamics are

$$\begin{aligned} x_{ob}(k+1) &= \mathbf{\Phi_{ob}} x_{ob}(k) + \mathbf{\Gamma_{ob}} u(k) \\ \hat{y}(k) &= \mathbf{K_s C_{ob}} x_{ob}(k) + \mathbf{K_s D_p} u(k) \end{aligned} \tag{2.66}$$

where

$$\begin{aligned} \mathbf{\Phi_{ob}} &= \begin{bmatrix} \mathbf{\Phi_o} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Phi_o} - \mathbf{K_e C_o} \end{bmatrix} \\ \mathbf{\Gamma_{ob}} &= \begin{bmatrix} \mathbf{\Gamma_o} \\ \mathbf{0} \end{bmatrix} \\ \mathbf{C_{ob}} &= \begin{bmatrix} \mathbf{C_p} & \mathbf{C_p} \end{bmatrix} \end{aligned} \tag{2.67}$$

Proposition 2 *If the sampled LTI system is strictly-input passive or strictly-output passive and K_e is chosen so that the eigenvalues of $\mathbf{\Phi_o} - \mathbf{K_e C_o}$ are inside the unit circle the observer described by (2.65) is both strictly-input passive with finite l^2-gain and strictly-output passive.*

Proof 8 *First by choosing the eigenvalues to be inside the unit circle there exist two matrices $\mathbf{Q_2} > 0$ and $\mathbf{P_o} > 0$ such that the following Lyapunov inequality is satisfied*

$$-\mathbf{Q_2} = (\mathbf{\Phi_o} - \mathbf{K_e C_o})^{\mathsf{T}} \mathbf{P_o} (\mathbf{\Phi_o} - \mathbf{K_e C_o}) < 0 \tag{2.68}$$

In order to satisfy the requirements of Lemma 4 we consider the following symmetric positive definite matrix

$$\mathbf{P_{ob}} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mu \mathbf{P_o} \end{bmatrix} > 0 \tag{2.69}$$

and show that there exists a $\mu > 0$ that satisfies (2.73). Note the following inequalities hold from our original strictly-input passive system.

$$\begin{aligned}
-\mathbf{Q_1} &= \mathbf{\Phi_o}^{\mathsf{T}}\mathbf{P}\mathbf{\Phi_o} - \mathbf{P} < 0 \\
-\mathbf{Q_3} &= -(\mathbf{K_s}\mathbf{D_p} + \mathbf{D_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}} - \mathbf{\Gamma_{ob}}^{\mathsf{T}}\mathbf{P_{ob}}\mathbf{\Gamma_{ob}}) \\
&= -(\mathbf{K_s}\mathbf{D_p} + \mathbf{D_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}} - \mathbf{\Gamma_o}^{\mathsf{T}}\mathbf{P}\mathbf{\Gamma_o}) < 0 \qquad (2.70)
\end{aligned}$$

To simplify the expression we define

$$\mathbf{C_1} \overset{\triangle}{=} \mathbf{\Gamma_o}^{\mathsf{T}}\mathbf{P}\mathbf{\Phi_o} - \mathbf{K_s}\mathbf{C_p} \qquad (2.71)$$

Therefore the proposed strictly-input passive system described by (2.66) has to satisfy

$$\begin{bmatrix} \mathbf{Q_1} & \mathbf{0} & -\mathbf{C_1^{\mathsf{T}}} \\ \mathbf{0} & \mu\mathbf{Q_2} & -\mathbf{C_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}} \\ -\mathbf{C_1} & -\mathbf{K_s}\mathbf{C_p} & \mathbf{Q_3} \end{bmatrix} > 0 \qquad (2.72)$$

Using a similarity transformation, (2.72) is equivalent to

$$\begin{bmatrix} \mathbf{Q_1} & -\mathbf{C_1^{\mathsf{T}}} & \mathbf{0} \\ -\mathbf{C_1} & \mathbf{Q_3} & -\mathbf{K_s}\mathbf{C_p} \\ \mathbf{0} & -\mathbf{C_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}} & \mu\mathbf{Q_2} \end{bmatrix} > 0 \qquad (2.73)$$

The following upper block matrix, $\mathbf{O}$, satisfies (2.64) due to Proposition 1, Theorem 3-(II,III), and Lemma 4.

$$\mathbf{O} = \begin{bmatrix} \mathbf{Q_1} & -\mathbf{C_1^{\mathsf{T}}} \\ -\mathbf{C_1} & \mathbf{Q_3} \end{bmatrix} > 0 \qquad (2.74)$$

Since $\mathbf{O} > 0$, and $Q_2 > 0$, then from using Proposition 8.2.3-v in [14] which is based on the Schur Complement Theory we need to show that

$$\mathbf{O} > 0, \text{ and} \tag{2.75}$$

$$\mu\mathbf{Q_2} - \begin{bmatrix} \mathbf{0} & -\mathbf{C_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}} \end{bmatrix} \mathbf{O}^{-1} \begin{bmatrix} \mathbf{0} \\ -\mathbf{K_s}\mathbf{C_p} \end{bmatrix} > 0$$

$$\mu\mathbf{Q_2} - \mathbf{C_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}}\mathbf{O}^{-1}\mathbf{K_s}\mathbf{C_p} > 0 \tag{2.76}$$

Thus denoting $\lambda_m(\cdot)/\lambda_M(\cdot)$ as the minimum/maximum eigenvalues for a matrix, noting that the similarity transform of $\mathbf{Q_2} = \mathbf{P_2}\mathbf{\Lambda_2}\mathbf{P_2}^{\mathsf{T}}$, and defining $\mathbf{M} \overset{\triangle}{=} \mathbf{C_p}^{\mathsf{T}}\mathbf{K_s}^{\mathsf{T}}\mathbf{O}^{-1}\mathbf{K_s}\mathbf{C_p}$, μ needs to satisfy

$$\mu > \frac{\lambda_M(\mathbf{P_2}^{\mathsf{T}}(\mathbf{M} + \mathbf{M}^{\mathsf{T}})\mathbf{P_2})}{2\lambda_m(\mathbf{Q_2})} \tag{2.77}$$

Therefore μ exists and satisfies (2.73) which completes the proof.

Note that the proof given in [21] only shows *sufficiency* for *passive* systems and implicitly assumes that the discrete sampled plant is *strictly-input passive*. Furthermore, the results there can not be applied to our desired design of an observer which uses the integrated output of a *strictly-input passive* or *strictly-output passive* plant.

Since we are using the observer on continuous *LTI* systems which are either *strictly-input passive* with *finite L^2-gain*, or *strictly-output passive* and the corresponding discrete observer is both *strictly-input passive* with *finite l^2-gain* and *strictly-output passive* we can simplify our implementation by setting the feedback gain $K_p = 0$ in Fig. 2.6. We note that K_p may still be helpful in converting a continuous passive signal into a discrete *strictly-output passive* signal with an observer, however we found the analysis to be quite difficult. Similar to Corollary 4, we present Corollary 6 as it applies to using an observer of a *strictly-output passive* plant.

Corollary 6 *If using an observer for either a LTI system which is strictly-input passive with finite gain or is strictly-output passive, the following state equations describe the relationship between the inputs r_{op}, v_{op} and scattering gain b to the outputs $\hat{u}_{op}(k), \hat{f}_{op}(k)$.*

$$\begin{aligned}
\hat{x}(k+1) &= \mathbf{\Phi_{efo}}\hat{x}(k) + \mathbf{\Gamma_{efo}}(\sqrt{2b}v_{op}(k) + r_{op}(k)) + \mathbf{K_e}p(k) \\
\hat{f}_{op}(k) &= \mathbf{C_{efo}}\hat{x}(k) + \mathbf{D_{efo}}(\sqrt{2b}v_{op}(k) + r_{op}(k)) \\
\hat{u}_{op}(k) &= \sqrt{2b}\hat{f}_{op}(k) - v_{op}(k)
\end{aligned} \tag{2.78}$$

where

$$\begin{aligned}
\mathbf{G} &= \mathbf{I} + b\mathbf{K_s}\mathbf{D_p} \\
\mathbf{C_{efo}} &= \mathbf{G}^{-1}\mathbf{K_s}\mathbf{C_p} \\
\mathbf{D_{efo}} &= \mathbf{G}^{-1}\mathbf{K_s}\mathbf{D_p} \\
\mathbf{\Phi_{efo}} &= \mathbf{\Phi_o} - \mathbf{K_e}\mathbf{C_o} - b\mathbf{\Gamma_o}\mathbf{C_{efo}} \\
\mathbf{\Gamma_{efo}} &= \mathbf{\Gamma_o}(\mathbf{I} - b\mathbf{D_{efo}})
\end{aligned} \tag{2.79}$$

Note that Corollary 6 describes a standard observer not connected to a wave junction when $b = 0$. See Appendix A.2 for detailed equations related to simulating such an actual observer.

Corollary 7 *If using an observer for either a LTI system which is strictly-input passive with finite gain or is strictly-output passive, the following state equations describe the relationship between the inputs r_{oc}, u_{oc} and scattering gain b to the outputs*

$\hat{v}_{oc}(k), \hat{e}_{oc}(k)$.

$$\begin{aligned}\hat{x}(k+1) &= \mathbf{\Phi_{feo}}\hat{x}(k) + \mathbf{\Gamma_{feo}}(\sqrt{\frac{2}{b}}u_{oc}(k) + r_{oc}(k)) + \mathbf{K_e}p(k)\\ \hat{e}_{oc}(k) &= \mathbf{C_{feo}}\hat{x}(k) + \mathbf{D_{feo}}(\sqrt{\frac{2}{b}}u_{oc}(k) + r_{oc}(k))\\ \hat{v}_{oc}(k) &= u_{oc}(k) - \sqrt{\frac{2}{b}}\hat{e}_{oc}(k)\end{aligned} \tag{2.80}$$

where

$$\begin{aligned}\mathbf{G_1} &= \mathbf{I} + \frac{1}{b}\mathbf{K_s}\mathbf{D_p}\\ \mathbf{C_{feo}} &= \mathbf{G_1}^{-1}\mathbf{K_s}\mathbf{C_p}\\ \mathbf{D_{feo}} &= \mathbf{G_1}^{-1}\mathbf{K_s}\mathbf{D_p}\\ \mathbf{\Phi_{feo}} &= \mathbf{\Phi_o} - \mathbf{K_e}\mathbf{C_o} - \frac{1}{b}\mathbf{\Gamma_o}\mathbf{C_{feo}}\\ \mathbf{\Gamma_{feo}} &= \mathbf{\Gamma_o}(\mathbf{I} - \frac{1}{b}\mathbf{D_{feo}})\end{aligned} \tag{2.81}$$

See Appendix A.2 for detailed equations related to simulating such an actual observer.

2.3.2 Stable Control With A Cooperative Scheduler

SOS is an operating system for embedded devices with wireless transceivers such as the Berkeley motes. *SOS* uses a high priority and low priority queues with timers which signal a task through the queue in order to implement the soft real time scheduler (note that most other operating systems such as *TinyOS* which use just a single FIFO message queue could be used to notify the control task as well) [44]. For simplicity we will use *SOS* to discuss one possible implementation for our l^2*-stable* control system illustrated in Fig. 2.6. The following is an outline for a suitable device driver:

1. Provide an interface for the controller to register a function to enable the device driver to send $u_{op}(i)$ to. Also allow the controller to specify a desired sample time T, wave impedance b, and K_p (note K_p does not need to be a matrix, it could be a scalar to modify all parts of $f_{op}(i)$ equally. Note that the driver will buffer $v_{oc}(i)$ while the controller will buffer $u_{op}(i)$.

2. Provide an interface for the controller to send outgoing $v_{oc}(i)$ to.

3. Calculate $f_{op}(i)$ based on the *IPES* given in Definition 4-I,II.

4. Calculate the corresponding $u_{op}(i)$, and $e_{doc}(i)$ based on the buffered $v_{oc}(i)$, the servicing of the buffer is where the $v_{op}(i-c(i))$ delay comes in effect. Since data can be popped directly from the buffer, we do not need to worry about counting duplicate data. For simplicity if the buffer begins to get full we can safely drop data.

5. With the new $e_{doc}(i)$ and $f_{op}(i)$, calculate $e_p(i) = -e_{doc}(i) - K_p f_{op}(i)$ and apply to *ZOH*.

The controller, is notified by the driver through the high-priority queue and implements the right side of Fig. 2.6. Note that the lower-priority queue can be used for more time-consuming tasks, such as changing control parameters and loading new modules. This may cause temporary delays, but l^2*-stability* will be maintained. Note that old data does not have to be simply dropped (which satisfies Lemma 2) in order for the system to recover from these longer periodic delays. Using Lemma 3-2 we can calculate the two-norm of all M, in which $i = 0, 1, ..., M-1$ of the non-processed inputs $s(u_{op}, M) = \|u_{op}(i)\|_2$ and multiply it by the sign of the sum of the non-processed inputs $sn(u_{op}, M) = \texttt{sgn}(\sum_{i=0}^{M-1} u_{op}(i))$ such that the input for $u_{oc}(i) = sn(u_{op}, M)s(u_{op}, M)$. This will improve tracking and highlights why we split the buffers appropriately. The driver can do a similar calculation in order to calculate $v_{op}(i)$.

2.4 Simulation

We shall control a motor with an ideal current source, which will allow us to neglect the effects of the motor inductance and resistance for simplicity. The fact that the current source is non-ideal, leads to a non-passive relationship between the desired motor current and motor velocity [121]. There are ways to address this problem using passive control techniques by controlling the motors velocity indirectly with a switched voltage source and a minimum phase current feedback technique [64], and more recently incorporating the motors back voltage measurement which provides an exact tracking error dynamics passive output feedback controller [65].

The motor is characterized by its torque constant, $K_m > 0$, back-emf constant K_e, rotor inertia, $J_m > 0$, and damping coefficient $B_m > 0$. The dynamics are described by

$$\dot{\omega} = -\frac{B_m}{J_m}\omega + \frac{K_m}{J_m}i \tag{2.82}$$

and are in a (strictly) positive real form which is a necessary and sufficient condition for (strict input) passivity [133, Section V.A.2)] [135, Defintion 1]. We choose to use the passive “proportional-derivative” controller described by (2.7) and define $\tau = \frac{B}{K}$ in order to factor out K and yield

$$K_{PD}(s) = K\frac{\tau s + 1}{s} \tag{2.83}$$

Using loop-shaping techniques we choose $\tau = \frac{J_m}{B_m}$ and choose $K = \frac{J_m \pi}{10 K_m T}$. This will provide a reasonable crossover frequency at roughly a tenth the Nyquist frequency and

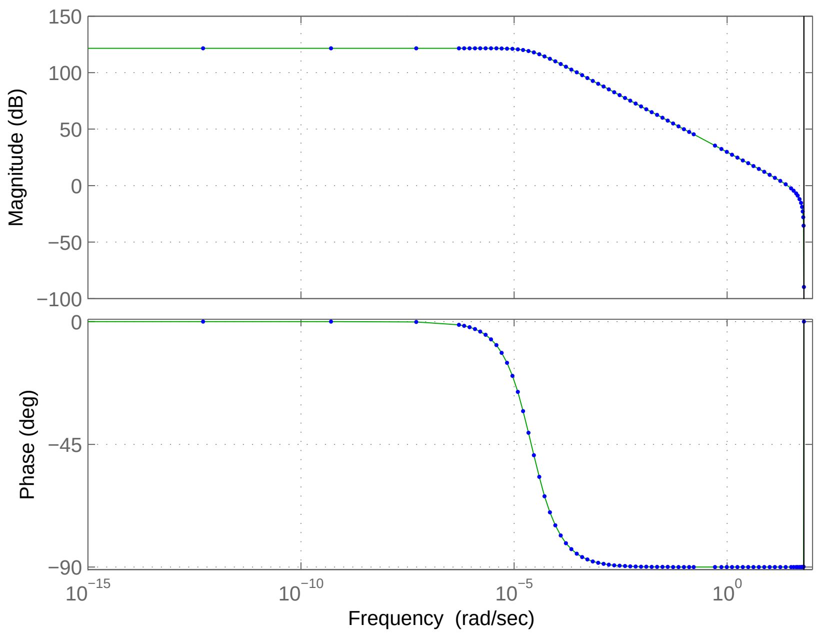

Figure 2.7. Bode plot depicting crossover frequency for baseline plant with observer and controller.

maintain a 90 degree phase margin. We choose to use the same motor parameter values given in [65] in which $K_m = 49.13$ (mV×rad ×sec), $J_m = 7.95 \times 10^{-3}$ (kg $\times$ m^2), and $B_m = 41(\mu\text{N} \times \text{sec}/\text{meter})$. With $T = .05$ seconds, we use Corollary 5 to synthesize a *strictly-output passive* controller from our continuous model (2.83), and Corollary 6 to implement the observer. We also use Corollary 3 in order to compute the appropriate gains for both the controller $K_{s_c} = 1$ and the *strictly-output passive* plant $K_{s_p} = 20$. Note that arbitrarily choosing $K_{s_c} = \frac{1}{T} = 20$ would have led to a incorrectly scaled system in which the crossover frequency would essentially equal the Nyquist frequency (only because a zero exists extremely close to -1 in the z-plane). Fig. 2.7, Fig. 2.8, and Fig. 2.9 indicates that our baseline system performs as expected. We chose

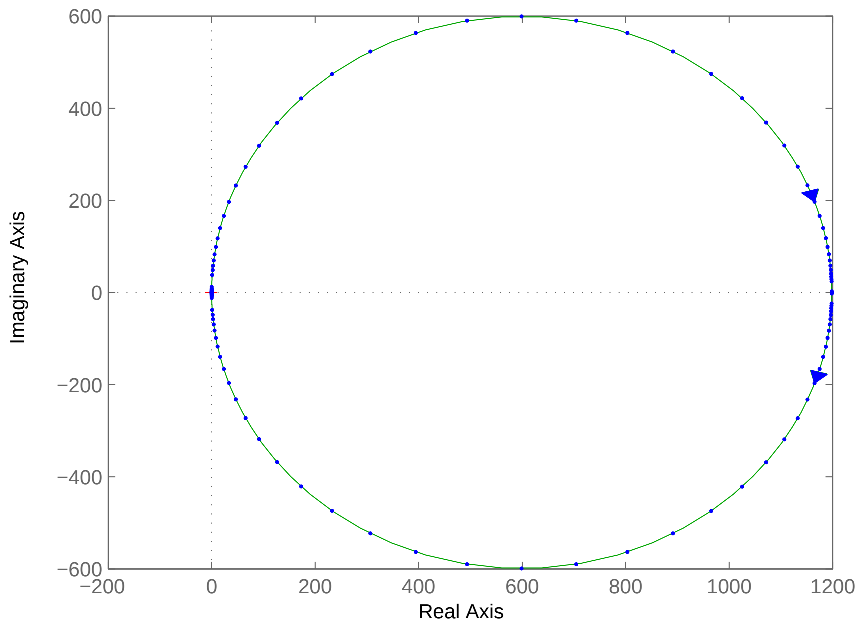

Figure 2.8. Nyquist plot for the continuous plant (solid line) and the synthesized discrete counterpart (solid dots) with observer.

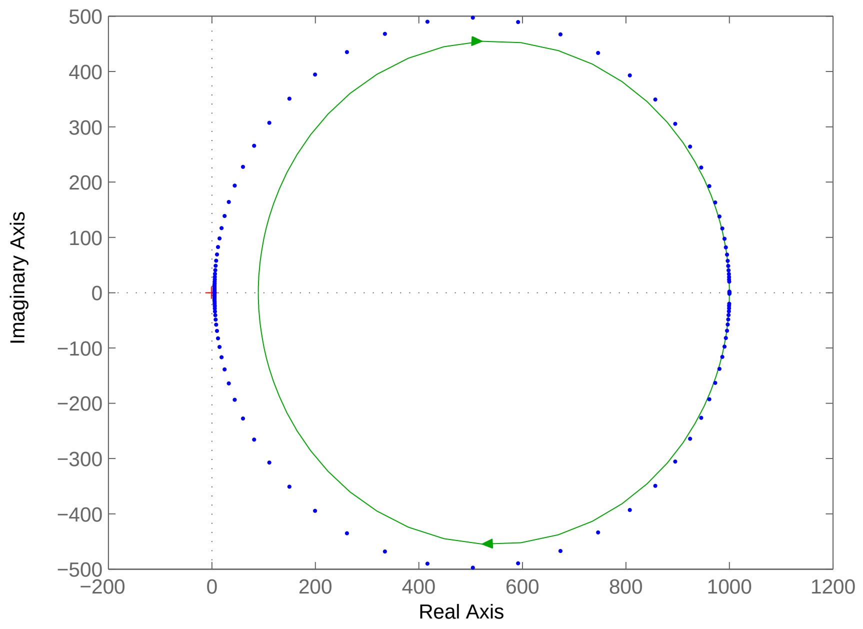

Figure 2.9. Nyquist plot for the continuous controller (solid line) and the synthesized discrete counterpart (solid dots).

$K_e = [16.193271, 1.799768]^{\mathsf{T}}$ for our observer in which the poles are equal to a tenth of the poles of the discrete *passive* plant synthesized by Proposition 1, this by definition forces all the poles inside the unit circle. Since the plant is *strictly-output passive* we chose $K_p = 0$. For the controller we chose $K_c = 0.001$ in order to make it *strictly-output passive*. Fig. 2.10 shows the step response to a desired position set-point $\theta_d(k)$ which generates an approximate velocity reference for $r_{oc}(z) = -H_t(z)\theta_d(z)$. $H_t(z)$ is a zero-order hold equivalent of $H_t(s)$, in which $\omega_{traj} = 2\pi$ and $\zeta = .9$.

$$H_t(s) = \frac{\omega_{traj}^2 s}{s^2 + 2\zeta\omega_{traj} + \omega_{traj}^2} \tag{2.84}$$

Note, that it is important to use a second order filter in order to achieve near perfect tracking, a first order filter resulted in significant steady state position errors for relatively slow trajectories. Finally in Fig. 2.11 we see that the proposed control network maintains similar performance as the baseline system. Note that by increasing $b = 5$ significantly reduced the overshoot caused by a half second delay (triangles $b = 1$/squares $b = 5$). Also note that even a two second delay (large circles $b = 5$) results in only a delayed response nearly identical to the baseline system.

2.5 The Study of Interconnected Passive Systems Using Wave Variables.

One apparent limitation with the use of wave variables in control theory is that typically they are connected in a series configuration in order to preserve *passive* mapping (i.e. Figure 2.6). Yet there do exist other ways to interconnect wave variables for *LTI* systems as is done with the design of wave digital filters [32]. The manner in which wave ports are interconnected in order to realize a digital filter is done differently than is done in control implementations. For example in Figure 2.6 the waves u_{op} and u_{oc} are each computed in a manner similar to a voltage *incident* wave (a), and the waves

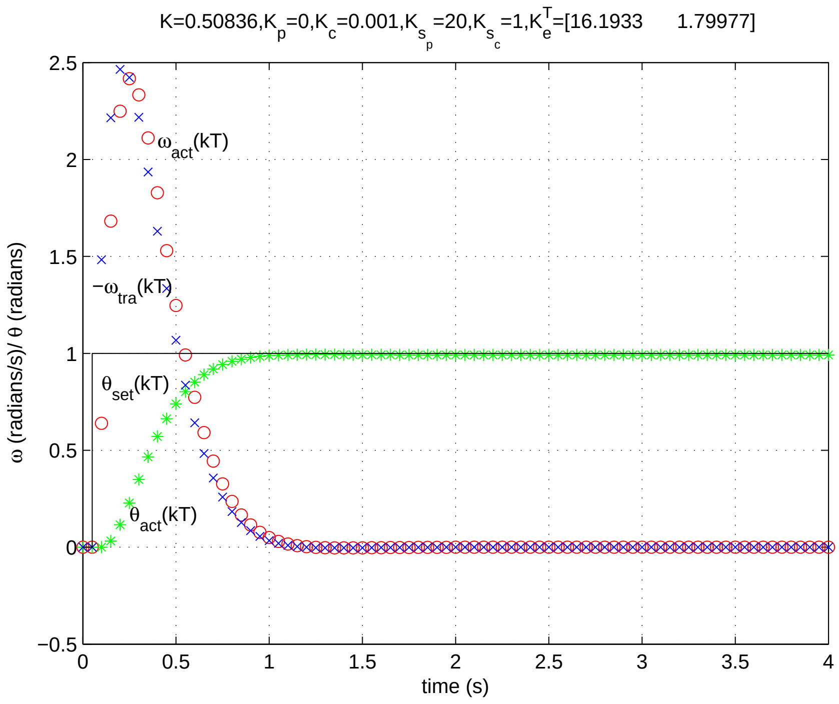

Figure 2.10. Baseline step response for motor with *strictly-output passive* digital controller and *strictly-output passive* observer.

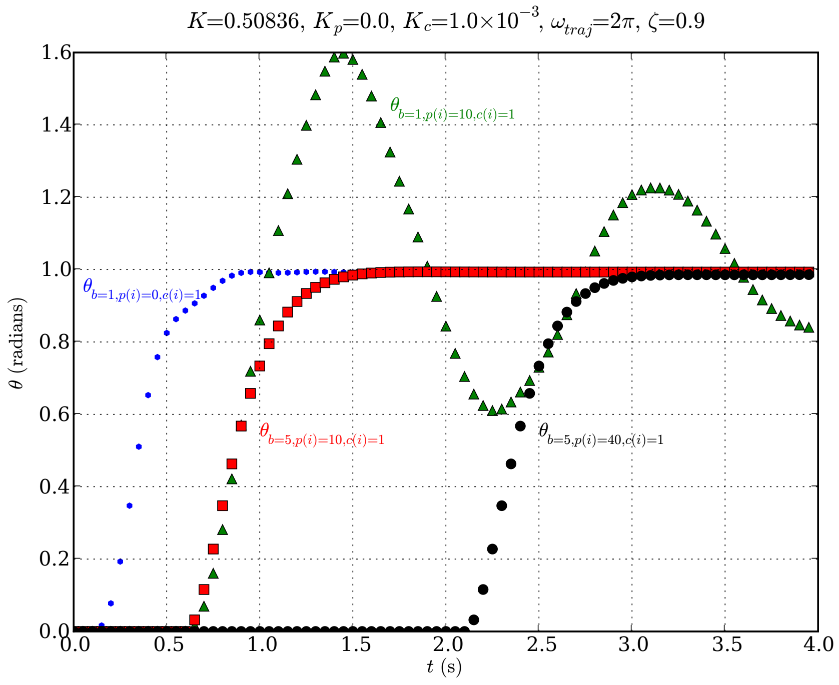

Figure 2.11. Step response for motor with *strictly-output passive* digital controller and *strictly-output passive* observer as depicted in Fig. 2.6 with delays.

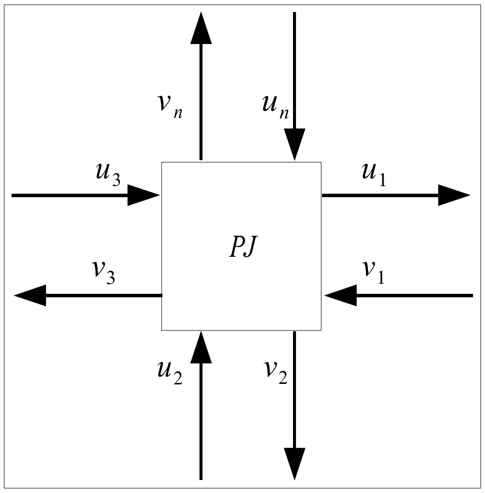

Figure 2.12. The *power junction*.

v_{op} and v_{oc} are each computed in a manner similar to a voltage *reflective* wave (b) [32]. For wave digital filters a voltage *incident* waves can be thought of as a wave traveling into a two port junction, likewise a *reflective* wave travels out of a two port junction. When interconnecting two port elements for a wave digital filter, a voltage *incident* wave should connect to a voltage *reflective* wave or vice-versa [32, Section IV-A-2)]. If we denote u_{op} and v_{oc} as a *reflective* waves (with outgoing arrows) and denote u_{oc} and v_{op} as *incident* waves (with incoming arrows), then the interconnection rules appear to be in agreement. Clearly, if we can straighten out these differences, then we can discuss various wave interconnections, such as series and parallel adaptors, etc. For example the *unit element* can be used to represent identical fixed delays $p = c$, and the *quasi-reciprocal line* (*QUARL*) can represent different fixed delays such that $p \neq c$ [32, Table 2]. In this remaining discussion, we continue to use the same definitions for wave variables as provided by [90] (i.e. (2.4), (2.5)). From the extensive literature search we have done in this area, we have yet to see wave variables connected as follows:

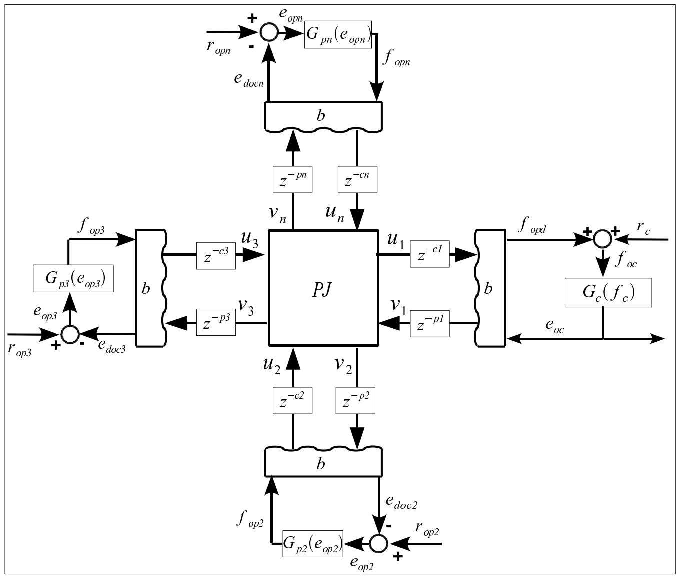

Figure 2.13. An example of a *power junction* control network.

Definition 5 *A "power junction" is implemented as follows (see Figure 2.12):* n *systems with the corresponding wave variable pairs* $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$ *are interconnected such that the* (u_1, v_1) *pair is a corresponding (power output, power input) pair and the remaining* $(u_k, v_k),\ k \in \{2, \ldots, n\}$ *are (power input, power output) pairs. The power junction is passive and lossless as long as*

$$(u_1^2 - v_1^2) = \sum_{k=2}^{n}(u_k^2 - v_k^2) \tag{2.85}$$

always holds. The following is sufficient to satisfy (2.85):

$$u_1^2 = \sum_{k=2}^{n} u_k^2 \tag{2.86}$$
$$v_1^2 = \sum_{k=2}^{n} v_k^2$$

Note 5 *Clearly this can be generalized to the case for vectors, and for having more than one pair of wave variables directed in an opposite direction as the remaining pairs. There are numerous ways such a "power junction" can be implemented, for example:*

1. *Let there be one controller G_1 with the corresponding wave variables (u_1, v_1) in which v_1 is the control output, and u_1 represents the "weighted" feedback from the remaining $n-1$ plants $G_k, k \in \{2, \dots, n\}$.*
2. *Each plant G_k has the corresponding wave variables (u_k, v_k) in which u_k is the corresponding plant sensor output and v_k is the corresponding "distributed" command from the controller to each individual plant (see Figure 2.13).*
3. *A basic "average" power distribution can be implemented as follows:*

$$\begin{aligned} v_k &= \frac{v_1}{\sqrt{n-1}} \\ u_1 &= \mathtt{sgn}(\sum_{k=2}^{n} u_k)\sqrt{\sum_{k=2}^{n} u_k^2} \end{aligned} \tag{2.87}$$

2.6 Conclusions

We have presented a theory to design a digital control network which maintains $l^2 - stability$ in spite of time varying delays caused by cooperative schedulers. We presented a fairly complete, and needed l^2 stability analysis, in particular the results in Theorem 1, and Theorem 2 (for the discrete-time case) appeared to be lacking from the open literature and were necessary in order to complete our proof. The other new results (not available in the open literature) which led to a l^2-*stable* controller design are as follows:

1. Theorem 3-I is an improvement which captures all *passive* systems (not just *PCH*) systems.

2. Theorem 3-II, and Theorem 3-III are completely original (the latter forced us to require that the driver had to implement the additional feedback (K_p) calculation to obtain passivity for the nonlinear case).

3. Corollary 1 allows us to set $K_p = \mathbf{0}$ if the continuous *LTI* plant is either *strictly-input passive* or *strictly-output passive*.

4. Theorem 4 is a new and general theorem to interconnect continuous nonlinear passive plants which we hope will lead to more elaborate networks interconnected in the discrete time domain. Theorem 5 is also new, in which no knowledge of the energy storage function is required to show stability of the network.

5. Proposition 1 shows how to synthesize a discrete *passive LTI* system from a continuous one.

6. Corollary 2 and Corollary 3 show how to respectively make the discrete *passive* plant *strictly-output passive* (*strictly-positive real*) and scale the output so that it will match the steady state output for its continuous counterpart.

7. Corollary 4 and Corollary 5 show how to implement the *strictly-output passive* network depicted in Fig. 2.6.

8. Proposition 2 shows how to implement a discrete *strictly-output passive LTI* observer for either a *strictly-input passive* or *strictly-output passive* continuous *LTI* system.

9. Corollary 6 and Corollary 7 show how to implement the observer when attached to a scattering junction.

10. The “power junction” (Definition 5) paves a new way to create different control networks which consist of wave variables.

Theorem 2 allows us to directly design *low-sensitivity strictly-output passive* controllers using the *wave-digital filters* described in [32].

CHAPTER 3

WIRELESS CONTROL SYSTEMS SUBJECT TO ACTUATOR CONSTRAINTS

Wireless communication systems are subject to many additional disturbances which traditional wired communication systems are just not exposed to. Wireless systems have to contend with random fading channels due to changes in the environment such as interference, rain, heat, and absorbing objects crossing their communication path. These time varying changes in the channel influence the data capacity of the network. If a controller is sending data at a rate which exceeds the capacity of the channel, then either large delays will occur in the transmission of the data and/or data will have to be dropped in order to not exceed the data capacity of the channel. In order to design *wnecs* which can tolerate random communication dropouts, in one approach [108, 110] use results presented in the literature related to the control of linear systems whose state equations vary randomly [55] and can be described by a Markovian jump linear system (*MJLS*) [49, 50]. The random communication dropouts in a feedback control system are captured by *MJLS* which are further described in Appendix B.1. Section 3.1 provides a motivating example which contains new results showing how a discrete mean square stable control system of a continuous first order plant in which the pole is strictly in the right half plane and subject to independent Bernoulli data drop outs will be destabilized when subject to input actuator saturation. Furthermore, it shows that if the single pole of the plant is not in the right half plane, then stability can be maintained inspite of actuator saturation. These results lead to our justification for the need to develop *wnecs* in which the hierarchical design is such that the network is composed of stable subcomponents.

The l^2-*stable* networks presented in Chapter 2 can tolerate both time varying delays and data drop outs. However, memoryless input nonlinearities such as actuator saturation can eliminate the desired *passivity* properties of a given plant. In [15] it is shown how to use a nonlinear controller to compensate for a large class of memoryless input nonlinearities $\sigma(\cdot)$ such as actuator saturation. The nonlinear controller $\beta(\cdot)$ can then be integrated in to linear controller-plant systems such that the net system is Lyapunov stable if the linear controller and plant are both positive real. Furthermore the system will be globally asymptotically stable if the linear controller and plant are both strictly positive real. This technique has been extended to apply to systems consisting of either continuous time or discrete time networks in which the controller-plant system consists of a *passive* and *exponentially passive* pair [41, 42]. In order to isolate the plant from the controller as we did in Chapter 2 and indicated in Figure 2.6 we show how to locate the nonlinear controller $\beta(\cdot)$ at the output of the plant and prove that it *recovers the passivity* lost due to $\sigma(\cdot)$. The modified control network with $\beta(\cdot)$ is shown in Figure 3.7.

Section 3.2 begins with the presentation of a new Theorem 6 showing how input saturation eliminates the *passive* input-output mapping we desire. In Section 3.2.1 our discussion continuous with the review that output saturation and other **sector**$[0, \infty)$ nonlinearities do not eliminate the *passive* input-output mapping for certain classes of continuous *LTI passive* systems. Section 3.2.2 we introduce the *inner-product recovery block* (*IPRB*) which we show how it recovers the inner-product mapping which was lost due to the memoryless nonlinearity. Furthermore we provide the new important Theorem 9 showing how *IPRB* recovers *passivity*, *strictly-output passivity*, and *strictly-input* *passivity* for various memoryless input nonlinearities. Section 3.2.3 provides the necessary new corollaries and figures which show how the *IPRB* is effectively integrated with the *IPESH* blocks used for the l^2-*stable* digital control networks which we are studying. Which leads us to the final Section 3.2.4 which shows a l^2-*stable* digital control network subject to memoryless input nonlinearities and provides the corresponding new theorem and proof.

3.1 Markovian Jump Linear Systems Subject to Actuator Saturation.

There are no results in the literature relating stochastic stability for *MJLS* when the actuators saturate. We shall focus our discussion on designing a control law for a first order discrete *MJLS* which is subject to independent Bernoulli dropouts from a wireless sensor. We will first analyze a system which is not subject to actuator saturation and then complete our discussion when the actuator saturates. Our continuous plant has the following form:

$$\begin{aligned} \dot{x} &= ax + bu \\ y &= cx \end{aligned} \tag{3.1}$$

Using a zero order hold and an ideal sampler the discrete equivalent plant, sampled at a rate T seconds, has the following form:

$$\begin{aligned} x_{k+1} &= a_d x_k + b_d u_k \\ y_k &= c x_k \end{aligned} \tag{3.2}$$

The scalars a_d and b_d have the following form:

$$a_d = e^{aT} \tag{3.3}$$
$$b_d = b\int_0^T e^{as}ds = \frac{b}{a}(e^{aT} - 1)$$

We propose the following control law:

$$u_k = \begin{cases} K_a \frac{y_k}{c}, \text{ if } y_k \text{ successfully received by controller,} \\ 0, \text{ otherwise.} \end{cases} \tag{3.4}$$

Assume y_k has a probability p of being successfully received at the controller. Define $K = K_a b_d$. We can now describe the controlled system in terms of the discrete Markov state $r_k \in \{1, 2\}$ with the following form

$$x_{k+1} = a_d x_k + K\delta(r_k - 1)x_k \tag{3.5}$$

in which $r_k = 1$ for a successfully received message and $r_k = 0$ otherwise. In order to evaluate the average stability of the system we state the following definition for systems which can be described by a scalar state x.

Definition 6 *[55, Definition III$_m$] Lyapunov stability of the m^{th} mean (LSMM)*
The equilibrium solution is said to possess stability of the m^{th} mean if given $\epsilon > 0$, there exists $\delta(\epsilon, k_o)$ such that $|x_o| < \delta$ implies

$$E\{|x(k; x_o, k_o)|^m\} < \epsilon, \ \forall k \tag{3.6}$$

To emphasize the importance of taking the absolute value of x in the expectation we state the following definition.

Definition 7 *Weak Lyapunov stability of the mean (WLSM)*
The equilibrium solution is said to possess stability of the mean if given $\epsilon > 0$, there exists $\delta(\epsilon, k_o)$ such that $|x_o| < \delta$ implies

$$-\epsilon < E\{x(k; x_o, k_o)\} < \epsilon, \ \forall k \tag{3.7}$$

We will show that a component of the probability density function of x can not be eliminated with a controller when actuator saturation is present and the plant is *anti-stable* ($a_d > 1$ or $a > 0$). As a result *LSMM* is not achievable for a first order *anti-stable* linear time invariant system subject to actuator saturation. Using the notations $\mu_k = E\{x(k)\}$, the mean of the system can be described by the following difference equation

$$\mu_{k+1} = (a_d + pK)\mu_k \tag{3.8}$$

Note the pK term results due to the independence of K from the state $x(k)$ since the feedback channel is modeled as an independent Bernoulli channel in which each transmission from the sensor has a probability p of successfully transmitting a message to the controller. From (3.8) a necessary and sufficient condition for *WLSM* is

$$|a_d + pK| < 1 \tag{3.9}$$

Furthermore the allowable control gain range for K is in the following bounds given by (3.10), note how the allowable range for K increases as p drops.

$$\frac{-(1 + a_d)}{p} < K < \frac{1 - a_d}{p} \tag{3.10}$$

It is interesting to note that a unique K exists to achieve *WLSM* a system for all $0 < p \leq 1$ and any finite x_o. Yet, it doesn't imply that the state x converges to zero. For example, the probabilistic dead-beat controller, when $K = \frac{-a_d}{p}$, will drive the state x further away from the origin if $p < \frac{1}{2}$ and $x_o > 0$; furthermore, if $a_d > 1$ the state $|x(k)| \rightarrow \infty$ as $k \rightarrow \infty$. Next we discuss the limiting effects of actuator saturation for our first order system. We show that the controllability region becomes bounded or equivalently if the state $|x| > |\frac{u_{max}}{(1-a_d)}|$ and $a_d > 1$ then the system (3.5) is uncontrollable and $|x(k)| \rightarrow \infty$.

Most systems have finite actuator authority. We can show directly that a stable system can be safely controlled when the actuator saturates as long as the actuator has some finite authority. For stable systems, in which $a_d < 1$ and the actuator has a control range $-u_{max} \leq u \leq u_{max}$, if $K = -a_d$ then any $u_{max} > 0$ will stabilize the system provided that we hold the control output u for jT seconds such that the following equality holds.

$$(a_d + K)x_k = a_d^j x_k + (\sum_{i=0}^{j-1} a_d^i)u \tag{3.11}$$

We are now converging at a different rate when the actuator saturates because we are re-defining the time interval between each T sample to vary at a rate jT in which j is the minimum integer required to obtain a u in which $|u| < u_{max}$. To simplify the discussion, a dead-beat controller is one in which the controller can drive the state of the plant to 0 in one time step in this first order system. For this system, the dead-beat controller is when $K = -a_d$. Note our bounds shown in (3.10) show that a dead-beat controller is always stable if $a_d \leq 1$. When $a_d > 1$ the mean stability is dependent on p; however, for deterministic systems this is always a stable gain with the maximum gain margin. As such, we can solve for a minimal fixed u capable of driving the state to 0 in

j steps by solving (3.11) in which $K = -a_d$.

$$\begin{aligned} u &= \frac{-a_d^j x_o}{\sum_{i=0}^{j-1} a_d^i} \\ &= \begin{cases} \frac{-(1-a)a_d^j}{1-a_d^j} x_o, & \text{if } a_d \neq 1; \\ \frac{-x_o}{j}, & \text{if } a_d = 1. \end{cases} \end{aligned} \tag{3.12}$$

From (3.12) we now can readily make the following conclusions in regards to the closed loop stability of these systems with saturation. If the system is stable ($a_d < 1$) or ***semi-stable*** ($a_d = 1$), then a fixed $|u| > 0$ for a finite j exists such that the system can be driven to 0 for any x_o. This is shown by solving the limit of $j \to \infty$ for the cases shown to solve u in (3.12). When $a_d = 1$ it can be easily shown that the lower bound for $u = 0$ in the limit as $j \to \infty$.

$$\begin{aligned} u \lim_{j\to\infty} &= \frac{-(1-a_d)a_d^j}{1-a_d^j} x_o \\ &= \frac{-(1-a_d)0}{1-0} x_o = 0x_o = 0 \qquad \text{if } a_d < 1 \end{aligned} \tag{3.13}$$

If the system is ***anti-stable*** ($a_d > 1$), then the range for which u can stabilize the system for a finite j is if $|u| > |(1-a_d)x_o|$ which is less restrictive than the dead-beat controller which would require $|u| \geq |a_d x_o|$. The proof is simply taking the limits for (3.12) as $j \to \infty$.

$$\begin{aligned} u \lim_{j\to\infty} &= \frac{-(1-a_d)a_d^j}{1-a_d^j} x_o \\ &= \frac{-(1-a_d)\infty}{1-\infty} x_o = (1-a_d)x_o \qquad \text{if } a > 1 \end{aligned} \tag{3.14}$$

For systems with probabilistic drop outs we can no longer claim a controller can achieve stability in the absolute mean if the initial system is *anti-stable*. For a first-order *anti-stable* system subject to actuator saturation if $|x_o| > |\frac{u_{max}}{(1-a)}|$ then the system is uncontrollable and x grows to ∞. Hence the statistics related to the probability of $x \neq 0$ will be finite in the limit as $k \to \infty$ which implies that *LSMM* is impossible to achieve with any controller. This is easily shown for the case with a dead-beat controller ($K = -a_d$). To be precise, we define $x_k = 0$ if a saturated actuator can still drive the state to zero if x_k is successfully received over the Bernoulli channel. The probability density function (*pdf*) which we denote as $p(x_k)$ has the following form:

$$p(x_k) = \begin{cases} [1-(1-p)^k]\delta(x_k) + (1-p)^k\delta(x_k - x_o a_d^k) & \text{if } 0 \leq k \leq k_{max}; \\ [1-(1-p)^{k_{max}}]\delta(x_k) + (1-p)^{k_{max}}\delta(x_k - x_o a_d^k) & \text{if } k > k_{max} \end{cases} \tag{3.15}$$

k_{max} is the integer in which the uncompensated state will grow to a point in which the controller can not recover and is determined by (3.14). An uncompensated state will continue to grow at the rate a_d^k until a successful feedback signal is received; hence, $k_{max} = \lceil \frac{\ln(u_{max}) - \ln(a_d - 1)}{\ln(a_d)} \rceil$. To keep the statistics, and discussion simple in (3.15), we assumed that the controller maintains $u = 0$ once the state $|x_k| > |\frac{u_{max}}{(1-a)}|$. In no way does this assumption affect the conclusion, that *LSMM* is impossible to attain with our system. With (3.15) any m^{th} moment can be computed such as the absolute mean ($m = 1$).

$$E\{|x_k|\} = \begin{cases} (1-p)^k |x_o| a_d^k & \text{if } 1 \geq k \leq k_{max}; \\ (1-p)^{k_{max}} |x_o| a_d^k & \text{if } k > k_{max} \end{cases} \tag{3.16}$$

Since $|x_o| > 0$ implies $k_{max} < \infty$ which implies that in the limit as $k \to \infty$ the $E\{|x_k|\} = \infty$. *Furthermore* this unbounded component of the absolute mean will be present regardless of what controller is chosen. This implies that stability of the second moment is impossible.

$$E\{x_k^2\} = \begin{cases} (1-p)^k x_o^2 a_d^{2k} & \text{if } 1 \geq k \leq k_{max}; \\ (1-p)^{k_{max}} x_o^2 a_d^{2k} & \text{if } k > k_{max} \end{cases} \tag{3.17}$$

The first part of (3.17) is identical for a non-saturated system, or a system in which $a_d \leq 1$. Instead of deriving the first part of (3.17) from the *pdf*, we can derive it from the following recursive equation:

$$\begin{aligned} E\{x_{k+1}^2\} &= E\{(a_d + K)^2 x_k^2\} = (a_d^2 + 2a_d pK + pK^2)E\{x_k^2\} \\ &= (a_d^2 + 2a_d pK + pK^2)^{k+1} E\{x_o^2\} \\ &= (a_d^2 + 2a_d pK + pK^2)^{k+1} x_o^2 \end{aligned} \tag{3.18}$$

Equation (3.18) notes that convergence of the second mean is possible iff $|a_d^2 + 2a_d pK + pK^2| < 1$. Rewriting this inequality and solving for the limits for the acceptable range of K which is possible to stabilize a system without saturation we find that $0 < a_d < \sqrt{\frac{1}{1-p}}$. The upper-limit is based on finding the zeros for K of the following inequality:

$$K^2 + 2a_d K + \frac{a_d^2 - 1}{p} < 0 \tag{3.19}$$

The solution for the zeros is of the following form:

$$K_z = -a_d \pm \sqrt{\frac{1}{p}[a_d^2(p-1) + 1]} \tag{3.20}$$

Equation (3.20) shows that the dead-beat controller $K = -a_d$ will always be a viable controller, and the only one possible in the limit for a_d. Furthermore we find that the dead-beat control gain $K = -a_d$ possesses the maximum gain margin which allows us to have the most robust gain if there is uncertainty with a_d. Hence a necessary and sufficient test for stabilizability of the absolute mean and second moment is to determine if a dead-beat controller exists.

In summary, it is possible to safely control a first order stable and *semi-stable* system with i.i.d. Bernoulli drop outs and actuator saturation; however, it is impossible to stabilize an *anti-stable* system with regards to the absolute mean and second moment. A controllable *LTI* system which is subject to actuator saturation and is *anti-stable* has a *null controllability region* $\mathcal{C}$. $\mathcal{C} \neq \mathbb{R}^n$, but is a bounded convex open set containing the origin [43]. A controllable *LTI* system subject to actuator saturation which is *semi-stable* has a *null controllability region* $\mathcal{C} = \Re^n$ [114]. A *semi-stable* system which is discrete will have all poles inside and/or on the unit circle (note this allows duplicate poles on the unit circle), in contrast to a discrete *anti-stable* system which will have all poles outside the unit circle. Likewise, a continuous *semi-stable* system will have all poles on the imaginary axis and/or in the left-half of the complex plane, in contrast to the continuous *anti-stable* system which has all poles in the right-half of the complex plane [46]. For example, a mass is *semi-stable*.

These results suggest that it may be impossible to stabilize in the absolute mean ($m = 1$) and second moment ($m = 2$) any *anti-stable LTI* system which is controllable, subject to actuator saturation that relies on feed-back over a Bernoulli channel. Furthermore, these results suggest that a controller can stabilize in the absolute mean and second moment any stable and *semi-stable LTI* system which is controllable, subject to actuator saturation and relies on feed-back over a Bernoulli channel.

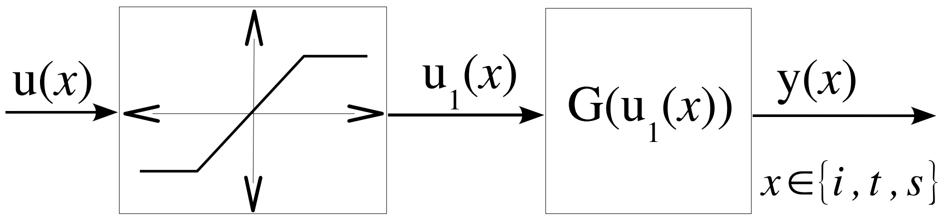

Figure 3.1. Actuator saturation depicted for either continuous time (t), discrete time (i), or Laplace domain (s).

3.2 *Passive* Systems Subject to Saturation

The input saturation block indicated in Figure 3.1 is a special type of actuator input memoryless nonlinearity $\sigma(u(x))$ in which $u(x) \in \Re^m$ with components $u_j(x),\ j \in \{1,\ldots,m\}$ which has the following form

$$u_{s_j}(x) = \begin{cases} k_j u_j(x),\ \text{if } |u_j(x)| \leq k_j u_{\max_j}(x) \\ k_j u_{\max_j} \texttt{sgn}(u_j(x)),\ \text{otherwise} \end{cases} \tag{3.21}$$

in which each element can have separate linear gain k_j and saturation level $k_j u_{\max_j} \texttt{sgn}(u_j(x))$. For the discussion $G(u_s(x))$ can be thought as either linear or nonlinear *passive* continuous or discrete time mapping in which $x \in \{i,t\}$, where i is a discrete time index, t represents continuous time. We denote $\langle u(x), y(x)\rangle_X$ as either the continuous time ($x = t,\ X = T$) inner-product or the discrete time ($x = i,\ X = N$) inner-product.

Theorem 6 *If for a passive system $G(u_s(x))$ which is subject to the input saturation nonlinearity described by (3.21) and there exists any reachable $u(x)$, $y(x)$ waveform* *(sequence) in which any input-output pair* $(u(x), y(x))$ *satisfies*

$$u_{s_J}^{\mathsf{T}}(t)y_J(t) = \epsilon < 0, \text{ for any } t \in (0, T], \text{ or} \tag{3.22}$$

$$u_{s_J}^{\mathsf{T}}(i)y_J(i) = \epsilon < 0, \text{ for any } i \in \{1, \dots, N-1\} \tag{3.23}$$

in which u_{s_J} *is a vector containing one or more of the possible* j *indexes which the corresponding product contributes a negative term to* ϵ *and satisfies* $|u_{s_J}(x)| = |k_J u_{\max_J}(x)|$*, then the continuous (or discrete) input-output mapping* $H : L_e^2(U) \to L_e^2(Y)$ *(*$H : l_e^2(U) \to l_e^2(Y)$*) is not passive. If no such reachable waveform (sequence) exists which satisfies (3.22) ((3.23)) the input-output mapping* H *is passive.*

Proof 9 *Assume that (3.22) (or (3.23)) condition exists, and we input a waveform (sequence)* $u(x)$ *such that* $|u_j(x)| \leq k_j u_{\max_j}(x)$ *is always satisfied up to time* $T - \delta t, \ \lim \delta t \to 0$ *(index* $(N-1)-1$*), therefore,*

$$\langle u(t), y(t) \rangle_{T-\delta t} = \beta(T - \delta t) \geq -\beta(0) \tag{3.24}$$

or

$$\langle u(i), y(i) \rangle_{N-1} = \beta(N-1) \geq -\beta(0) \tag{3.25}$$

and the final waveform (sequence) will satisfy (with $\alpha = 1$*)*

$$\langle u(t), y(t) \rangle_T = \alpha\epsilon + \delta t u_{s_{NJ}}^{\mathsf{T}}(T)y_{NJ}(T) + \beta(T - \delta t) \geq -\beta(0) \tag{3.26}$$

or

$$\langle u(i), y(i) \rangle_N = \alpha\epsilon + u_{s_{NJ}}^{\mathsf{T}}(N-1)y_{NJ}(N-1) + \beta(N-1) \geq -\beta(0) \tag{3.27}$$

in which the NJ notation is to account for parts of the vector which are not part of the vectors denoted by J. Therefore, there exists an $\alpha > 1$ such that

$$\alpha > -\frac{(\beta(0) + \beta(T - \delta t) + \delta t u_{s_{NJ}}^{\mathsf{T}}(T) y_{NJ}(T))}{\epsilon} \tag{3.28}$$

or

$$\alpha > -\frac{(\beta(0) + \beta(N-1) + u_{s_{NJ}}^{\mathsf{T}}(N-1) y_{NJ}(N-1))}{\epsilon} \tag{3.29}$$

such that instead we choose $u_J(T) = \alpha k_J^{\mathsf{T}} u_{\max_J}(T)$ ($u_J(N-1) = \alpha k_J^{\mathsf{T}} u_{\max_J}(N-1)$) we will violate (3.26) (or (3.27)) which violates passivity. Likewise if no such sequence exists then passivity is preserved.

Note 6 *It was given as an exercise in [25, Exercise VI-4.7] to show how passivity is lost for $[0, k)$ sector input nonlinearities (Appendix B.2) when the passive plant has the following form $G(u(s)) = \frac{1}{1+qs}$. Theorem 6 tells us that the passive plant with input saturation nonlinearity in* **sector** *$[0, u_{\max}k]$ will no longer be passive (Figure 3.2).*

Note 7 *One example of a passive system which maintains passivity when subject to actuator saturation is a positive semi-definite gain block, $G(u(x)) = Ku(x)$ in which $u^{\mathsf{T}}(x)G(u(x))u(x) = u^{\mathsf{T}}(x)Ku(x) \geq 0,\ \forall u(x)$. Hence by Theorem 6 the net system H is passive.*

3.2.1 Input Saturation Typically Destroys Passivity While Output Saturation Does Not.

As we have seen, actuator input saturation can destroy *passivity* for a large class of *passive* systems. However, when saturation occurs on the output of a *passive* system, *passivity* has been shown to be preserved for *certain cases*.

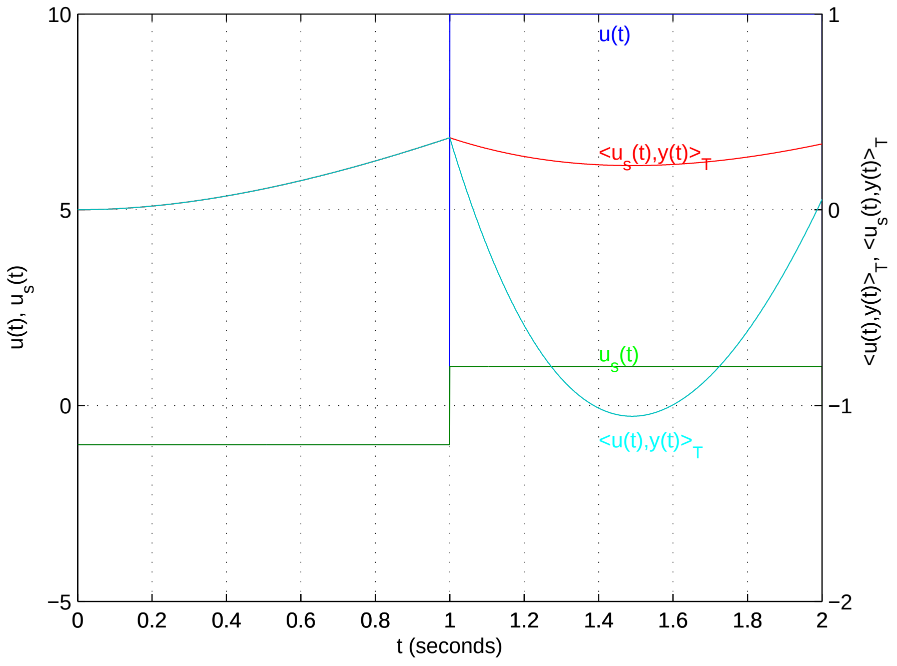

Figure 3.2. Simulation of a first order plant $G(u(s)) = \frac{1}{s+1}$ with input saturation in **sector**[0,1].

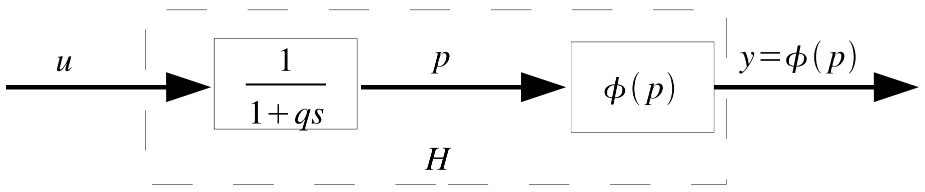

Figure 3.3. First order (*strictly*)-*positive real* system subject to any **sector**$[0,\infty)$ nonlinearity.

Theorem 7 *[25, Example VI.4.4] Consider the system shown in Figure 3.3; the input of H is u and its output is y. We have*

$$q\dot{p}(t) + p(t) = u(t),\ p(0) = p_o \tag{3.30}$$

$$y(t) = \phi[p(t)] \tag{3.31}$$

where $u, p, y : \mathbb{R}_+ \to \mathbb{R}$ and is continuous. If $q \geq 0$ and if $\phi \in$ **sector** *$[0, \infty)$, then H is passive.*

This important theorem provides us the justification to implement an integrator anti-windup block on the output of our controller for example, and still be able to maintain a passive controller. Next we will provide the proof for Theorem 8 which was given as an exercise [25, Exercise VI.4.8].

Theorem 8 *Consider the m-input-m-output generalization of the system shown in Figure 3.3 and denoted as $H : u \mapsto y$. Let $P, Q \in \mathbb{R}^{m\times m};\ p(t), u(t), y(t) \in \mathbb{R}^m$, and*

$$Q\dot{p}(t) + Pp(t) = u(t) \tag{3.32}$$

$$y(t) = \phi[p(t)] \tag{3.33}$$

H is passive if: $\phi : \mathbb{R}^m \to \mathbb{R}^m$, is continuous with

$$\phi^{\mathsf{T}}[p(t)]Pp(t) \geq 0,\ \forall p \in \mathbb{R}^m \tag{3.34}$$

and it is assumed further that

$$\phi(p)^{\mathsf{T}}Q = [\bigtriangledown V(p)]^{\mathsf{T}},\ \forall p \in \mathbb{R}^m \tag{3.35}$$

where $V : \mathbb{R}^m \to \mathbb{R}$ is in C^1 and $V(p) \geq 0,\ \forall p \in \mathbb{R}^m$. Furthermore, for "noninteracting" nonlinearities, i.e., for $i = 1, 2, \ldots, m$, $\phi_i(p) = f_i(p_i)$, and if each f_i is in the **sector** $[0, \infty)$*, then (3.34) and (3.35) are satisfied by $P = Q = I$ in which*

$$V(p) = \sum_{i=1}^{n} \int f_i(p_i) dp_i \tag{3.36}$$

.

Proof 10 *In order to show passivity we solve the inner-product for the combined system $H(u(t))$.*

$$\begin{aligned} \langle H(u(t)), u(t) \rangle_T &= \langle \phi[p(t)], Q\dot{p}(t) + Pp(t) \rangle_T \\ &= \langle \phi[p(t)], Q\dot{p}(t) \rangle_T + \langle \phi[p(t)], Pp(t) \rangle_T \end{aligned} \tag{3.37}$$

Since $\langle \phi[p(t)], Pp(t) \rangle_T \geq 0$ by the assumption given by (3.34) we can simplify (3.37) to be

$$\begin{aligned} \langle H(u(t)), u(t) \rangle_T &\geq \langle \phi[p(t)], Q\dot{p}(t) \rangle_T \\ &\geq \int_0^T \phi[p(t)]^\mathsf{T} Q\dot{p}(t) dt \qquad (3.38) \\ &\geq \int_{p(0)}^{p(T)} [\bigtriangledown V(p)]^\mathsf{T} dp \qquad (3.39) \end{aligned}$$

in which (3.38) results from the assumption given by (3.34) and substituting $dp(t) =$

$\dot{p}(t)dt$. *Finally from the gradient theorem (3.39) is simplified to*

$$\begin{aligned}\langle H(u(t)), u(t)\rangle_T &\geq V(p(T)) - V(p(0)) \\ &\geq -V(p(0)) \qquad (3.40)\end{aligned}$$

which is the definition of passivity and completes the first part of the proof. The solution of (3.36) is a direct result of solving (3.35) and assuming $\phi_i(p) = f_i(p_i)$ in which

$$[\bigtriangledown V(p)]^{\mathsf{T}} = [\frac{dV(p)}{dp_1}, \ldots, \frac{dV(p)}{dp_m}] = [f_1(p_1), \ldots, f_m(p_m)] \qquad (3.41)$$

.

3.2.2 The *Inner-Product Recovery Block.*

Figure 3.4 depicts how we prefer to implement the nonlinear controller $\beta(u(x))$ introduced in [15]. Instead of depicting $\beta(u(x))$ directly before the other controller G_c as depicted in Figure B.1, we locate $\beta(u(x))$ at the output of the plant $G : \sigma(u(x)) \mapsto p(x)$ in order to explicitly look at the map $H_p : u \mapsto y$. As indicated in Figure 3.4, $\beta(u(x))$ recovers the inner-product such that $\langle y(x), u(x)\rangle_X = \langle p(x), \sigma(u(x))\rangle_X$, therefore, we

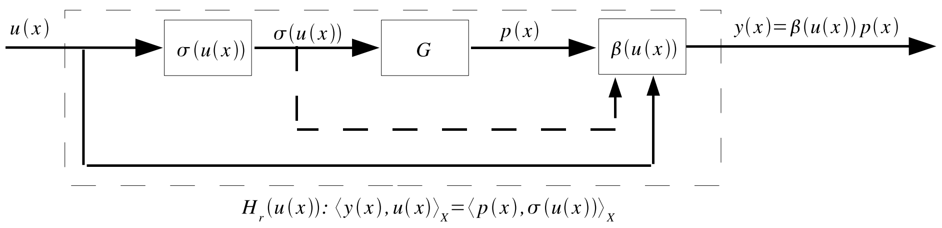
$u(x)$
$\sigma(u(x))$
$\sigma(u(x))$
G
$p(x)$
$\beta(u(x))$
$y(x)=\beta(u(x))p(x)$
$H_r(u(x))$: $\langle y(x), u(x)\rangle_X = \langle p(x), \sigma(u(x))\rangle_X$

Figure 3.4. The nonlinear controller $\beta(u(x))$ recovers the inner-product lost due to the memoryless nonlinearity $\sigma(u(x))$.

shall refer to $\beta(u(x))$ as the *inner-product recovery block.*

Lemma 5 *Given the inner-product recovery block $\beta(u(x))$ described in Appendix B.2 and (B.7), using either an exact model or measurement of the memoryless nonlinearity $\sigma(u(x))$. When using the inner-product recovery block as depicted in Figure 3.4, the following will always be satisfied:*

$$\langle y(x), u(x)\rangle_X = \langle p(x), \sigma(u(x))\rangle_X \tag{3.42}$$

Proof 11 *The proof is straight forward using (B.9) to yield (3.45) and (B.8) to yield (3.46).*

$$\begin{aligned}
\langle y(x), u(x)\rangle_X &= \langle \beta(u(x))p(x), u(x)\rangle_X && (3.43)\\
&= \langle p(x), \beta(u(x))^{\mathsf{T}}u(x)\rangle_X && (3.44)\\
&= \langle p(x), \beta(u(x))u(x)\rangle_X && (3.45)\\
&= \langle p(x), \sigma(u(x))\rangle_X && (3.46)
\end{aligned}$$

Lemma 5 allows us to prove Theorem 9.

Theorem 9 *Given the inner-product recovery block $\beta(u(x))$ described in Appendix B.2 and (B.7), using either an exact model or measurement of the memoryless nonlinearity $\sigma(u(x))$. When using the inner-product recovery block as depicted in Figure 3.4, the following can be said about $H_r : u(x) \mapsto y(x)$ given $G : \sigma(u(x)) \mapsto p(x)$.*

I. If G is passive then H_r is passive.

II. If G is strictly-output passive then H_r is strictly-output passive if $\sigma_{MAX}(\beta(u))^2 < \infty,\ \forall u \in \mathbb{R}^m$ (Definition 17).

III. *If G is strictly-input passive then H_r is strictly-input passive if there exists a $\gamma > 0$ such that $\sigma^{\mathsf{T}}(u)\sigma(u) \geq \gamma u^{\mathsf{T}} u,\ \forall u \in \mathbb{R}^m$.*

Proof 12 *Based on our assumptions, Lemma 5 shows that (3.42) will hold. Furthermore:*

I. *if G is passive then $\langle p(x), \sigma(u(x))\rangle_X \geq -\beta$, substituting (3.42) yields*

$$\langle y(x), u(x)\rangle_X \geq -\beta \tag{3.47}$$

which satisfies Definition 3 for passivity.

II. *if G is strictly-output passive then $\langle p(x), \sigma(u(x))\rangle_X \geq \epsilon\|(p(x))_X\|_2^2 - \beta$, substituting (3.42) yields*

$$\langle y(x), u(x)\rangle_X \geq \epsilon\|(p(x))_X\|_2^2 - \beta \tag{3.48}$$

next, we solve for $\|(y(x))_X\|_2^2$

$$\begin{aligned}\|(y(x))_X\|_2^2 &= \|(\beta(u(x))p(x))_X\|_2^2 \\ &\leq \sigma_{MAX}(\beta(u))^2\|(p(x))_X\|_2^2\end{aligned} \tag{3.49}$$

and substitute (3.49) into (3.48) to yield

$$\langle y(x), u(x)\rangle_X \geq \frac{\epsilon}{\sigma_{MAX}(\beta(u))^2}\|(y(x))_X\|_2^2 - \beta \tag{3.50}$$

which satisfies Definition 3 for H_r to be strictly-output passive as long as $\sigma_{MAX}(\beta(u))^2 < \infty$.

III. *if G is strictly-input passive then $\langle p(x), \sigma(u(x))\rangle_X \geq \delta\|(\sigma(u(x)))_X\|_2^2 - \beta$, substituting (3.42) yields*

$$\langle y(x), u(x)\rangle_X \geq \delta\|(\sigma(u(x)))_X\|_2^2 - \beta \tag{3.51}$$

next, our assumption that $\sigma^{\mathsf{T}}(u)\sigma(u) \geq \gamma u^{\mathsf{T}}u,\ \forall u \in \mathbb{R}^m$ implies that

$$\|(\sigma(u(x)))_X\|_2^2 \geq \gamma\|(u(x))_X\|_2^2 \tag{3.52}$$

and substitute (3.52) into (3.51) to yield

$$\langle y(x), u(x)\rangle_X \geq \delta\gamma\|(u(x))_X\|_2^2 - \beta \tag{3.53}$$

which satisfies Definition 3 for H_r to be strictly-input passive.

Note 8 *Theorem 9 is written from the least restrictive case to the most restrictive case in terms of the memoryless nonlinearities $\sigma(u)$ which can be tolerated. Theorem 9-I can be applied to all types of $\sigma(u)$. Whereas Theorem 9-II can be applied to a slightly smaller class of nonlinearities within $\sigma(u)$ such as those which typically have actuator saturation. They can not include unbounded quadratic nonlinearities, $\sigma(u_i) = u_i^p, p \geq 2,\ p \leq -1$, however, quadratic nonlinearities can occur as long as $\sigma(u)$ is linear or saturates as $u \to \infty$ for $p \geq 1$, or a deadzone or linearity occurs at the origin when $p \leq -1$. Also, near the origin, relay nonlinearities can not be tolerated.*

$$\sigma(u_i) = \begin{cases} \mathtt{sgn}(u_i),\ |u_i| > 0 \\ 0, u_i = 0 \end{cases} \tag{3.54}$$

Finally, Theorem 9-III does not include quadratic nonlinearities at the origin, or satura-

tion nonlinearities. However, all **sector**$[k_1, k_2]$ *nonlinearities (Appendix B.2) in which* $0 < k_1 \leq k_2$ *or* $k_1 \leq k_2 < 0$*, will preserve strictly-input passivity.*

3.2.3 Implementing the *Inner-Product Recovery Block* with the *Inner-Product Equivalent Sample and Hold*

By implementing the continuous time *inner-product recovery block* as depicted in Figure 3.5 we can directly use Theorem 3 and Theorem 9 in order to state the following Corollary:

Corollary 8 *Using the IPESH given in Definition 4 and the continuous time inner-product recovery block as depicted in Figure 3.5, the following relationships can be stated between the continuous one-port plant,* $G : \sigma(u(t)) \mapsto p(t)$*, and the discrete transformed one-port plant,* $H_{rd} : u(i) \mapsto y(i)$*:*

I. If G *is passive then* H_{rd} *is passive for all* $\sigma(u)$*.*

II. If G *is strictly-input passive then* H_{rd} *is strictly-input passive if* $\sigma_{MIN}(\beta(u))^2 > 0,\ \forall u \in \mathbb{R}^m$.

III. If G *is strictly-output passive then* H_{rd} *is strictly-input passive if* $\sigma(u)$ *is a* **sector**$[k_1, k_2]$ *nonlinearity such that* $\sigma_{MAX}(\beta(u))^2 = \max(k_1^2, k_2^2) < \infty$.

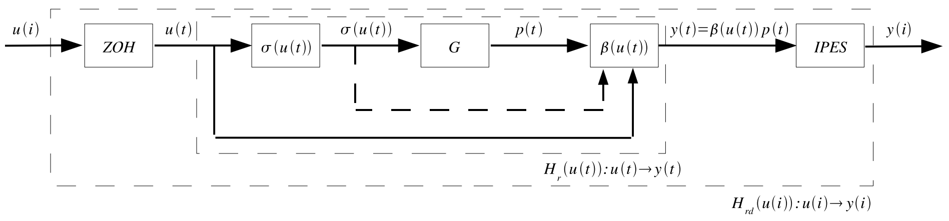

Figure 3.5. Continuous time *inner-product recovery block* used with *IPESH*.

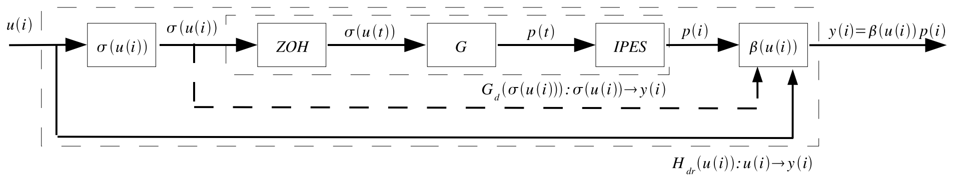

Figure 3.6. Discrete time *inner-product recovery block* used with *IPESH*.

Note 9 *See Appendix B.2 (Theorem 15 and Theorem 16) in order to fully understand why* $\sigma_{MAX}(\beta(u))^2 = \max(k_1^2, k_2^2)$.

Next, we note that by switching the order of the *ZOH* and the memoryless nonlinearity $\sigma(\cdot)$ is mathematically equivalent

$$\sigma(\texttt{ZOH}(u(i))) = \sigma(u(t)) = \texttt{ZOH}(\sigma(u(i))) \tag{3.55}$$

as depicted in Figure 3.6.

Corollary 9 *Using the IPESH given in Definition 4 and the discrete time inner-product recovery block as depicted in Figure 3.6, the following relationships can be stated between the continuous one-port plant,* $G : \sigma(u(t)) \mapsto p(t)$*, and the discrete transformed one-port plant,* $H_{dr} : u(i) \mapsto y(i)$*:*

I. *If* G *is passive then* H_{dr} *is passive for all* $\sigma(u)$.

II. *If* G *is strictly-input passive then* H_{dr} *is strictly-input passive if* $\sigma_{MIN}(\beta(u))^2 > 0, \ \forall u \in \mathbb{R}^m$.

III. *If* G *is strictly-output passive then* H_{dr} *is strictly-input passive if* $\sigma_{MIN}(\beta(u))^2 > 0, \ \forall u \in \mathbb{R}^m$.

IV. If G is LTI and strictly-output passive then H_{dr} is strictly-output passive if the **sector**$[k_1, k_2]$ *nonlinearity is such that $\sigma_{MAX}(\beta(u))^2 = \max(k_1^2, k_2^2) < \infty$.*

3.2.4 l^2-*stable* Digital Control Networks Subject to Memoryless Nonlinearities.

Theorem 10 *The digital control network depicted in Figure 3.7 in which the passive plant G_p is subject to memoryless nonlinearities $\sigma(\cdot)$ is strictly-output passive, which is sufficient for l^2-stability if*

$$\langle f_{op}, e_{doc}\rangle_N \geq \langle e_{oc}, f_{opd}\rangle_N \tag{3.56}$$

holds for all $N \geq 1$.

Proof 13 *Corollary 9-I shows that the mapping from $e_p(i)$ to $f_{op}(i)$ is passive for any memoryless actuator nonlinearity associated with the plant. Since the mapping is now passive, the strictly-output passivity and l^2-stability follows from Theorem 4.*

3.3 Conclusions

We began our discussion with a motivating example, showing how a common memoryless input nonlinearity, actuator saturation:

1. makes it impossible to achieve stability of the absolute mean and second moment for a first order system with a pole strictly in the right half plane (*new* Section 3.1),

2. makes it possible to achieve stability of the absolute mean and second moment for a first order system with a pole not strictly in the right half plane (*new* Section 3.1),

3. eliminates the *passive* mapping for a large class of *passive* systems (*new* Theorem 6).

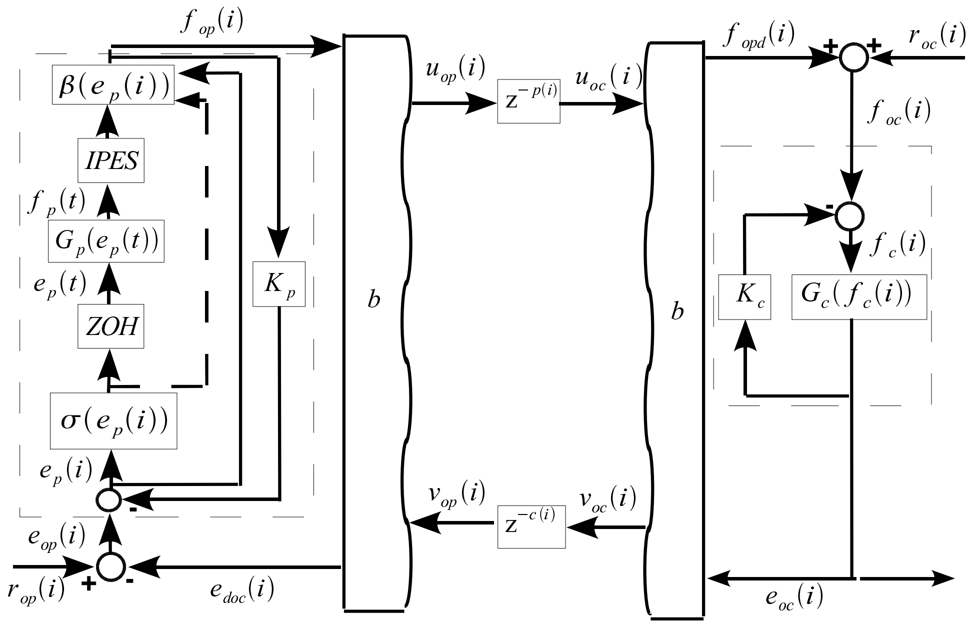

Figure 3.7. l^2-*stable* digital control networks subject to memoryless nonlinearities.

We continued our discussion noting that output saturation and other memoryless **sector** $[0, \infty)$ nonlinearities do not necessarily eliminate a *passive* mapping. Theorem 14 is reviewed as it relates to a first order *LTI strictly-positive real* system. We then solve an interesting exercise [25, Exercise VI.4.8] and provide a corresponding proof for Theorem 15 as it relates to preserving *passivity* for certain *LTI MIMO* systems.

We then presented the following *new* results:

1. Lemma 5 shows how the *IPRB* depicted in Figure 3.4 recovers the inner-product mapping of either a continuous or discrete time system which is lost due to a memoryless nonlinearity,

2. Theorem 9 shows how *passivity* is always recovered with an *IPRB*, however only certain classes of memoryless nonlinearities will allow a *strictly-output passive* or

strictly-input passive mapping to be preserved,

3. Corollary 8 shows how the *IPESH* can be used with a continuous time *IPRB* to preserve *passivity*, and preserve a *strictly-input passive* mapping for certain classes of memoryless nonlinearities, however a nonlinear *strictly-output passive* mapping will reduce to a *strictly-input passive* mapping

4. Corollary 9 is similar to Corollary 8, except that it relates to implementing the *IPRB* in discrete time, and it notes that a *LTI strictly-output passive* mapping can be typically preserved, the key to such a simple realization is (3.55) and the carefully designed system depicted in Figure 3.6,

5. Theorem 10, completes our discussion, showing how a *IPRB* can be used in discrete time to implement a l^2-*stable* digital control network as depicted in Figure 3.7.

CHAPTER 4

WIRELESS DIGITAL CONTROL OF CONTINUOUS PASSIVE PLANTS OVER TOKEN RING NETWORKS

4.1 Introduction

Many papers are concerned with formalizing methods to determine optimum ways to route messages from various sources to desired sinks. In Section 4.1.1 we provide a brief summary of [134] which shows how classical optimal control techniques can be used to determine an optimal static routing policy which minimizes energy consumption. In Section 4.1.2 we discuss some more interesting papers which use feedback to achieve higher network capacities [33, 72]. Section 4.1.3 discusses [77] which provides the ideal channel capacity required to maintain stability of a platoon of d *LTI* agents. Also, [28] is reviewed as it shows how to use feedback to control uncooperative users of networking resources. Unfortunately, most of these analysis are not concerned with attempting to characterize or understand the effects of the delay of the information transmitted in the network. In Section 4.1.4, we review two wireless *MAC* protocols: m-phase time division multiple access (*TDMA*), and *ALOHA*. *TDMA* typically outperforms a random access protocol such as *ALOHA*, however, for correlated flows such as those under a constant bit rate source (*CBR*), *ALOHA* can obtain greater capacity than *TDMA* [138]. Also, in [139] the time varying delay statistics can be computed for *TDMA* and *ALOHA MAC*s subject to various sources such as a *CBR* source. These papers, do not characterize the delay characteristics for round trip communication patterns which will be seen in our *passive* networks. Therefore, we choose to study a simple token passing *MAC* protocol which will allow us to compute the network capacity and delays for our *passive* networks.

In Section 4.2 we determine the delay characteristics of *MAC*s which use token passing by developing the appropriate Markov chains to describe the system. As such, the capacity of our network is calculated from the average number of round trip steps $\tau_m = \{M_m[v]\}$ taken for the delivery of control data which originated from the starting node m. τ_m is calculated using a convenient set of formulas given in [113] and are provided in Appendix C.1 for convenience. Note that a wireless token ring protocol was chosen to control the spacing of vehicles in the Automated Highway System program and the Berkeley Aerobot Project [27]. This protocol is robust and can handle problems such as duplicate tokens, lost stations (nodes), and dynamically adding new stations to the ring. Furthermore, if m is large, smaller rings of stations could be established with an alternating carrier frequency in order to increase network capacity. In Section 4.2 we will focus on the basic analysis of a single ring of m stations. Looking at performance of a single ring is further justified when looking at potential telesurgery applications in which $m = 4$, consisting of a station at the operator, robot and an aerial unmanned autonomous vehicle (the aerial vehicle can be treated as two stations in the ring) [100].

In [56] methods to reduce the data communicated over a network using sample based lossy data reduction (*LDR*) for telemanipulation systems has been investigated. It is concerned with developing *LDR* algorithms which provide good compression while attempting to maintain a high level of *transparency*. They proposed a measure which could be reflective of *transparency* as

$$I = \int_0^t [(f_m - f_{sd})^2 + (e_c - e_m)^2] d\tau \tag{4.1}$$

(Figure 2.1) in which the flows (f_m, f_{sd}) were velocities (v_{hsi}, v_{to}) of the human system interface (*HSI*) and teleoperator (TO) and the efforts (e_c, e_m) were the corresponding forces (F_{to}, F_{hsi}). We are interested in evaluating the performance of our plant G_p controller G_c network with our discrete random characterization of the delay of our ring network. In Section 4.3 we will study further the resulting distortion between a desired position set point $\theta_{set}(i)$ and the resulting position output from the plant $\theta_{act}(i)$ and examine how a *passive* discrete time varying *LDR* algorithm affects this distortion. These results are compared to a *novel* asynchronous control technique, in which the controller is only run when new data is received from the plant. When the asynchronous controller is not run, it will drop the current reference input and its output will reset to 0 after being held at its last computed level for a period equal to that of the sample and hold period of the plant. Typically, when data for the controller is dropped over a *passive* control network, the controller either assumes a 0-input and updates its output accordingly or it attempts to make a prediction of what the correct input should be. Our variable compression scheme is one such scheme which attempts to make a smooth prediction on the input to tolerate delay and data dropouts. Although less steady state error and distortion occurs when using a variable compression scheme over a 0-input prediction scheme, it does not perform as well as using an asynchronous controller. A detailed simulation and discussion is provided in Section 4.3.2.1 with a corresponding Theorem 11 (Section 4.3.2.2) which shows that the asynchronous controller is indeed *strictly-output passive*, as indicated by our simulations.

4.1.1 Network Routing Based on Classic Optimal Control Design

Wu and Cassandras [134] formulated the routing problem of a fixed network topology as an optimal control problem [62]. Their problem formulation yields a single nonlinear programming (*NLP*) problem in order to maximize network lifetime. The problem is stated to determine the optimal routing probability $w_{ij}^*(t)$ for node i routing to node j at time t. This optimal is based on a model which is dependent only on the inter-node spacing $d_{i,j}$ and a basic $\frac{1}{d^\beta}$ signal path loss model to account for increased energy consumption for selecting a node further than your nearest neighbor which is closer to a destination node. A key result is that an optimal policy can be implemented which is not time varying, (i.e. $w_{ij}^*(t) = w_{ij}^*, \forall t$). In order to pose a solvable problem, however, it is assumed that their optimal policy will be independent of the medium access control mechanism chosen (*MAC*) and that inter-node interference will not be affected by their selection of neighbors in order to guarantee a reliable transmission. Both assumptions are unrealistic and as such their results are limited at best. Further more their final solution, is static and the implementation has no feedback mechanism to correct itself.

4.1.2 Cooperative Multicast Routing and Error Correcting Routing With Feedback

Maric and Yates [72] look at cooperative multicast techniques to maximize network lifetime. They present an algorithm which specifies the nodes' order of transmission and power level in order to maximize the network lifetime known as *maximum lifetime accumulative broadcast* (*MLAB*). The key assumptions in their analysis is that there is enough bandwidth W in their network and a sufficient number of orthogonal channels to avoid interference. They build off the fundamental idea that a relay channel utilizing unreliable overheard information is essential to achieving capacity [22]. Each relay node can store messages which can't be decoded on their own; however, the power from these messages can be added in order to successfully decode a message. This yields a set of linear constraints in order to minimize the scheduled transmit power level at each node p_i. They continue to state their problem to minimize the maximum p_i, $\hat{p} = \max_p p_i$, required for a successful broadcast in a network. Making $\hat{p}$ a constraint of p allowed them to pose their problem as a linear program. Using basic set theory and some inductive proofs they determine a minimum power level, p^*, such that an optimum schedule, $\mathbf{x}$, from the set of all possible schedules, $\chi_n(\mathcal{D})$, can be determined such that the set of destination nodes, $\mathcal{D}$, will reliably receive the data transmitted from the source node. Furthermore they were able to provide a distributed algorithm which actually relies on feedback from their neighbors.

The following paper illustrates how to use feedback techniques in order to find the optimal number of redundant packets to send in order to maximize an objective function $J(u_b, n, p)$ [33]. n is the number of packets sent, p_b is the probability a packet will be received over a network at the source and u_b is the number of redundant packets which are sent in order to successfully reconstruct all transmitted data. The packet can be successfully recovered if at least $n - u_b$ blocks are received. Their control input is u_b and their controller should determine u_b^* which maximizes J (4.2).

$$u_b^* = \max_{u_b} \; (n - u_b) \sum_{i=0}^{u_b} \binom{n}{i} p_b^i (1 - p_b)^{(n-i)} \tag{4.2}$$

They continue their discussion indicating that in reality their model has uncertainties and that using feed-forward techniques which determine an appropriate u_b^* will not be robust to correlated losses such as those captured with a two state Markov model. So they design a controller which feeds back the number of packets received per block $y_b = (n - u_b) z_b$. This formulation is unique in that it accounts for when blocks are lost. Their controller is then implemented to find the peak based on the derivatives of y_b and u_b.

$$u_{b+1} = u_b + \beta \text{sgn}(y_b^d u_b^d) \tag{4.3}$$

As can be seen from (4.3) β behaves like a gain for an integrator. They complete their discussion, showing how this feedback has roughly a 10% improvement in packet recovery rate with correlated packet losses as opposed to a control system which just adjusts its u_b as a measure of p_b in a feed-forward manner. Note the results were generated using the freely available *NS-2* simulator in which recent work has been done to simulate the 802.15.4 layer in which the *MAC* layer for the MICAz motes radios comply to [96, 141].

4.1.3 Fundamental Bounds and Controlling Uncooperative Users.

A more theoretical paper [77], which is heavily based on set and geometric theory along with a host of new definitions, attempts to formulate a model of a network of sources (sensors), controllers (network), and sinks (actuators). With this model of a network the paper proves that a platoon of d agents, each communicating his position to his neighbor with channel capacity c_i can stabilize their motion about a given trajectory if and only if (4.4) is met.

$$\log_2 |\det A_0^+| + \cdots + \log_2 |\det A_{d-1}^+| < \min_{i=0,\ldots,d-1} c_i \tag{4.4}$$

Note that A_i^+ is the unstable part of the matrix A_i. Other assumptions are that the controllers have infinite memory, the unstable eigenvalues are real and distinct, and that the channels are perfect. It appears they are achieving maximum capacity by sending a control command to their neighbor in order to compress the state information of the sensor.

The next paper to be discussed involves maintaining cooperative network flows over a network [28]. They build a controller to balance network flows which can be implemented with edge routers over a network using *TCP*. The model is formulated using

Kelly's primal and dual optimization framework for users to maintain an optimal cooperative share of bandwidth [51, 130]. Using Lyapunov's second method they prove they can provide a controller to correct Kelly's controller to adjust the cost q_i it feeds back to an uncooperative user who is not using his assigned utility function $U_i(x_i)$, in which x_i is the users current sending rate. This is simply done by feeding back the difference from the actual network flow x and the network flow from the model $\hat{x}$ and sending the adjusted cost $\hat{q}_i$ back to the potentially uncooperative node (4.5).

$$\tilde{q}_i = q_i - \rho_i(\hat{x} - x) \tag{4.5}$$

4.1.4 Comparing Medium Access Control Algorithms Which Use Time Division Multiple Access and Slotted *ALOHA*.

The following papers have analyzed network induced delays for m-phase time division multiple access (*TDMA*) and slotted *ALOHA* medium access control *MAC*'s [136, 137]. Each *MAC* implementation captures both ends of the design spectrum, *TDMA* is a deterministic protocol in which each mote gets an allocated time slot to transmit data as opposed to *ALOHA* in which each mote independently and randomly accesses the channel with probability $p = p_q p_t$. p_q is the probability the mote has a message in its queue to transmit and p_t is the probability a mote is allowed to transmit a message. The former paper provides analytical and simulated results showing that *TDMA* will provide minimal probability of lost packets p_L with a minimum mean delay μ_o and variance σ_o^2 when delivering messages from a source to a destination node. The latter paper shows that by introducing a constraint to drop a packet based on a bounded delay BD constraint, *ALOHA* will improve it's performance; however, still fail to meet the performance gained by *TDMA*. Another interesting paper provides a nice throughput analysis for sensor networks with Rayleigh fading channels, topologies which have a square, triangle, hexagonal or randomly distributed grid. The random grid is generated from a Poisson point process [69]. Note the Rayleigh model with an *ALOHA* network is such that every node *is connected* in the sense that each node has the probability to *interfere* with the wireless transmission of a source node to a destination node The capacity results of the analysis for *ALOHA* seem to confirm the result which states that if n sensors need to transmit data to one node the rate scales to roughly $\Theta(1/n)$ [71]. We highlight this fact because it guides us to look for creating distributed control systems and cooperative routing algorithms to more efficiently utilize the network and achieve better performance. Another paper worthy of mention in this field is [67]. This paper gives some nice control performance measures related to *TDMA*, Carrier Sensing Multiple Access/Collision Avoidance (*CSMA*), and other random access *MAC* protocols, showing that *TDMA* provided the lowest H_2 norm for control of an inverted pendulum and cart.

4.2 Capacity and Delay of Single Ring Token Networks

A ring network consists of m stations in which each station has a successor (the station it will pass its token to) and a predecessor (the station it will receive a token from). For simplicity we will consider rings in which station i will have the following predecessor, successor pairs $(pr_i, sc_i), \forall i \in \{1, \dots, m\}$:

$$(pr_i, sc_i) = \begin{cases} (i+1, m), & \text{if i=1;} \\ (1, m-1), & \text{if i=m;} \\ (i+1, i-1), & \text{otherwise.} \end{cases} \tag{4.6}$$

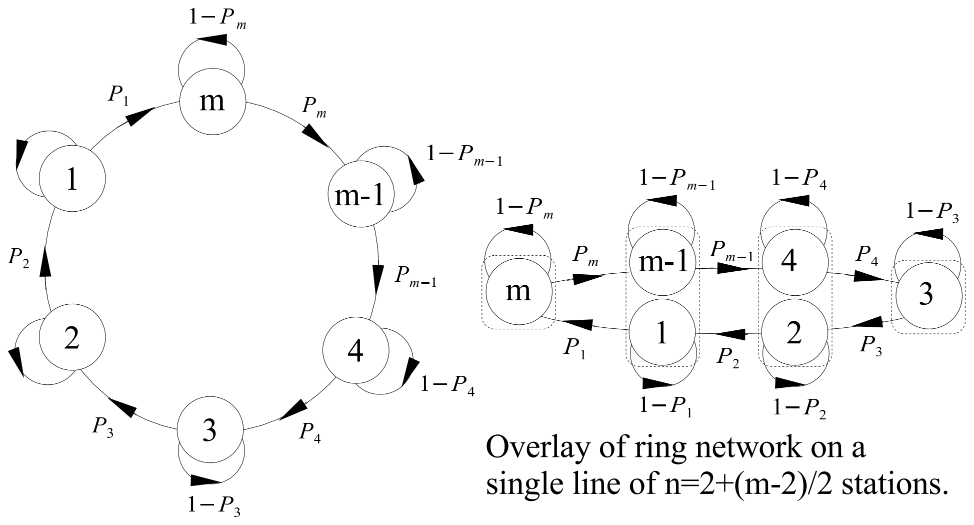

Figure 4.1. Ring networks consisting of m stations and $n = \frac{m-2}{2} + 2$ stations.

Figure 4.1 illustrates a corresponding ring network in which each station has a probability P_i of successfully sending a packet of n_p bits (which typically includes n_d data bits, n_h header bits and n_{fcs} frame check sequence bits) and receiving an acknowledgment of n_{ack} from its successor. For simplicity we assume that the successful transmission of a packet and receiving an acknowledgment is equivalent to passing a token to its successor. The packet dropout rate is denoted as P_{per} which is derived from the bit error rate P_{ber} such that $P_{per} = 1 - (1 - P_{ber})^{n_p}$ (Appendix C.2). Therefore, we conservatively estimate $P_i = (1 - P_{ber})^{n_p + n_{ack}}$. This assumption is conservative, since most people neglect the acknowledgment entirely.

Definition 8 *Let the index $i = \{0, 1, 2, \ldots\}$ denote a packet time slot. Let station m generate a packet of data to transmit around a ring network as depicted in Figure 4.1 at a given rate r such that the next transmission index $i_{next} = i_{current} + r$ in which $i_{current} = i$ when a packet is currently being transmitted from station m (the remaining stations will only relay m's message around the ring network). The minimum ideal rate* *in which station m can generate a packet is $r_{pc} = m$. Therefore, we define the ideal packet capacity as*

$$\lambda_{pc} = \frac{1}{r_{pc}} = \frac{1}{m} \tag{4.7}$$

The average packet capacity λ_p depends on the average round trip time τ_m and has the following form

$$\lambda_p = \frac{1}{\tau_m} \tag{4.8}$$

In order to calculate τ_m we note that the following Markov chain describes a single packet journey in our ring network from station m and back to m. We capture the final round trip packet delivery from station 1 to station m with the corresponding absorbing state 0. The transition matrix P is as follows:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ P_1 & 1-P_1 & 0 & 0 & \dots & 0 \\ 0 & P_2 & 1-P_2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots\dots & & 0 & P_m & P_m - 1 \end{bmatrix} = \begin{bmatrix} I & \mathbf{0} \\ R & Q \end{bmatrix} \tag{4.9}$$

This is in canonical form see Appendix C.1). We can now compute $\mathbf{N}$ in which each $N_{i,j}$ element $i, j \in \{1, \dots, m\}$ is the average number of times each station j is visited if the packet originated in station i before the token reaches the absorbing state 0.

$$\mathbf{N} = (I - Q)^{-1} \tag{4.10}$$

$$\mathbf{N} = \begin{bmatrix} \frac{1}{P_1} & 0 & \ldots\ldots\ldots\ldots & & & 0 \\ \frac{1}{P_1} & \frac{1}{P_2} & 0 & \ldots\ldots & & 0 \\ \frac{1}{P_1} & \frac{1}{P_2} & \frac{1}{P_3} & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \frac{1}{P_1} & \frac{1}{P_2} & \frac{1}{P_3} & \ldots\ldots & & \frac{1}{P_m} \end{bmatrix} \tag{4.11}$$

Using the formulas in Table C.1 the mean arrival time to station m when the initial packet stated at station i is obtained by solving τ:

$$\tau = \begin{bmatrix} \frac{1}{P_1} \\ \frac{1}{P_1} + \frac{1}{P_2} \\ . \\ . \\ \sum_{i=1}^{m} \frac{1}{P_i} \end{bmatrix} \tag{4.12}$$

Or equivalently

$$\tau_i = \begin{cases} \frac{1}{P_i}, i = 1; \\ \tau_{i-1} + \frac{1}{P_i}, 1 < i \leq m. \end{cases} \tag{4.13}$$

Using (C.2) in Appendix C.1 to solve for σ_v^2 results in the following

$$\sigma_{v_i}^2 = 2 \sum_{j=1}^{i} \frac{\tau_j}{P_j} - \tau_i(1 + \tau_i). \tag{4.14}$$

Solving for $\sigma^2_{v_\delta} = (\sigma^2_{v_i} - \sigma^2_{v_{i-1}})$ results in the following:

$$\sigma^2_{v_\delta} = \frac{2\tau_i}{P_i} - \tau_i(1+\tau_i) + \tau_{i-1}(1+\tau_{i-1}) \tag{4.15}$$

$$= \frac{2\tau_i}{P_i} - \tau_i - \tau_i^2 + (\tau_i - \frac{1}{P_i})(1+\tau_i - \frac{1}{P_i}) \tag{4.16}$$

$$= \frac{1-P_i}{P_i^2}. \tag{4.17}$$

Therefore, $\sigma^2_{v_\delta}$ can be equivalently written in the following recursive form

$$\sigma^2_{v_i} = \begin{cases} \frac{1-P_1}{P_1^2}, \text{if } i = 1; \\ \sigma^2_{v_{i-1}} + \frac{1-P_i}{P_i^2}, 1 < i \leq m. \end{cases} \tag{4.18}$$

We now state the following lemma.

Lemma 6 *Given a ring network as described by (4.6) and depicted in Figure 4.1. In which each station $i \in \{1, \ldots, m\}$ has a probability P_i of successfully sending a message to its successor station and $1 - P_i$ of unsuccessfully sending the message per attempt, the following holds:*

i. The average steps it takes to relay a message from station m around the ring network is $\tau_m = \sum_{i=1}^m \frac{1}{P_i}$, for the special case $P_i = p$ then $\tau_m = \frac{m}{p}$.

ii. The corresponding variance in the round trip time is $\sigma^2_{v_m} = \sum_{i=1}^m \frac{1-P_i}{P_i^2}$, for the special case $P_i = p$ then $\sigma^2_{v_m} = \frac{m(1-p)}{p^2}$.

Note 10 *The corresponding packet capacity of the ring network is $\lambda_p = \frac{1}{\tau_m}$, for the special case $P_i = p$ then $\lambda_p = \frac{p}{m}$. The solution of λ_p is a direct substitution of $\tau_m = \frac{m}{p}$ (Lemma 6-i) into (4.8) from Definition 8.*

Proof 14 *Most of the proof has been provided in the subsequent discussion. In summary:*

i. The solution for τ_m is provided by (4.12).

ii. The solution for $\sigma^2_{v_m}$ is easily obtained by our recursive solution given by (4.18).

Note 11 *For the l^2-stable digital control network depicted in Figure 2.6 with $n-2$ equally spaced relay motes, we can create a ring network as depicted in Figure 4.1 in which station m is the plant G_p and station $\frac{m}{2} = n-1$ is the controller G_c. The controller only returns control data if it has received data from the plant. If we neglect the queuing delay associated with a CBR source and assume that the controller can compute a control command and immediately issue a response to the plant and we assume each node is equally likely to successfully transmitting a packet to its corresponding successor, then the average packet transmission delay from G_p to G_c and vice versa is $\frac{m}{2p} = \frac{n-1}{p}$ with a corresponding variance of $\frac{(n-1)(1-p)}{p^2}$.*

Although, the packet capacity is a convenient abstraction for characterizing the network, we still need to quantify the actual rate of data being transmitted. Therefore, we provide the following definition:

Definition 9 *The data capacity of a ring network is the number of actual data bits per second which can be transmitted round trip from a given source node, in which the data is relayed from every station in the network. Therefore, the data capacity in the ring network is*

$$\lambda_d = \frac{f_{bit}\lambda_p n_d}{(n_p + n_{ack})} \tag{4.19}$$

in which λ_p is the packet capacity, n_d is the number of actual data bits which get transmitted for a successful packet delivery, n_p is the number of packet bits, n_{ack} is the number of bits required for the acknowledgment, and f_{bit} is the transmit data rate (bits per second).

For the special case when each node is equally spaced in a line network with n nodes ($m = 2(n-1)$ stations) then the corresponding data capacity is

$$\lambda_d = \frac{f_{bit} n_d p}{m(n_p + n_{ack})} = \frac{f_{bit} n_d p}{m(n_d + n_h + n_{fcs} + n_{ack})} \tag{4.20}$$

in which p is dependent on $n_d,\ n_p,$ and n_{ack} and the node spacing d. Building from the analysis given in Appendix C.2.1 for determining the packet error rate for the CC2420 and using the following parameters given in Table 4.1:

TABLE 4.1

CC2420 PARAMETERS SUMMARY.

Term	Symbol	Value
bits per second	f_{bit}	250×10^3 bps
header bits	n_h	$13 * 8 = 104$ bits
frame check sequence bits	n_{fcs}	$2 * 8 = 16$ bits
ack bits	n_{ack}	$11 * 8 = 88$ bits
data bits	n_d	$0 \leq n_d \leq 960$ bits
typical transmit power	P_T	$-24 \leq P_T \leq 0$ dBm
worst case transmit power	P_T	$-27 \leq P_T \leq -3$ dBm
typical noise figure	NF	15.44 dB
worst case noise figure	NF	20.44 dB

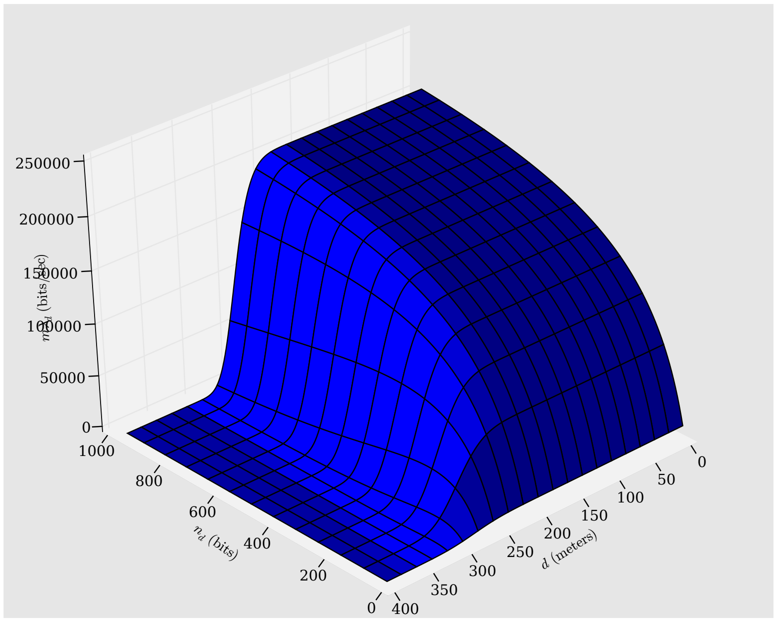

Figure 4.2. $m\lambda_d$ for the CC2420 as a function of d and n_d for a ring network in free space ($n = 2,\ P_T = -3\text{dBm},\ NF = 20.44\text{dB}$).

we can calculate λ_d as a function of d and n_d which is displayed in Figure 4.2 and Figure 4.4.

These are fairly easy estimates to remember and use when attempting to determine the average delays in a ring network. However, there is a way to get an even closer estimate of the corresponding delay in the network. Using techniques similar to those used by [58, 136], we describe a two dimensional Markov chain to track the head of queue *HoQ* delay at the m^{th} station of an outgoing packet which is generated from a *CBR* source of a packet every r^{th} slot and transmitted over a token ring network. The chain will be described by the (d, s) tuple in which $d \in \{-r+1, -r+2, \ldots, D\}$ denotes the delay of the *HoQ* in which D is the maximum delay before the packet will be dropped or compressed and $s \in \{1, 2, \ldots, m\}$ is the station where the token is in the network.

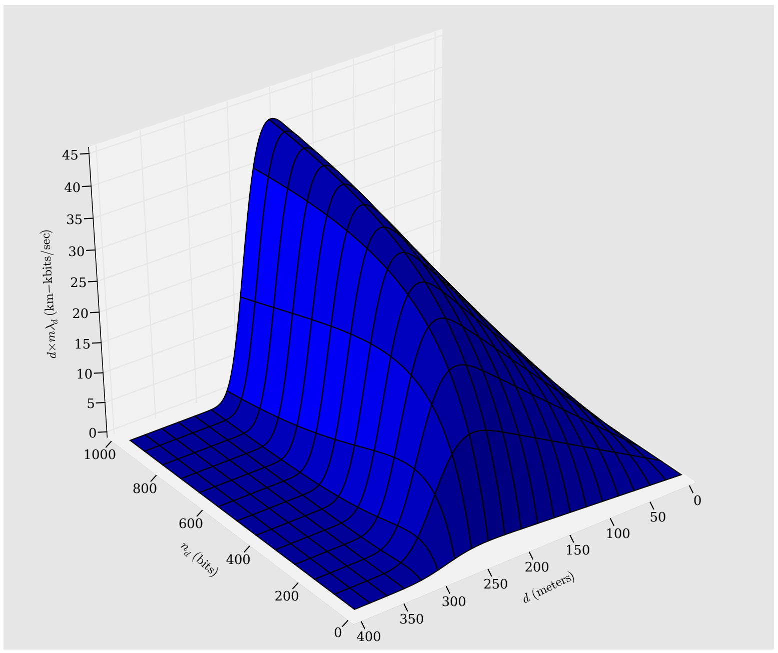

Figure 4.3. $d \times m \times \lambda_d$ for the CC2420 as a function of d and n_d for a ring network in free space ($n = 2,\ P_T = -3$ dBm, $NF = 20.44$dB).

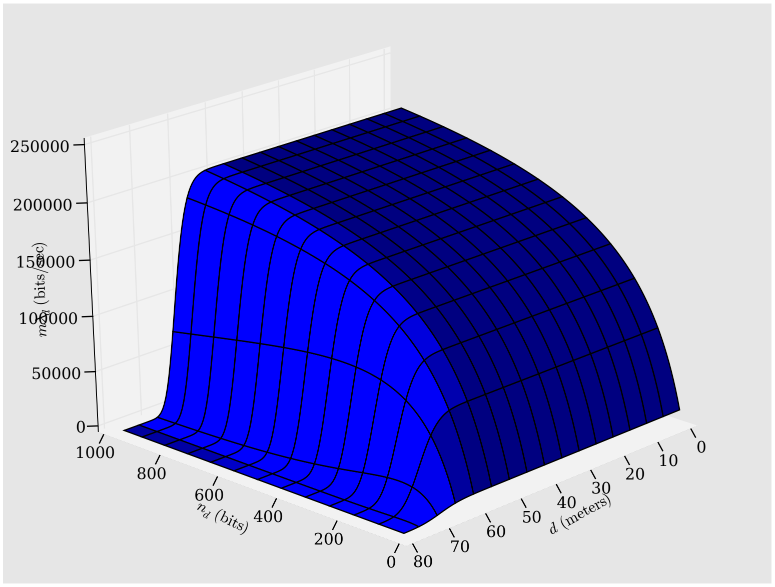

Figure 4.4. $m\lambda_d$ for the CC2420 as a function of d and n_d for a ring network not in free space ($n = 3.3,\ d_o = 8$ meters, $P_T = -3$ dBm$,\ NF = 20.44$dB).

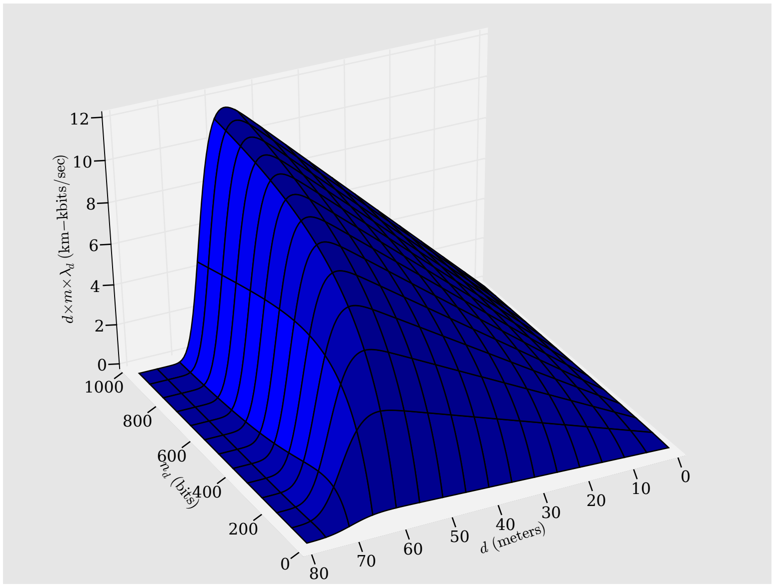

Figure 4.5. $d \times m \times \lambda_d$ for the CC2420 as a function of d and n_d for a ring network not in free space
($n = 3.3,\ d_o = 8$ meters, $P_T = -3$ dBm$,\ NF = 20.44$dB).

Lemma 7 *Given a ring network as described by (4.6) and depicted in Figure 4.1 and assuming that a packet of n_p bits will be rotated through the network (even if no new data is present). In which each station $i \in \{1, \dots, m\}$ has a probability P_i of successfully sending a message to its successor station and $1 - P_i$ of unsuccessfully sending the message per attempt, the following transition matrix $P \in \mathbb{R}^{((D+r)m)\times((D+r)m)}$ describes the HoQ delay:*

$$P = \begin{array}{c} (-\mathbf{r}+\mathbf{1},\mathbf{s}) \\ (-\mathbf{r}+\mathbf{2},\mathbf{s}) \\ \dots \\ (-\mathbf{1},\mathbf{s}) \\ (\mathbf{0},\mathbf{s}) \\ (\mathbf{1},\mathbf{s}) \\ \dots \\ (\mathbf{D}-\mathbf{1},\mathbf{s}) \\ (\mathbf{D},\mathbf{s}) \end{array} \left[\begin{array}{ccccccccc} \mathbf{0} & \mathbf{P_r} & \mathbf{0} & \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots\dots} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P_r} & \mathbf{0} & \multicolumn{4}{c}{\dots\dots\dots\dots\dots\dots} & \mathbf{0} \\ \multicolumn{9}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{P_r} & \mathbf{0} & \multicolumn{2}{c}{\dots\dots\dots\dots} & \mathbf{0} \\ \mathbf{P_m} & \mathbf{0} & \dots\dots & \mathbf{0} & \mathbf{P_r}-\mathbf{P_m} & \mathbf{0} & \multicolumn{2}{c}{\dots\dots} & \mathbf{0} \\ \mathbf{0} & \mathbf{P_m} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{P_r}-\mathbf{P_m} & \mathbf{0} \; . & \mathbf{0} \\ \multicolumn{9}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ \mathbf{0} & \dots\dots & \mathbf{0} & \mathbf{P_m} & \mathbf{0} & \multicolumn{2}{c}{\dots\dots\dots} & \mathbf{0} & \mathbf{P_r}-\mathbf{P_m} \\ \mathbf{0} & \multicolumn{2}{c}{\dots\dots\dots} & \mathbf{0} & \mathbf{P_r} & \mathbf{0} & \multicolumn{2}{c}{\dots\dots} & \mathbf{0} \end{array}\right] \tag{4.21}$$

in which $\mathbf{P_r} \in \mathbb{R}^{m\times m}$ is the transition matrix which describes the evolution of the token,

$$\mathbf{P_r} = \begin{bmatrix} 1-P_1 & 0 & \multicolumn{3}{c}{\dots\dots\dots\dots} & 0 & P_1 \\ P_2 & 1-P_2 & 0 & \multicolumn{3}{c}{\dots\dots\dots\dots} & 0 \\ 0 & P_3 & 1-P_3 & 0 & \multicolumn{2}{c}{\dots\dots\dots} & 0 \\ \multicolumn{7}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ 0 & \multicolumn{2}{c}{\dots\dots\dots} & 0 & P_{m-1} & 1-P_{m-1} & 0 \\ 0 & \multicolumn{3}{c}{\dots\dots\dots\dots} & 0 & P_m & 1-P_m \end{bmatrix} \tag{4.22}$$

and $\mathbf{P_m} \in \mathbb{R}^{m\times m}$ is the part of the transition matrix of actually shrinking the HoQ *delay from d to $d-r+1$ when the token is at station m*

$$\mathbf{P_m} = \begin{bmatrix} 0 & 0 & & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \cdots & 0 \\ 0 & & 0 & P_m & 0 \end{bmatrix}. \tag{4.23}$$

The $(\mathbf{d}, \mathbf{s})$ is a short hand to show how the states of the chain correspond to the rows of the transition matrix P in which d is a fixed column of integers and s would have each row correspond to the next state in the chain describing the evolution of the token i.e.

$$(\mathbf{d}, \mathbf{s}) = \begin{matrix} (d,1) \\ (d,2) \\ (d,3) \\ (\cdot,\cdot) \\ (d,m-1) \\ (d,m) \end{matrix} . \tag{4.24}$$

Proof 15 *Creating the transition matrix is fairly straight forward:*

1. $(-r+1) \leq d < 0, s \in \{1, \ldots, m\}$*: we track the evolving state of the token s, this is done using $\mathbf{P_r}$, since the delay will increase by 1 after a transition from any state s, we simply offset the matrix $\mathbf{P_r}$ by $m(d+1)$ columns and place zeros elsewhere.*

2. $0 \leq d < D, s \in \{1, \ldots, m\}$*: once the delay d from the HoQ is ≥ 0 a packet is available for transmission at station m, hence if station m has the token and successfully transmits to station 1 with probability P_m then the delay d will be reduced such that*

$d = d - (r-1)$. This outcome is captured by matrix $\mathbf{P_m}$ and is offset by $d(m+1)$ columns. All other transitions will occur and the delay d will increase by one, hence $\mathbf{P_r} - \mathbf{P_m}$ is correspondingly offset by $m(d+1)$ columns with zeros elsewhere.

3. *$d = D, s \in \{1, \ldots, m\}$: once the delay $d = D$ the packet will either be successfully delivered or dropped, therefore, the delay will shrink such that $d = d - (r-1)$ after the transition. Thus, we only need to track the state of the token s with $\mathbf{P_r}$ which will correspondingly be offset by $d(m+1)$ columns with zeros elsewhere.*

Note 12 *To calculate the average delay of the HoQ we simply solve for the steady state distribution of our Markov chain:*

$$\pi P = \pi \tag{4.25}$$

in which $\pi = (\pi_{(-r+1,1)}, \pi_{(-r+1,2)}, \ldots, \pi_{(-r+1,m)}, \pi_{(-r+2,1)}, \ldots, \pi_{(D,m)})$ is a row vector. The delay distribution, $d_i(0, \leq d \leq D)$ is given by

$$d_i = \frac{P_m \pi_{(i,m)}}{\sum_{k=0}^{D} P_m \pi_{(k,m)}} = \frac{\pi_{(i,m)}}{\sum_{k=0}^{D} \pi_{(k,m)}} \tag{4.26}$$

where $\sum_{k=0}^{D} P_m \pi_{(k,m)}$ is the normalizing constant, in which we only consider successful transmissions with probability P_m from station m to station 1 in deriving the delay distribution.

Note 13 *Calculating the packet loss probability p_o is simply*

$$\begin{aligned} p_o &= \frac{(1-P_m)\pi_{(D,m)} + \sum_{k=1}^{m-1} \pi_{(D,k)}}{\lambda} \\ &= r[(1-P_m)\pi_{(D,m)} + \sum_{k=1}^{m-1} \pi_{(D,k)}] \end{aligned} \tag{4.27}$$

in which $\lambda = \frac{1}{r}$, if we are in any other state besides m the packet will be dropped, when *in state m there is only a probability of $(1 - P_m)$ of dropping the packet and that is accounted for.*

Note 14 *For the l^2-stable digital control network depicted in Figure 2.6 with $n - 2$ equally spaced relay motes, we can create a ring network as depicted in Figure 4.1 in which station m is the plant G_p and station $\frac{m}{2} = n - 1$ is the controller G_c. The controller only returns control data if it has received data from the plant. If we consider the queuing delay associated with a CBR source at m and consider that the controller will not immediately compute a control command but pass along computations from previous transmissions. Then we can more closely approximate the delay of the delivery of data from the controller to the plant as if it was provided a CBR source r as well. This average delay from G_p to G_c and vice versa can be computed as follows:*

$$\tau_{pc} = \sum_{i=1}^{D} i d_{p_i} + \sum_{i=m-1}^{\frac{m}{2}-1} \frac{1}{P_i} \text{ for the delay from } G_p \text{ to } G_c, \tag{4.28}$$

$$\tau_{cp} = \sum_{i=1}^{D} i d_{c_i} + \sum_{i=\frac{m}{2}}^{m-1} \frac{1}{P_i} \text{ for the delay from } G_c \text{ to } G_p \tag{4.29}$$

in which d_{x_i} is the corresponding delay distribution for either the plant or controller if they were supplied a CBR source r. If $P_i = p$ then $\tau_{pc} = \tau_{cp}$ such that

$$\tau_{pc} = \tau_{cp} = \sum_{i=1}^{D} i d_i + \frac{(m-2)}{2p} \tag{4.30}$$

and the corresponding variance is

$$\sigma_v^2 = \sum_{i=1}^{D} i^2 d_i - (\sum_{i=1}^{D} i d_i)^2 + \frac{(n-2)(1-p)}{p^2}. \tag{4.31}$$

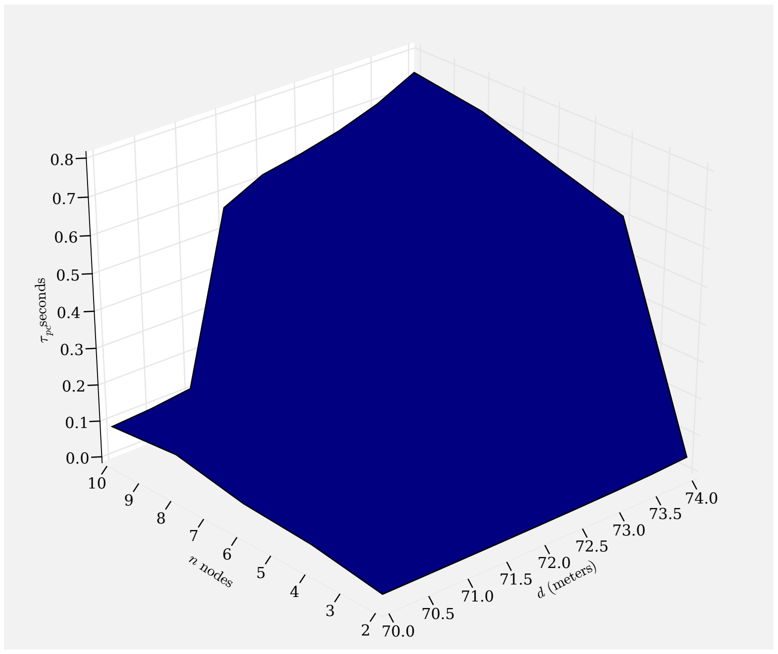

Figure 4.6. Theoretical mean delay of a data packet which contains 2 samples of u_{oc} in which each sample has 40 bits ($n_p = 288$ bits, $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, $r = 86$, $D = 516$) calculated using (4.30).

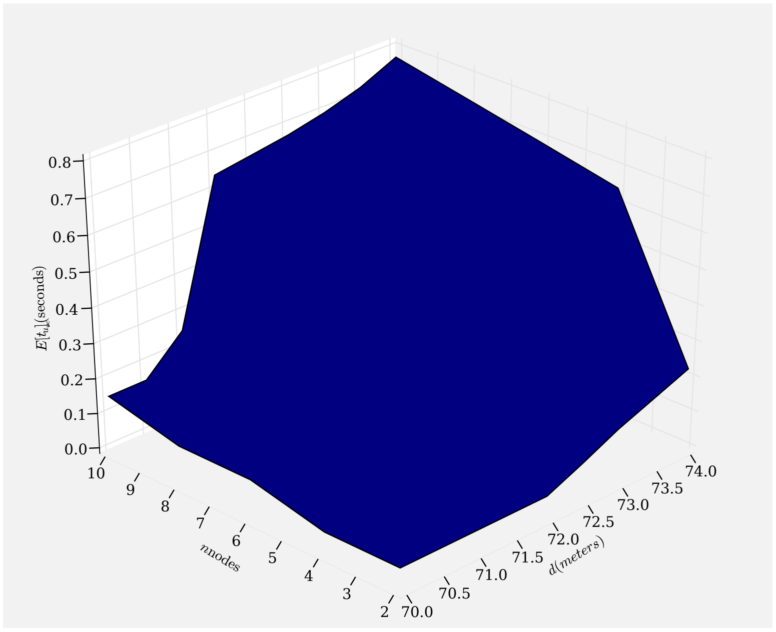

Figure 4.7. Simulated mean delay of u_{oc} in which up to 2 40 bit samples of u_{oc} can be transmitted (maximum $n_p = 288$ bits, $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, maximum $r = 86$, $D = 516$).

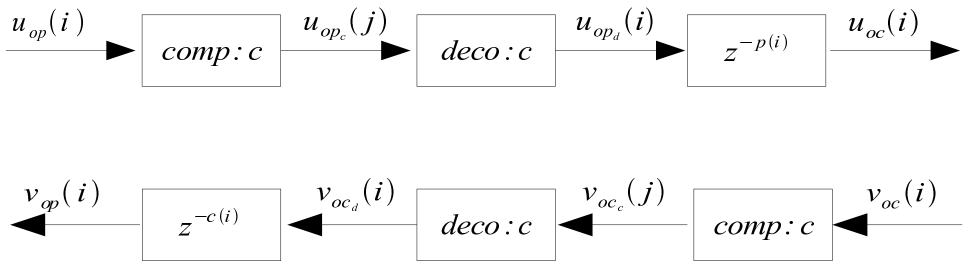

Figure 4.8. Transmission of $u_{op}(i)$ and $v_{oc}(i)$ with compression $comp:c$ and decompression $deco:c$ blocks.

4.3 Distortion in Single Ring Token Networks

We have characterized the delay in Section 4.2, however, in order to account for the time varying queuing delay we assume that when the delay exceeds D slots that the packet should be dropped. However, dropping packets leads to drift in the actual position of the plant $\theta_{act}(i)$ when it is controlled using velocity feedback. Instead of dropping the data when it reaches the maximum delay D we propose the following *LDR* algorithm.

4.3.1 *LDR* Algorithm to Compress Data Which Exceeds a Delay D Before Transmission

We propose an adaptive compression scheme which is similar to the *compressor-expander* scheme used in [11] [10]. The main contribution is we formalize how to further compress samples which have already been compressed. We also introduce a distortion measure to evaluate the performance of proposed compression schemes. A final contribution, is that our simulations reveal similar performance and we discover that by simply dropping data when the FIFO is full, not running the controller when data is not present in the controllers receive FIFO, and by using no compression scheme we actually can reduce the distortion. Assume for simplicity we are sampling data at a rate equal to the time it takes to transmit a packet to another station in a ring network. After r samples are taken we place them in a packet to be transmitted over our network. We note that the most recent sample has a delay 0 and the oldest sample has a delay r. When the delay of the outgoing packet reaches D the oldest sample of data will have a delay of $r + D$. Instead of dropping this sample we propose to compress the data in the queue. The reason is two fold, the first is that by dropping data, drift in the desired position will occur, second, this unusually long delay indicates that we are incurring one of our rare events of multiple failed transmission attempts or possibly the channel has changed and the probability of successful delivery P_i has dropped at various stations. By compressing the data and decompressing the data *adaptively* we should be able to adapt to the randomly changing channel. Unfortunately this will cost us a few extra bits to denote the compression ratio of the given sample, however, it will provide a larger range of operation and hopefully reduced distortion. Assume that the outgoing packet is either $u_{op}(i)$ from the plant or $v_{oc}(i)$ from the controller. Since no duplicate packets will be sent or received in error we can neglect tracking the corresponding sampled index in order to preserve *passivity*. Each sample which has not been transmitted will be stored in a FIFO with a corresponding compression ratio c. The top of the FIFO will be denoted by the index $D+r-1$, the bottom of the FIFO will be denoted by the index 0 and the tail of the FIFO (t_f) will point to the next available slot for data. We will denote accesses from the FIFO with brackets and incoming data at time index i with parentheses. The FIFO and the data within it are handled as follows:

1. Initialize $t_f = D + r - 1$

2. If a new sample is received,

   (a) and if $t_f > -1$ place $u_{op}[t_f] = u_{op}(i)$ and $c[t_f] = 1$, update the tail $t_f =$

$t_f - 1$

(b) else set $t_{fp} = t_f + 1$ and $c_{max} = c[t_{fp}]$,

while(($t_{fp} < D + r - 1$) and ($c_{max} == c[++t_{fp}]$)){};

(done while) $p_h = t_{fp} - 1$.

i. if $p_h == 0$ then

$$u_{op_c} = \sqrt{\frac{c[p_h](u_{op_c}[p_h])^2 + (u_{op}(i))^2}{1 + c[p_h]}},\ c = 1 + c[p_h] \tag{4.32}$$

$$u_{op_c} = u_{op_c}\mathtt{sgn}(c[p_h]u_{op_c}[p_h] + u_{op}(i)) \tag{4.33}$$

$u_{op_c}[0] = u_{op_c}$, $c[0] = c$.

ii. else set $p_l = p_h - 1$ and

$$u_{op_c} = \sqrt{\frac{c[p_h](u_{op_c}[p_h])^2 + c[p_l](u_{op_c}[p_l])^2}{c[p_l] + c[p_h]}},\ c = c[p_l] + c[p_h] \tag{4.34}$$

$$u_{op_c} = u_{op_c}\mathtt{sgn}(c[p_h]u_{op_c}[p_h] + c[p_l]u_{op_c}[p_l]) \tag{4.35}$$

```
$u_{op_c}[p_h] = u_{op_c}$, $c[p_h] = c$,
for ($p = p_h - 1$;$p > 0$;$p--$)
    $u_{op_c}[p] = u_{op_c}[p-1]$; $c[p] = c[p-1]$;
$u_{op_c}[0] = u_{op}(i)$, $c[0] = 1$.
```

3. If a packet is ready for transmitting pop r samples off of FIFO and place in the outgoing send queue, shift all element up to the head of the FIFO and set $t_f = t_f - r$.

4. When a packet of data is received it is pushed onto the receive FIFO with oldest data pushed in first and decompressed as follows:

```
c=0,i=0
while(True)
    if($c == 0$ and data in FIFO)
        $c$=pop($c_{fifo}$);$u_{oc}(i)$=pop($u_{fifo}$);$i++$;$c--$
    else if($c$ not $== 0$)
        $u_{oc}(i) = u_{oc}(i-1)$;$i++$;$c--$
    else
        wait for new data in FIFO
```

Lemma 8 *The proposed compression and transmission scheme is passive.*

Proof 16 *As long as (2.44) holds for all N then our compression and transmission scheme will remain passive. Figure 4.8 illustrates how a compression scheme can be evaluated and still account for time varying delays in the transmission of the compressed data. We denote the sum of $c[k]$ from the tail up to the head of the queue $s_c[j] = \sum_{k=t_f+1}^{j} c[k],\ j \in \{t_f+1, \dots, D+r-1\}$. We note that the proposed compression algorithm satisfies the following inequality:*

$$u_{op_c}^2[j] = \frac{1}{c[j]} \sum_{k=i-s_c[j]-1}^{i-s_c[j]+c[j]-2} u_{op}^2(k) \tag{4.36}$$

in which i is incremented with each sample that is pushed in to the FIFO. If no data has been transmitted, or the receive FIFO is empty then $u_{op_d}(i) = 0$, therefore:

$$\sum_{i=0}^{N-1} u_{op_d}^2(i) \leq \sum_{i=0}^{N-1} u_{op}^2(i) \tag{4.37}$$

Since the transmission of data over the network will not produce duplicate transmis-

sions the time varying delay $p(i)$ is such that $\sum_{i=0}^{N-1} u_{oc}^2(i) \leq \sum_{i=0}^{N-1} u_{op_d}^2(i)$, therefore

$$\sum_{i=0}^{N-1} u_{oc}^2(i) \leq \sum_{i=0}^{N-1} u_{op}^2(i) \tag{4.38}$$

which satisfies (2.44) for passivity.

4.3.2 Evaluating the *LDR* Algorithm by Measuring Distortion.

Definition 10 *For the l^2-stable digital control network depicted in Figure 2.6 in which the flow output is denoted as $f_p(t)$. Denote the sampled integrated output of the plant as $\theta_{act}(i)$, assume that the user will provide a desired set point $\theta_{set}(i)$ to the input of a discrete second order trajectory generator which is a zero-order hold equivalent of $H_t(s)$ as described by (2.84). The mean squared distortion is*

$$I_\theta = \frac{1}{T} E\{\sum_{i=0}^{T} (\theta_{set}(i) - \theta_{act}(i))^{\mathsf{T}} (\theta_{set}(i) - \theta_{act}(i))\} \tag{4.39}$$

in which $E[\cdot]$ denotes the expectation of the summation of the squared error which is dependent on the set point, the controller, the plant dynamics, and the time varying delays incurred due to the communication network.

For the example given in Section 2.4, we estimated the corresponding distortion by averaging the summation given in (4.39) over 50 trials for a given number of n nodes and spacing of d meters. We chose $\theta_{set}(i)$ to use a square wave profile which is 0.0 radians for $0 \leq t < 8.0$ seconds, 1.0 radians for $8 \leq t < 16.0$ seconds, -1.0 radians for $16.0 \leq t < 24.0$ seconds, 0 radians for $24.0 \leq t$ seconds (Figure 4.9). Since we chose a modest sampling rate of $.05$ seconds, and chose to use the variable compression scheme previously discussed we generate data at $5 * 8/.05 = 800.0$ bits per second. Which is a small fraction of the maximum data rate that can be achieved

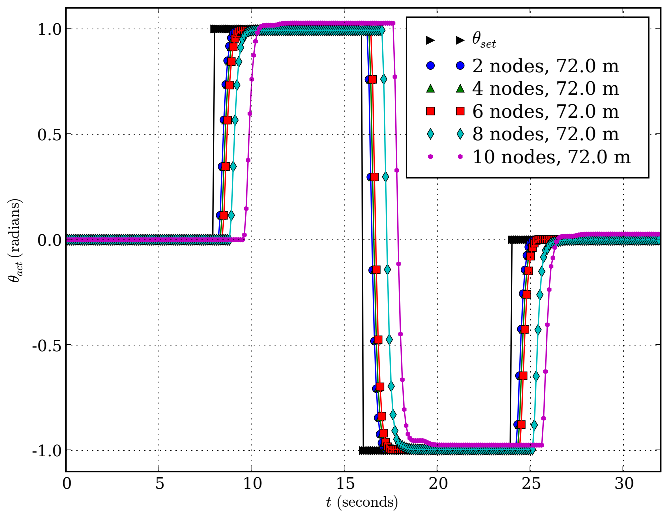

Figure 4.9. Typical response to distortion profile for token network using adaptive compression in which $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 2 samples will be transmitted if available in a single packet.

between a small number of nodes. As such, the delay between receiving data from the plant and controller is roughly the sampling rate ($.05$ seconds) for distances less than 70 meters. Furthermore, since the arrival delay is driven by the low sampling rate, the variance of the delay is extremely small. However, as the distance exceeds 70 meters, the capacity drop off is extremely sharp if we allow up to 12 samples to be transmitted per packet, such that the average delays are around 14 seconds when $d = 72.25$ meters and $n = 10$ nodes (Figures 4.10,4.11). Figures 4.10,4.12,4.11, and 4.13 provide the corresponding mean and variance of the delays as a function of inter node distance d and number of nodes n. On the other hand if we still store up to 12 samples but only transmit up to 2 samples per attempt, the increase in the mean delay and the corresponding variance is significantly reduced for the same range of nodes and transmission distance (Figures 4.14,4.16,4.15, and 4.17). In spite of the significant random delays being incurred with increase in nodes n and node spacing d, the overall distortion I_θ degrades fairly gradually as is seen in Figure 4.18 (note that $I_\theta = 0.5$ corresponds to keeping $\theta_{act} = 0$).

4.3.2.1 Comparing *LDR* Algorithm With Dropping Data in a Full FIFO.

As we have seen, the *LDR* algorithm behaves exceptionally well for extremely unreliable communication channels. When the nodes are spaced 72.0 meters apart, the probability of a successful packet transmission is $P = (1.0 - .00631781)^{8\times(11+2+13+2\times5)} = 16\%$. Which corresponds to a capacity of 5596 bits/second for two nodes, and 622 bits/second for ten nodes ($m = 18$). Thus, to maintain an effective 800 bit/second data rate for ten nodes the data needs a compression ratio of $800/622$ and negligible distortion is seen. However, as the compression ratio increases so does the distortion in Figure 4.18, which is seen in the delay and resulting steady state errors in

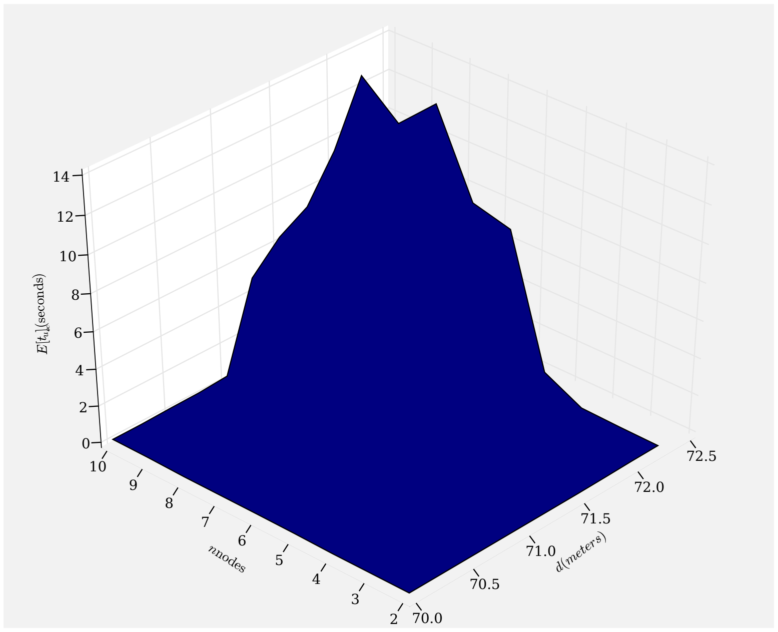

Figure 4.10. Mean delay of u_{oc} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 12 samples will be transmitted if available in a single packet.

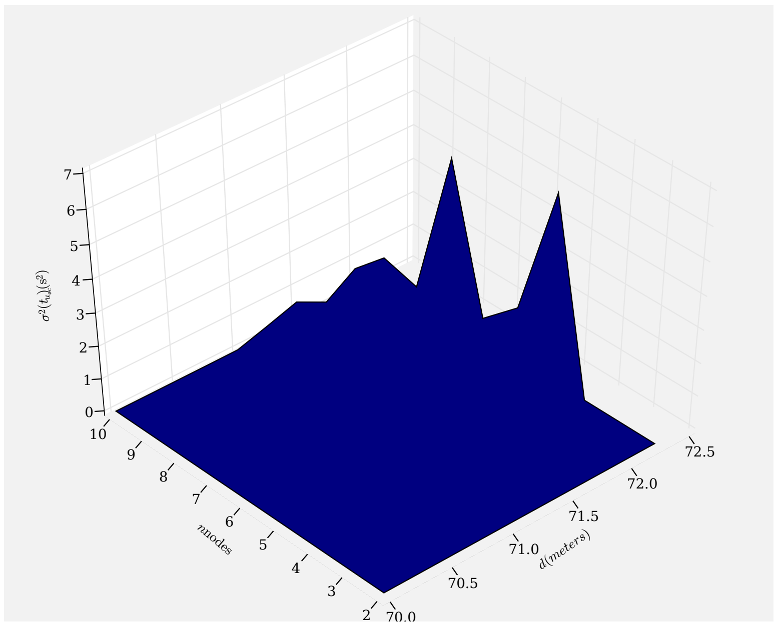

Figure 4.11. Variance of the mean delay of u_{oc} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 12 samples will be transmitted if available in a single packet.

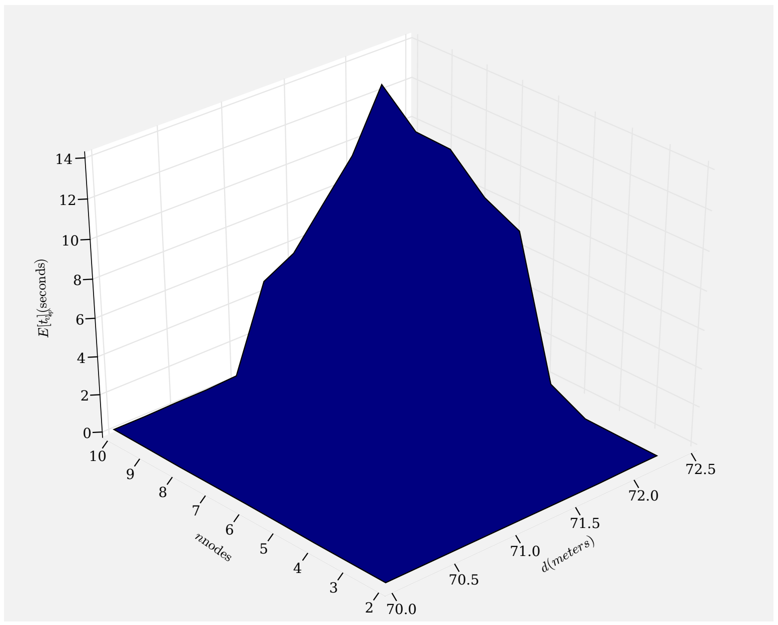

Figure 4.12. Mean delay of v_{op} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 12 samples will be transmitted if available in a single packet.

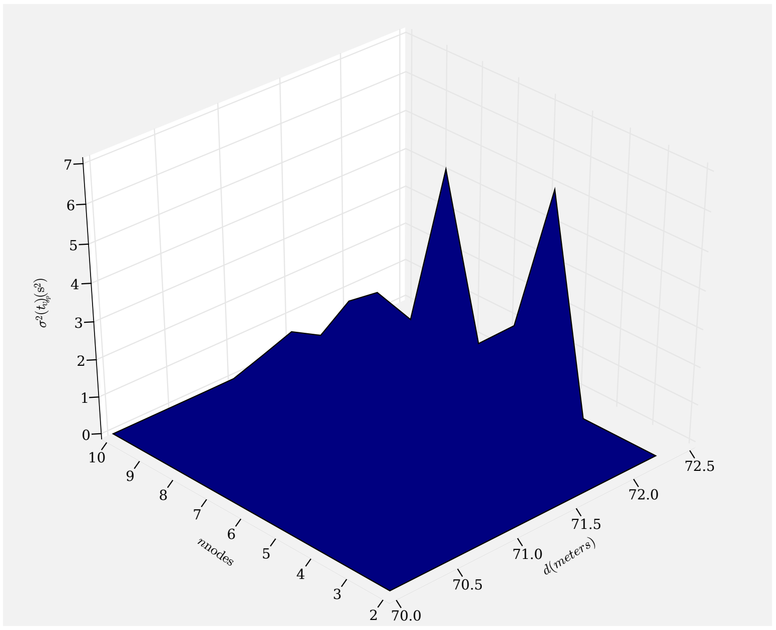

Figure 4.13. Variance of the mean delay of v_{op} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 12 samples will be transmitted if available in a single packet.

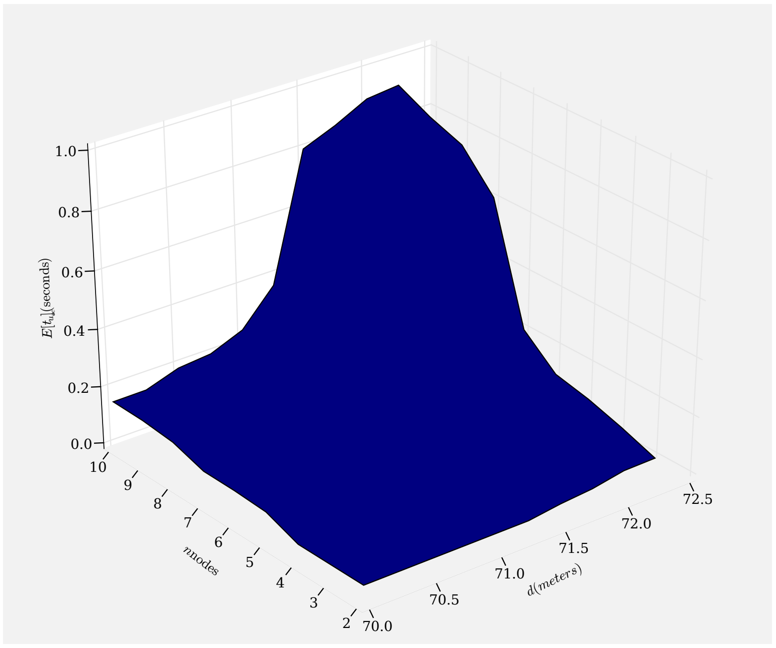

Figure 4.14. Mean delay of u_{oc} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 2 samples will be transmitted if available in a single packet.

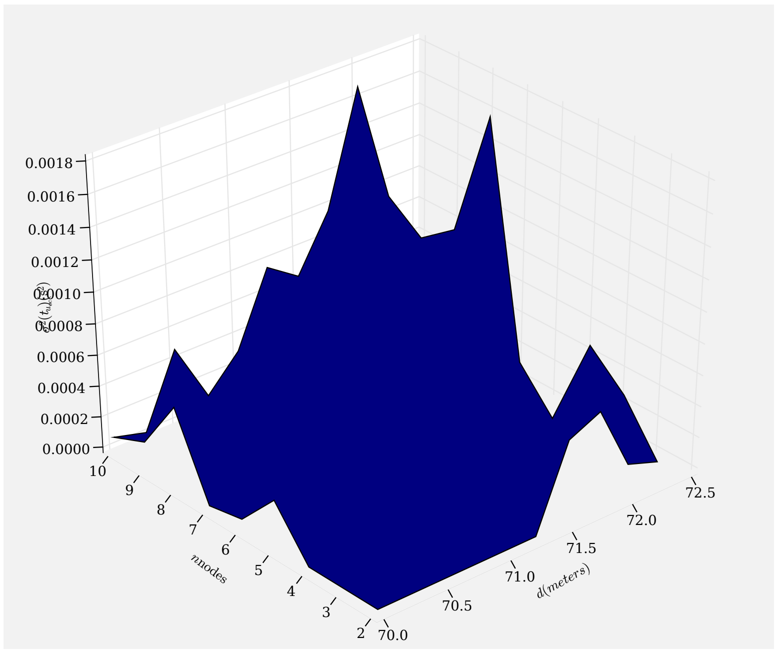

Figure 4.15. Variance of the mean delay of u_{oc} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 2 samples will be transmitted if available in a single packet.

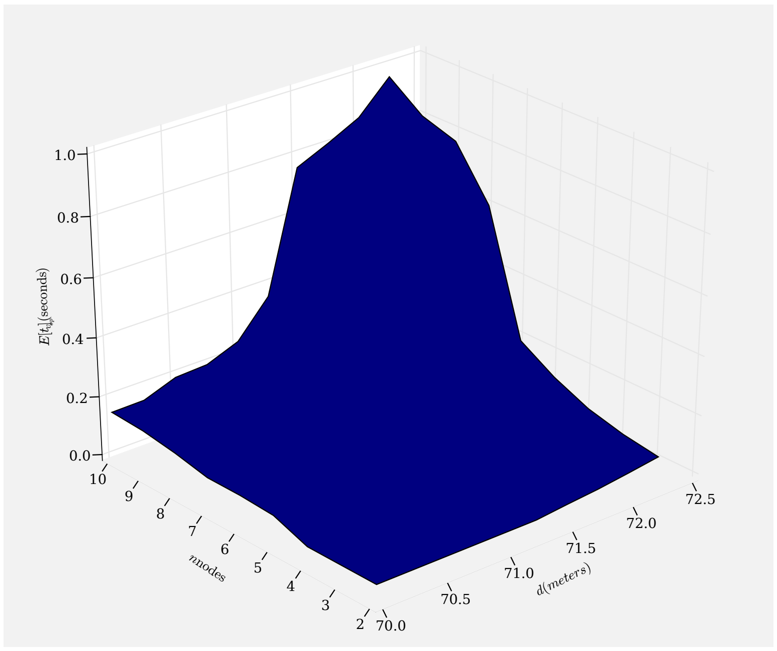

Figure 4.16. Mean delay of v_{op} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 2 samples will be transmitted if available in a single packet.

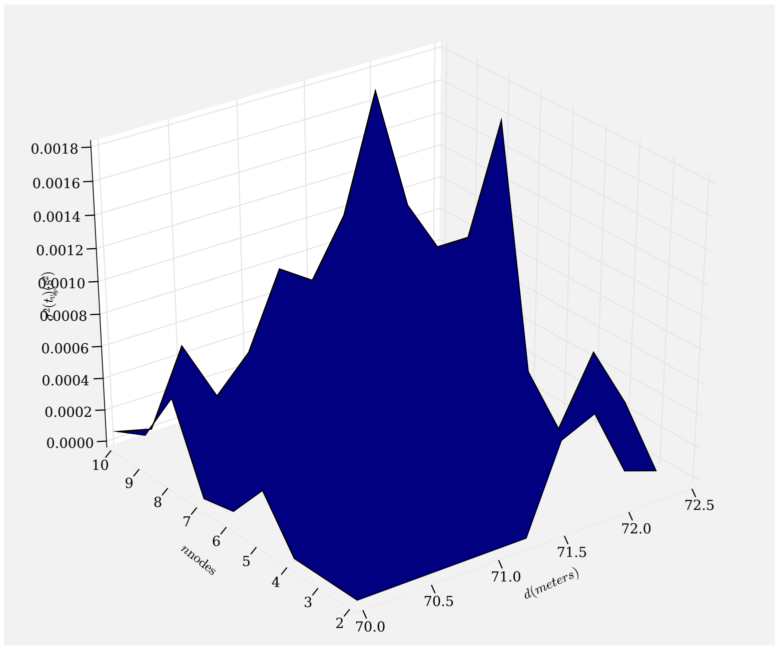

Figure 4.17. Variance of the mean delay of v_{op} assuming $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 2 samples will be transmitted if available in a single packet.

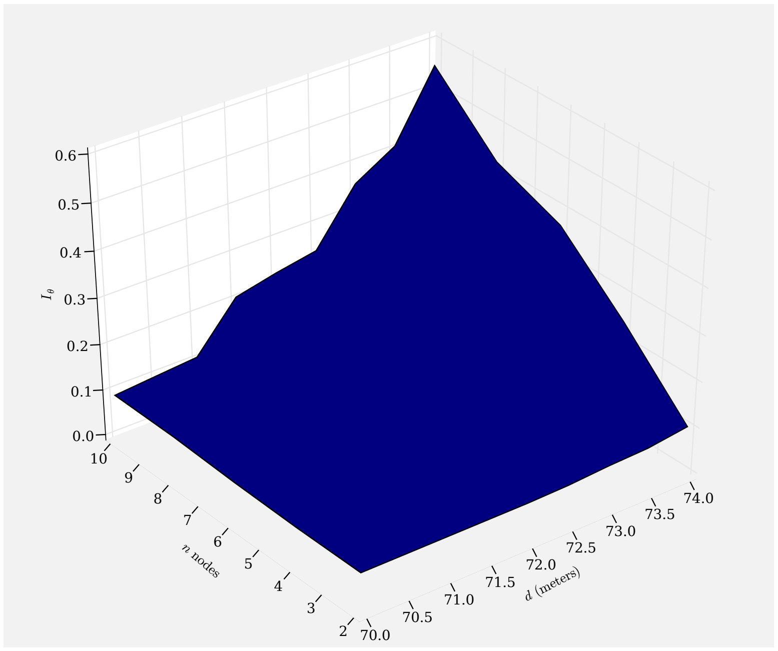

Figure 4.18. Distortion for motor example in which $S = -90$dBm, $P_T = -3$dBm, $n = 3.3$, $d_o = 8.0$ meters, each sample consists of 40 bits, and up to 2 samples will be transmitted if available in a single packet.

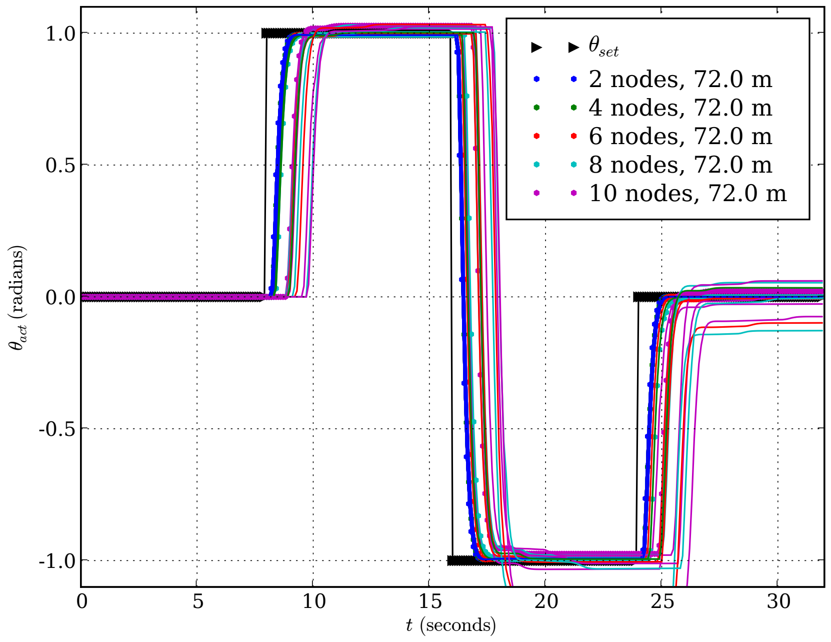

Figure 4.19. Typical step responses using *LDR* algorithm for nodes of 2, 4, 6, 8, 10 and transmission distances of 70, 70.5, 71, 71.5, 72.0, 72.5, 73.0, 73.5, 74 meters (each color in plot corresponds to a unique number of nodes).

Figure 4.19. Although the step responses are fairly smooth, when the nodes are 74.0 meters apart the probability of a successful packet transmission is $P = (1.0 - .010108)^{8\times(11+2+13+2\times5)} = 5.3\%$. Which corresponds to a capacity of 1861 bits/second for two nodes, and 207 bits/second for ten nodes ($m = 18$) with a required compression ratio nearing 4.

However, when a FIFO is full and if we choose to use no compression and either drop the oldest or the current data sample, we can reduce the distortion at the lower data rates. This was an *unexpected* result, after reading about previous simulations in which dropped data resulted in steady state error [11, Figure 9] [10, Figure 5, Figure 6]. However, we have been implementing our controller in a slightly different manner when

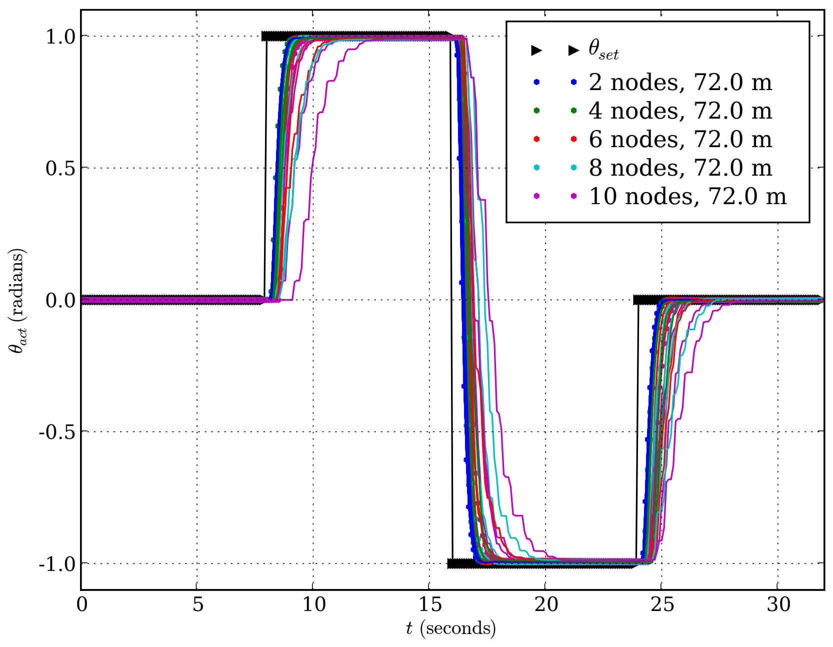

Figure 4.20. Typical step response by dropping either the latest sample or current sample when FIFO is full for nodes of $2, 4, 6, 8, 10$ and transmission distances of $70,\ 70.5,\ 71,\ 71.5,\ 72.0,\ 72.5,\ 73.0,\ 73.5,\ 74$ meters (each color in plot corresponds to a unique number of nodes).

not using compression. The controller only computes a new command when data from the plant is received, which implies we do not calculate a control command which will steer us away from our desired location by using a zero input. Furthermore, we achieved a significant improvement in distortion by simply dropping the sampled control input when data is not available from the plant. These two seemingly simple changes lead to a significant improvement in reducing distortion as can be seen in the step responses in Figure 4.20 and distortion plot in Figure 4.21.

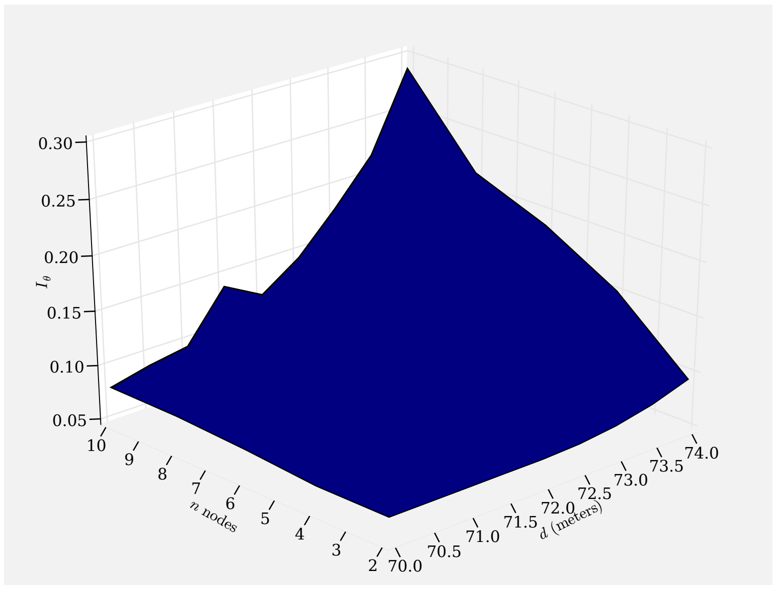

Figure 4.21. Distortion plot by dropping either the latest sample or current sample when FIFO is full.

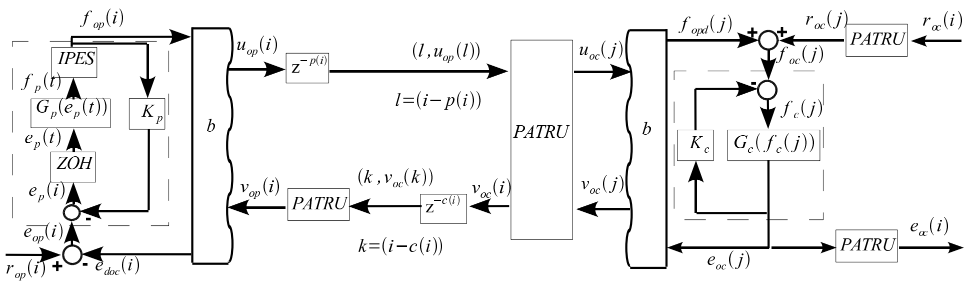

Figure 4.22. *Passive* digital control network with *Passive* Asynchronous Transfer Unit (*PATRU*).

4.3.2.2 Asynchronous Passivity

Figure 4.22 indicates how to implement a *passive* digital controller in an asynchronous manner. The transfer of data between the controller and the plant is handled by the *Passive Asynchronous Transfer Unit* (*PATRU*).

Definition 11 *Define the set I as the set of received indexes $l = (i - p(i))$ from the plant which correspond to the received tuple $(l, u_{op}(l))$ and the set J as the set of received indexes $k = (i - c(i))$ from the controller (via the PATRU) which correspond to the received tuple $(k, v_{oc}(k))$. When the plant and controller are initially enabled the sets I and J are empty. For simplicity of discussion we assume that the controller can instantly compute a new control command e_{oc} when new data arrives from the plant. The PATRU then handles the transfer of data as follows:*

1. *If the periodically generated tuple $(l, u_{op}(l))$ from the plant has arrived to the PATRU on the controller side then:*

```
if $l \in I^c$:
    $u_{oc}(j) = u_{op}(l)$
    $r_{oc}(j) = r_{oc}(i)$
    $I = l \cup I$
    calculate new $v_{oc}(j),$ and $e_{oc}(j)$
    $v_{oc}(i) = v_{oc}(j)$
    $e_{oc}(i) = e_{oc}(j)$
    $j = j + 1$
else:
    $v_{oc}(i) = 0$
    $e_{oc}(i) = 0$
transmit $(k, v_{oc}(k))$
```

2. *Otherwise if no periodically generated tuple* $(l, u_{op}(l))$ *from the plant has arrived to the PATRU on the controller side then:*

   $v_{oc}(i) = 0$

   $e_{oc}(i) = 0$

   transmit $(k, v_{oc}(k))$

3. *If the periodically generated tuple* $(k, v_{oc}(k))$ *from the PATRU on the controller side has arrived to the PATRU on the plant side then:*

   if $k \in J^c$*:*

   $v_{op}(i) = v_{oc}(k)$

   $J = k \cup J$

   else:

   $v_{op}(i) = 0$

4. *Otherwise if no periodically generated tuple* $(k, v_{oc}(k))$ *from the PATRU on the controller side has arrived to the PATRU on the plant side then:*

   $v_{op}(i) = 0$

Using definition 11 we give the following lemma:

Lemma 9 *Using the PATRU as defined in definition 11,*

$$\langle f_{op}(i), e_{doc}(i) \rangle_{N_i} \geq \langle e_{oc}(j), f_{opd}(j) \rangle_{N_j} \tag{4.40}$$

holds for all N_i *and* N_j*.*

Proof 17 *To begin, we note that* $N_i > N_j$ *since the controller will only process received data from the plant. From the scattering transform we also know that (4.40) can be* *equivalently written as*

$$\|(u_{op}(i))_{N_i}\|_2^2 - \|(v_{op}(i))_{N_i}\|_2^2 \geq \|(u_{oc}(j))_{N_j}\|_2^2 - \|(v_{oc}(j))_{N_j}\|_2^2. \tag{4.41}$$

It is sufficient for (4.41) to hold if both

$$\|(u_{op}(i))_{N_i}\|_2^2 \geq \|(u_{oc}(j))_{N_j}\|_2^2 \tag{4.42}$$

and

$$\|(v_{oc}(j))_{N_j}\|_2^2 \geq \|(v_{op}(i))_{N_i}\|_2^2 \tag{4.43}$$

hold. By definition 11 we know that $u_{oc}(j)$ can only consist of unique samples of $u_{op}(i)$ therefore (4.42) is obviously satisfied. Likewise, $v_{op}(i)$ can only consist of unique samples of $v_{oc}(j)$ or the value 0 therefore (4.43) is satisfied.

With lemma 9 we state the additional lemma which shows that using the *PATRU* an expression can be obtained which is sufficient for a *strictly-output passive* system when $i = j$.

Lemma 10 *Using the PATRU as defined in definition 11 in the control network depicted in Figure 4.22. The following inequality is satisfied:*

$$\langle f_{op}(i), r_{op}(i)\rangle_{N_i} + \langle e_{oc}(j), r_{oc}(j)\rangle_{N_j} \geq \epsilon(\|(f_{op}(i))_{N_i}\|_2^2 + \|(e_{oc}(j))_{N_j}\|_2^2) - \beta \tag{4.44}$$

in which $\epsilon = \min(\epsilon_{op}, \epsilon_{oc})$ and $\beta = \beta_{op} + \beta_{oc}$. When $i = j$ the network is strictly-output passive.

The proof follows along the lines as the one provided for theorem 4.

Proof 18 *First, by theorem 3-I, G_p is transformed to a discrete passive plant. Next, by* *theorem 2 both the discrete plant and controller are transformed into a strictly-output passive systems. The strictly-output passive plant satisfies*

$$\langle f_{op}(i), e_{op}(i)\rangle_{N_i} \geq \epsilon_{op}\|(f_{op}(i))_{N_i}\|_2^2 - \beta_{op} \tag{4.45}$$

while the strictly-output passive controller satisfies (4.46).

$$\langle e_{oc}(j), f_{oc}(j)\rangle_{N_j} \geq \epsilon_{oc}\|(e_{oc}(j))_{N_j}\|_2^2 - \beta_{oc} \tag{4.46}$$

Substituting, $e_{doc}(i) = r_{op}(i) - e_{op}(i)$ *and* $f_{opd}(j) = f_{oc}(j) - r_{oc}(j)$ *into (4.40) (which holds by lemma 9) yields*

$$\langle f_{op}(i), r_{op}(i) - e_{op}(i)\rangle_{N_i} \geq \langle e_{oc}(j), f_{oc}(j) - r_{oc}(j)\rangle_{N_j}$$

which can be rewritten as

$$\langle f_{op}(i), r_{op}(i)\rangle_{N_i} + \langle e_{oc}(j), r_{oc}(j)\rangle_{N_j} \geq \langle f_{op}(i), e_{op}(i)\rangle_{N_i} + \langle e_{oc}(j), f_{oc}(j)\rangle_{N_j} \tag{4.47}$$

so that we can then substitute (4.45) and (4.46) to yield

$$\langle f_{op}(i), r_{op}(i)\rangle_{N_i} + \langle e_{oc}(j), r_{oc}(j)\rangle_{N_j} \geq \epsilon(\|(f_{op}(i))_{N_i}\|_2^2 + \|(e_{oc}(j))_{N_j}\|_2^2) - \beta \tag{4.48}$$

in which $\epsilon = \min(\epsilon_{op}, \epsilon_{oc})$ *and* $\beta = \beta_{op} + \beta_{oc}$*. Thus (4.48) satisfies (2.19) when* $i = j$ *in which the input is the row vector of* $[r_{op}, r_{oc}]$*, and the output is the row vector* $[f_{op}, e_{oc}]$*.*

Interestingly, we can describe the controllers behavior in terms of i since the *PATRU* only transfers data to the controller when available from the plant or sends a 0 to the plant when no control data is available. Equivalently we can simply transfer a 0 to the controller when no data is available from the plant and use a switched controller G_c in which $G_c = G_{co}$ when data is present from the plant and set $G_c = 0$ when the *PATRU* fills in the missing data for $v_{oc}(i),\ v_{op}(i),$ and $e_{oc}(i)$ with 0.

Theorem 11 *Using the PATRU as defined in definition 11 in the control network depicted in Figure 4.22. The digital control network in Figure 4.22 is strictly-output passive.*

Proof 19 *From lemma 10 we have shown that (4.44) holds. Given definition 11, both*

$$\|(e_{oc}(j))_{N_j}\|_2^2 = \|(e_{oc}(i))_{N_i}\|_2^2 \tag{4.49}$$

and

$$\langle e_{oc}(j), r_{oc}(j)\rangle_{N_j} = \langle e_{oc}(i), r_{oc}(i)\rangle_{N_i} \tag{4.50}$$

will hold, therefore,

$$\langle f_{op}(i), r_{op}(i)\rangle_{N_i} + \langle e_{oc}(i), r_{oc}(i)\rangle_{N_i} \geq \epsilon(\|(f_{op}(i))_{N_i}\|_2^2 + \|(e_{oc}(i))_{N_i}\|_2^2) - \beta \tag{4.51}$$

holds in which $\epsilon = \min(\epsilon_{op}, \epsilon_{oc})$ *and* $\beta = \beta_{op} + \beta_{oc}$.

4.4 Conclusions

We have completed a comprehensive study and simulation of *passive* control over wireless networks. The papers which we have studied, addressed one way directional flow, however, *passive* networks have correlated bi-directional flow in which the plant sends sensor data to the plant, and the controller sends command data back to the plant. In order to complete our study, we needed new theorems to determine the network capacity and corresponding delays for bi-directional flow. We chose to study a token passing *MAC*, in which:

1. the plant (station $m = 2(n-1)$) and controller (station $m/2$) were stations of a n node ring network depicted in Figure 4.1,

2. we determined the network capacity, mean round trip travel time and variance of a packet of data in a ring network, τ_m, which is described by a Markov chain and transition matrix (4.9), and the corresponding formulas are given in Lemma 6,

3. in order to account for the overhead of the data acknowledge, header, and frame control sequence we introduced the corresponding definition for *data capacity* (Definition 9)

4. Definition 9 (and an extremely useful analysis relating packet error rate to node spacing with wireless transceivers such as the CC2420 in Appendix C.2) allows us to generate figures such as:

   a) Figures 4.2,4.4 in which the maximum data capacity is attained by sending the longest possible packet until a distance spacing of the nodes is such that p is fairly low,

   b) and Figures 4.3,4.5 in which a maximum spacing is indicated which provides the maximum data capacity $\times$ distance for relaying data over a network,

5. in order to account for the delays associated with buffering, pending data in a FIFO, we provide Lemma 7 for studying the corresponding *HoQ* delay,

6. Lemma 7 shows that the delay will undergo a *phase* shift in which it will suddenly increase at a critical number of nodes and node separation distance, as seen in Figure 4.6 and verified by simulation in Figure 4.7 such that

   a) the sudden increase in delay is equal to the maximum allowed buffer delay D and it occurs when the data rate $r < \frac{1}{\lambda_p} = \frac{2(n-1)}{p}$,
   b) this is intuitive since once data is generated at a rate that exceeds the capacity of the network, then the delay will continue to grow unbounded until packets are dropped which occurs when the FIFO is full.

In order to evaluate control performance over our *passive* wireless network we:

1. introduced a new definition for distortion Definition 10 which allowed us to evaluate and compare:

   a) a new adaptive *LDR* algorithm as described in Section 4.3.1, which we showed to be *passive* (Lemma 8),
   b) a *novel strictly-output passive* asynchronous controller as depicted in Figure 4.22 which only computes a new possibly non-zero control command when valid data from the plant is received (Section 4.3.2.1)

2. provide a new Theorem 11 (with corresponding new Lemma 10, and Lemma 9) showing that the asynchronous controller, governed by the *PATRU* defined in Definition 11, is indeed *strictly-output passive*,

3. Figure 4.21 shows the corresponding improvement in distortion for the asynchronous controller, as compared to the one which uses the adaptive compression scheme as shown in Figure 4.18.

CHAPTER 5

IMPLEMENTATION USING NECLAB

5.1 Introduction

This chapter is taken from [54]. *neclab* was successfully demonstrated at the Information Processing in Sensor Networks 2006 (IPSN 2006). At the conference we successfully balanced a ball on a beam (Figure 5.1) with wireless feedback using *neclab*.

Typically, when a controls engineer needs to develop a new closed-loop control system she develops the control system in phases. The first phase is to develop a mathematical model of the system and synthesize a controller. The second phase is to simulate the control system using tools such as MATLAB [74]. In the third phase, using the results from the simulations, the engineer integrates sensors, actuators, remote data acquisition and control equipment into the system. This is done in order to acquire additional data and refine the models in order to optimize the controller. When the third phase is complete, the engineer has optimized and deployed a robust control system. Systems with a higher degree of autonomy will also have fault detection and remote monitoring systems. Typically these digital control systems are developed using a dedicated data acquisition system attached to a cable interfaced to a computer running a real-time-control software, such as RTLinux [140]. For control systems in which a wired control system is not possible or desired, the available design tools for the engineer are limited at best.

In this chapter, a software environment is introduced called *neclab*, that is a software environment designed to allow easy deployment of networked embedded control systems, in particular wireless networked embedded control systems called *wnecs*. The components of *neclab* are presented in the following and described in terms of a classical control experiment, the ball and beam.

Note that most of the tools currently available to aid the engineer develop software for wireless embedded systems are geared specifically for sensing. The majority uses Berkeley's *TinyOS* [12]. Note also that the majority of the *TinyOS* applications listed in [12], are not designed to be wirelessly reconfigurable. For example, one reconfigurable system which uses *TinyOS* is Harvard's moteLab [131], where each mote is connected to a dedicated programming board that is connected to an Ethernet cable. This is necessary in order for each mote to be reconfigured in order to use *TinyOS*. A reliable protocol, called Deluge, to enable wireless programming of *TinyOS* applications has been developed [47]. Deluge is currently part of the *TinyOS* development tree, and should be an integral part of the next stable release of *TinyOS*.

We considered Deluge but in view of our sensor and control applications of interest we decided to work with an alternative to the *TinyOS* operating system called *SOS* [124]. *SOS* offered an alternative working design for network reprogramming for the three following reasons. First the *SOS* operating system utilizes a basic kernel which should only have to be installed on the mote once. The second key element is that the *SOS* kernel supports small, typically one-twentieth the size of a *TinyOS* application, dynamically loadable *modules* over a network. Last, the *SOS* kernel supports a robust routing protocol, similar to MOAP [116], to distribute *modules* over a wireless network.

We built *neclab*, our networked embedded control system software environment using *SOS*. Specifically, *neclab* is a collection of software consisting of five main components. The first component, *build utilities*, is a set of utilities designed to build and download all required software tools and libraries in order to use *neclab*. The second component, *SOS*, is an operating system developed by the Networked and Embedded Systems Lab (NESL) at UCLA. *SOS* is a highly modular operating system built around a message passing interface (MPI) which supports various processor architectures, including those on the MICA2 Motes. The third component, *sos utilities*, are the utilities to facilitate code development and deployment of *SOS modules*. The fourth component, *necroot*, is a file system structure and language designed to seamlessly interconnect individual motes for distributed control. The fifth component, *FreeMat utilities*, are a set of utilities to facilitate *wnecs* design using FreeMat. FreeMat is a free interpreter similar to MATLAB but has two important advantages. First, FreeMat supplies a direct interface for C, C++, and FORTRAN code. Second, FreeMat has a built-in API for MPI similar to MatlabMPI [52].

neclab provides a mini-language built on facilities similar to those supported by UNIX. A modern UNIX OS supports facilities such as pipes, sockets and filters. *neclab* allows the engineer to develop a *wnecs* by designing control *modules* which can be interconnected using *networking message pipe*s. A *networking message pipe* is an abstraction to pass arrays of structured binary data from one module to another over a network. A *networking message type* indicates how the data should be handled, for example, descriptors are used to indicate standard, error, routing, and control messages which are passed over a network; e.g. control messages are indicated by the *control message type*.

Specifically, a *networking message pipe* is used to interconnect data flows between *networking source*s, *networking filter*s, and *networking sink*s. A *networking source* creates data which will be sent over a network to *networking filter*s, and *networking* *sink*s. Similarly, a *networking filter* will receive data from either a *networking source* or another *networking filter*. The *networking filter* will proceed to optionally modify the data and send the new data to another *networking filter* or *networking sink*. A *networking sink* is where the network data flow for a given route ends. In order to implement *networking message pipe*s we will use the network message passing protocol provided by *SOS*. Like UNIX, *SOS* provides a way to run and halt programs which have a corresponding process id. These executable programs on *SOS* are known as *modules*. *neclab* provides an interface to pass *networking configuration* messages to a module in order to configure and enable the network flow of data between *modules* in the *wnecs* at run-time.

Using these facilities we will demonstrate an implementation of a highly parallel *wnecs* in which a secondary controller is reconfigured, while the primary controller maintains a stable control-loop. Once reconfigured, the roles of the two controllers will be switched. Other, highlights will illustrate that a controls engineer can actually create concise routing tables by simply describing a *wnecs* with *neclab*. This is a natural result of *wnecs* in general.

neclab's use of *SOS*'s dynamic memory allocation services, easily allows for a system which enables a control engineer to work through the second and third phases of her design project. This highly configurable environment without wires would have been difficult to implement with *TinyOS* since *TinyOS* does not support dynamic memory management. *SOS* on the other hand does. Other *SOS* features which *neclab* utilized are the ability to dynamically share functions and load executable files (modules) over a wireless channel while the *SOS* kernel is still running. *SOS* implies flexibility, and as a result it was chosen as the core component to *neclab*. For a more detailed discussion on the advantages and differences of *SOS* as compared to other solutions, refer to [44].

neclab is not the first project to utilize *SOS*. Other projects such as Yale's XYZ Sensor Node project [70] and various projects at NESL are starting to use *SOS* such as the RAGOBOT [35].

In presenting *neclab*, we will illustrate its use by presenting a typical undergraduate control lab problem modified to be a *wnecs*. We will then generalize this problem, by describing a tutorial application, which a user can create if five MICA2 Motes and a programming board are available. As the tutorial application is described we will highlight the various components of *neclab* which highlight the many issues that have to be addressed in order to develop a robust *wnecs*.

5.2 Problem Description

Consider an undergraduate controls laboratory experiment that teaches a student how to control the position of a metal ball on a beam. The experiment uses a motor as an actuator, and two variable Wheatstone bridges for sensing. The bridges measure angular position of the motor, and the absolute position of the ball on the beam. In the first laboratory experiment, the students are required to determine the actual model parameters of the ball and beam plant [126]. The next lab [125] teaches the student how to control the exact position of the ball on the beam. The student designs and implements the control system using MATLAB, Simulink [74], and the Wincon server for real-time control [98]. The sensor inputs and outputs are accessible through the MultiQ board. We are going to replace this system using *neclab*, the MICA2 motes and a general purpose I/O boards developed for the MICAbot [79].

Figure 5.1 illustrates such a system. With *neclab* installed on a host computer a MICA2_N_gateway mote is typically accessed via a serial port. Figure 5.1 indicates that MICA2_N_gateway is interconnected to a MIB510 programming board. See

[23] for additional details on the MIB510. In order to control the ball and beam plant the following control loops (jobs) need to be implemented (spawned). First the MICA2_N_actuator needs to reliably control the angular position α of the beam. This is achieved by controlling the angular position θ_l of the motor. The actuator will receive a desired angular position set-point and will control the motor. In networking terms, the MICA2_N_actuator behaves as a *networking sink* for networking messages, and takes actions based on the messages sent to its *control standard input*. According to [125] the desired response time for controlling α should be around 0.5 seconds. This is a fairly aggressive target to meet for the low data rate wireless network control system. As a result, we have initially kept this control loop internal to the MICA2_N_actuator. In order to do this we link this code statically to the kernel in order to guarantee a stable control loop on start-up. The second control loop involving the actual position of the ball, requires around a 4 second settling time, which is reasonable to implement over the wireless channel. This loop is accomplished by MICA2_N_sensor sampling data from the ball position sensor output with the ATmega 128L built-in 10 bit A/D converter (see [5] for additional information on this chip's features). The MICA2_N_sensor behaves as a *networking source* for generating networking messages, sending its data along to MICA2_N_controller-A and MICA2_N_controller-B respectively. Depending on which controller is enabled, the enabled controller will behave as a *networking filter* by calculating an appropriate command to send to MICA2_N_actuator based on the users desired set-point received. Figure 5.2, illustrates how this system can be implemented using *SOS* modules and messages which we will refer to as we discuss *neclab*.

5.3 *neclab*

Looking at figure 5.2, one can appreciate the number of distinct software components required for an engineer to obtain a working *wnecs*. In this figure the engineer has successfully built and installed kernels on five motes, loaded all the required modules on to the network, and created routing tables in order to create a stable closed loop controller to monitor and maintain the position of the ball. In order to use *neclab* the engineer must first download the 1.x version of *SOS* to a unix based PC. The location *SOS* is installed will be referred to as SOSROOT; *neclab* will be in the SOSROOT/contrib/*neclab* directory which will be referred to as NECLABROOT. From there all the engineer needs to do is follow the instructions in the NECLABROOT/README_SOS-1.X file. The first task that *neclab* will do for the user is download, build and install all the necessary tools to build and install software on the MICA2 motes so to work with the FreeMat environment. Once the tools are installed, *neclab* will apply some minor patches in order to fully utilize the *SOS* software. From there the engineer should test and fully understand the example blink_lab. The blink_lab is discussed in Appendix A.

5.3.1 *build utilities*

neclab has been designed so that a user does not require root access to build and install her tools. This offers two distinct advantages, the first being that the user can modify any appropriate component that is needed to maximize the performance of the system. For example, *neclab* actually downloads and allows the user to build an optimized BLAS (Basic Linear Algebra Subprograms) library using ATLAS (Automatically Tuned Linear Algebra Software) [132]. Second it provides all users with a consistent tool-kit eliminating potential software bugs associated with not using a consistent tool-chain. The key tool used is the build_tool_makefiles program which reads a configuration file located in NECLABROOT/etc/build.conf and generates a custom makefile for all the following tools:

- perl – key tool for various utilities in *neclab* [129]
- avr-binutils – used for the MICA2 and MICAz motes [37]
- avr-gcc – used for the MICA2 and MICAz motes [36]
- avr-libc – used for the MICA2 and MICAz motes [82]
- ATLAS – used with FreeMat[132]
- FreeMat [6]
- uisp – used to load an image on to the MICA2 and MICAz motes [81]
- tcl – (Tool Command Language) required for the tk library [95]
- tk – graphical user interface toolkit library required for python [95]
- python – [128] used in conjunction with pexpect [115] for automation
- SWIG – [8] is a tool to connect programs written in C and C++ with high-level programming languages such as perl and python.

The auto-generated makefiles are then installed in each corresponding NECLABROOT/src/build_utilities/$< tool >$ directory and invoked to download, build and install each $< tool >$. The installation directory is in NECLABROOT/tools/. As a result any user can build and use her own tools, without having to ask the system administrator for permission!

5.3.2 *SOS*, and *sos utilities*

Referring back to Figure 5.2, on the local host PC (HOST_PC), the engineer has just finished creating a *wnecs* using the *neclab_run* tool provided by *neclab*. Each mote has a kernel; however, they do not have to be unique, as is clearly shown in Figure 5.2. For example, the MICA2_N_gateway mote has the sosbase kernel which has a statically linked module which we will refer to by it's process id SOSBASE_PID. Other motes such as MICA2_N_sensor, MICA2_N_controller-A, and MICA2_N_controller-B have a blank_sos kernel with no statically linked modules. Finally the MICA2_N_actuator has a custom kernel, custom_sos, with statically linked modules ANG_POS_SEN_PID (a module which manages the angular position sensor), and MOT_ACT_PID (a module which controls the angular position on the motor). The custom_sos kernel was required to generate a pwm output in order to drive the H-Bridge on the MICAbot board. The remaining modules which are dynamically loaded and unloaded over the network, are either in an active or inactive state. When in an inactive state they do not consume any additional processor RAM but do take up program space in the flash.

In order to load and unload the modules the following programs are required. First, the sos server, sossrv, needs to be built, installed and started. *neclab* manages this with the makerules and *neclab_{sim,run}* commands. The makerules first manage building and installing sossrv into NECLABROOT/tools/bin. The *neclab_{sim,run}* commands can be used either to begin a simulation of a *wnecs* with the *neclab_sim* command or to run a *wnecs* using the *neclab_run* command. Either command can be treated as equivalent for discussion. The *neclab_run* command starts the sossrv and connects it to the MICA2_N_gateway mote. Figure 5.2 indicates that sossrv is listening for clients on HOST_PC via the loop-back interface on port 7915 while listening for incoming packets filtered through the MICA2_N_gateway attached to /dev/ttyS0.

Next the *neclab_run* tool creates an input fifo and starts the *SOS* modd_gw client. The modd_gw client is an *SOS* application that runs natively on a PC, it provides a shell like interface in which user input can be redirected to the fifo for automation. The modd_gw client maintains a database file .mod_version_db local to where it is started. This database tracks the different dynamic modules which are loaded and unloaded from the network. If another engineer chose to use the same motes in the lab, they will either need access to this file or re-install new kernels on all the motes. As a result *neclab* makes sure that the modd_gw is started such that the database is located in a publicly accessible directory such as /tmp so others can access and run their own experiments. The *neclab_run* tool then proceeds to build all the required network modules, load them on to the network, and establish the networking routes for the control system. Lastly, the closed loop control system is enabled and can be monitored and modified as necessary.

Another tool *neclab* provides is the create_module_proto tool to generate a new module prototype for beginning development. *neclab* has built into its makerules a mechanisms to generate tags files to assist in tracking all the interrelationships that modules have with the kernel and other modules. Once the engineer has a satisfactory implementation she can use the appropriate options from the *neclab_run* tool to easily rebuild and re-install new module images as necessary. These tools provide a stream-lined mechanism for simulating and generating *wnecs*. Building off of *SOS*'s novel technique of tracking module version numbers as they are dynamically loaded and unloaded. *neclab* has created a file-structure known as *necroot* to track, simulate, and develop *wnecs*.

5.3.3 *necroot*

Modules are tracked by their process id, similar to a process id generated by the ps command on a unix machine. This id is actually used to direct messages which are passed on an *SOS* network. Every message passed on the *SOS* network has a corresponding destination process id and address. The process id field; however, is only 8 bits, which supports only 255 unique process ids. Clearly more than 255 unique modules will be developed to be run on the *SOS* operating system, so there needs to be a clean way to address this limitation. *SOS* uses two files to define the association of a process id with a given module, mod_pid.h and mod_pid.c. The gen_mod_pid tool combined with makerules and the *necroot* design allow for dynamic generation of mod_pid.h and mod_pid.c for a corresponding lab project.

In *necroot* the NECLABROOT/src/*necroot*/modules.conf provides a line by line list in which each entry consists of a full-path to a corresponding module and a short description of the module. Each item is delimited by a colon. The module process id is parsed directly from each modules.conf entry and added to mod_pid.h. The corresponding description is then added to mod_pid.c for simulation and debugging. Furthermore, static modules and modules to be loaded over the network are uniquely identified by grouping these modules between the $< static >, < /static >$, and $< network >, < /network >$ tags respectively. The modules.conf is to serve as a global file, in it the key modules for *neclab* are listed. These entries include the moduled_pc module (MOD_D_PC_PID) and *neclab*'s configuring module (CONFIGURING_PID). The moduled_pc module, is a statically linked module which runs on the modd_gw client. The configuring module provides an interface for creating *networking source*s, *networking sink*s and *networking filter*s. It also provides the interface to configure modules and enable the *control standard input*/*control standard output* networking design referred to in the introduction. The remaining modules are identified in the engineer's NECLABROOT/labs/ball_beam_lab/etc/modules.conf.local file.

The next issue is to create a design which would allow a user to easily manage programming and tracking each corresponding mote's kernel and module configuration. This is solved by the construction of the NECLABROOT/src/*necroot*/etc/network.conf and the corresponding NECLABROOT/labs/ball_beam_lab/etc/network.conf.local files. Each entry follows nearly the same language structure as the build.conf file described in the *build utilities* section. The only difference instead of identifying a tool, each entry describes a mote and all the properties local to that mote. Using the build_mote_makefiles tool, a custom makefile for each mote described in the network configuration files is generated. Then the *necroot* directory structure is generated in the ball_beam_lab directory. Furthermore the engineer can use *neclab_run* with the appropriate options to build and install the corresponding kernel on to her motes. For reference, a typical network.conf entry for a mote is as follows:

```
<mote>
SOS_GROUP = 13
ADDRESS = 2
NECLAB_PLATFORM = mica2
KERNEL_DIR = ${NECLABROOT}/src/patched_sos-1.x/sosbase  #$
X = 6
Y = -12
Z = 5
LOC_UNIT = UNIT_FEET
TX_POWER = 255
```

```
#CHANNEL = ? has no effect for mica2
</mote>
```

The configuration language is GNU Make combined with some XML-like tags to separate each mote entry. Although, not true for mobile motes, each non-mobile mote has a fixed location. This information is typically used for geographic routing protocols, such as those implemented on Yale's XYZ platform. *SOS* has built in the capability to track the location of each mote into its kernel; hence, the X, Y, Z, and LOC_UNIT entries. The *neclab* project assisted in this design by introducing the Z dimension, a LOC_UNIT to associate relative position with a given dimension, and a gps_loc variable to track GPS location information. The gateway mote should typically assign itself a GPS location in order to assist with routing and interconnecting of other laboratories around the globe. Presently, gps_loc maintains precision down to one second. A sample GPS entry, here at Notre Dame would have the following format (note the Z coordinate is an elevation relative to sea-level in feet).

```
GPS_X_DIR = WEST
GPS_X_DEG = 86
GPS_X_MIN = 14
GPS_X_SEC = 20
GPS_Y_DIR = NORTH
GPS_Y_DEG = 41
GPS_Y_MIN = 41
GPS_Y_SEC = 50
GPS_Z_UNIT = UNIT_FEET
GPS_Z = 780
```

Note, each platform that *SOS* supports has a wireless radio setting unique to each mote.

In the above sample the radio was set to full power. *neclab* is also working on formalizing the interface to modify the frequency and channel of the radio. As the comment notes, the MICA2 mote does not currently support the channel interface; however, the MICAz mote does. The other parameters such as the *SOS*_GROUP id is used to filter radio packets out in the network that do not belong to that particular group. Each mote entry should have a unique, (*SOS*_GROUP, ADDRESS) pair. Being able to group motes and designate different channels, provides one way to allow multiple laboratories to interact and perform co-operative tasks.

For example, the ball and beam lab, can be modified to simulate the transfer of one ball from one station to another, as might be done in transferring parts from one assembly cell to another. Furthermore, by building in the ability to group the motes and assign each group to a separate radio channel, we have created a mechanism to create groups which can co-operate without worrying about generating radio interference. This feature we plan to exploit with the MICAz motes when developing routing algorithms between groups. The routing will be implemented between the gateway motes of a group. The gateway mote will typically be attached to their respective HOST_PC and the routing of packages will occur over the Internet.

Finally, the issue of declaring the desired kernel is addressed with the KERNEL_DIR entry. The KERNEL_DIR entry provides the full path to the desired kernel to be loaded and built. Any modules that the engineer plans to statically link in her kernel should be addressed in her respective kernel makefile.

After the engineer has successfully built her newly created *necroot* file structure, the following root tree will result.

```
[user@host_pc ball_beam_lab]$ find ./ -name "mote*"
./root/group13/mote1
```

```
./root/group13/mote2
./root/group13/mote3
./root/group13/mote4
./root/group13/mote5
./root/group14/mote1
./root/group14/mote2
./root/group14/mote3
./root/group14/mote4
./root/group14/mote5
[user@host_pc ball_beam_lab]$
```

In each mote directory there is a module configuration file, corresponding to every module identified in the corresponding modules.conf and modules.conf.local files. These are used in conjunction with the configuring module to set module parameters and set up routes in the network. Looking in the mote1 directory for example the user will see the following additional files.

```
[user@host_pc ball_beam_lab]$ ls -1 root/group13/mote1/*.mod
root/group13/mote1/ball_beam_con.mod
root/group13/mote1/ball_pos_sen.mod
root/group13/mote1/configuring.mod
root/group13/mote1/moduled_pc.mod
root/group13/mote1/ang_pos_sen.mod
root/group13/mote1/mot_act.mod
[user@host_pc ball_beam_lab]$
```

These user editable files provide an interface in order to use the configuring module to create routes. These routes can be configured using the command line or can optionally be declared in the routing.conf file. All lines starting with a # are comments. What the following segment of routing.conf should illustrate is that we have created a compact language to describe networking routes. The net_source, net_filter, and net_sink commands are intended to emphasize that a module will be configured to be in one of those three states. The optional arguments such as –d{0,1} are to show that the engineer can describe up to two destinations and the behavior is determined by the respective –m{0,1} option which describes the *networking message type* for each destination. This allows an engineer to redirect control data flow into error data flow for monitoring for example. It should also be noted that although we are using the *necroot* file system to maintain and describe our routes, an ad-hoc routing module could be designed to utilize the configuring module interface in order to build arbitrary networking routes based on some metric. This language can further be utilized to describe these routes using a graphical user interface.

The ball_beam_lab's etc/routing.conf file is as follows:

```
#!/bin/sh
#routing.conf
#define some constants
TIMER_REPEAT=0;TIMER_ONE_SHOT=1
SLOW_TIMER_REPEAT=2;SLOW_TIMER_ONE_SHOT=3
MSG_STANDARD_IO=35;MSG_ERROR_IO=36
MSG_CONTROL_IO=37;MSG_ROUTING_IO=38
cd $NECLABROOT/labs/ball_beam_lab/root/group13
#Main control-loop
#MICA_2_N_sensor -> MICA_2_N_controller-A -> MICA_2_N_actuator
#Secondary control-route
```

```
#MICA_2_N_sensor -> MICA_2_N_controller-B
#Configure the MICA_2_N_sensor (mote2/ball_pos_sen) routes
net_source -m mote2/ball_pos_sen -t "$TIMER_REPEAT:1024"  \
           --d1=mote3/ball_beam_con --m1=$MSG_CONTROL_IO  \
           --d2=mote4/ball_beam_con --m2=$MSG_CONTROL_IO
#Configure MICA_2_N_controller-A (mote3/ball_beam_con) routes
net_filter -m mote3/ball_beam_con -t "$TIMER_REPEAT:2048" \
           --d1=mote5/mot_act --m1=$MSG_CONTROL_IO        \
           --d2=mote1/moduled_pc --m2=$MSG_ERROR_IO
#Configure MICA_2_N_controller-B (mote4/ball_beam_con)
net_sink -m mote4/ball_beam_con -t "$TIMER_REPEAT:2048"
#Configure MICA_2_N_actuator (mote5/mot_act)
net_sink -m mote5/ball_beam_con -t "$TIMER_REPEAT:256"
#Activate the routes $
net_enable {mote5/mot_act,mote4/ball_beam_con}
net_enable {mote3/ball_beam_con,mote2/ball_pos_sen}
```

As shown in Figure 5.2, three routes are clearly established. One route establishes the control loop from sensor to controller to actuator. A second route delivers sensor and controller debugging information back to a gateway mote. A third route enables a second controller to act as a slave. The ball-position-sensor module on mote2 is configured as a *networking source*, generating a *control standard input output* message to be sent to the ball-beam-controller module, on mote3 every second or 1024 counts. The ball-beam-controller on mote3 is configured as a *networking filter*. As a *networking filter* it handles *control standard input output* messages from mote2's ball-position-sensor, computes the appropriate command and sends a *control standard input output* message to mote5's motor-actuator module. Furthermore mote3's ball-beam-controller module will generate a debugging message destined for the gateway mote1's non-resident moduled_pc module every two seconds. By sending information to the gateway mote, *neclab* can support remote monitoring. The moduled_pc module was arbitrarily chosen to illustrate that an arbitrary non-resident module id can be reserved in order to pass a message up to the sossrv program and a client can be designed to handle and display this message on the engineers HOST_PC display. Terminating the control loop route as a *networking sink*, mote5's motor-actuator, handles *control standard input output* messages from mote3's ball-beam-controller and actuates the beam. The motor-actuator controls the angular position of the beam and requires a faster control loop of a quarter of a second; hence, the timer is set to 256 counts.

5.3.4 *FreeMat utilities*

The *FreeMat utilities* will provide an interface for users familiar with MATLAB to appropriately modify configuration parameters for *neclab* modules designed and written in C. These utilities are currently the least developed for *neclab*; however, the major design problems have been confronted. There were numerous obstacles which had to be overcome in order to develop these utilities. The first accomplishment was getting FreeMat to build and install. Second was to integrate ATLAS, so that the engineer can have an optimized matrix library for simulation and development. Third was to develop the configuring module to allow a higher level networking protocol to be implemented in order to interconnect modules in a manner similar to the MPI protocols identified in the introduction. Lastly, was identifying SWIG as a possible candidate to assist with generating shared libraries to allow us to interface our *SOS* modules with FreeMat. We have used SWIG to generate interface files and shared libraries for python which we have used to create our initial graphical user interface application. We have also used SWIG to interface to our configuring module and do initial testing of the module. Although, FreeMat does provide a native interface for C and C++ programs, we feel that learning to effectively use SWIG to handle interfacing with *SOS* will allow a more flexible development environment in which users of *neclab* can use whatever high-level software tools they desire to use.

The *FreeMat utilities* will allow an engineer to generate her own routing tables while allowing users to receive and monitor data from modules using the *FreeMat client*. The *FreeMat client* will connect to the sossrv and relay all data destined for the *FREEMAT_MOD_PID*. Setting parameters and displaying data should be transparent due to the configuring interface provided by the combined configuring module and the data flash swapping interface provided by *SOS*. The configuring interface provides all the necessary elements for a module to establish up to two routes, configure a timer and change up to 8 bytes of parameter data in the modules RAM. To handle larger datasizes the user can either rely on the larger *SOS* RAM memory block of 128 bytes. The difficulty is that there are only four 128 byte RAM blocks available for the MICA2 and MICAz motes on *SOS*. The problem is further compounded in that the radio requires at least one 128 byte block to receive an *SOS* message over the network. In order to simultaneously send and receive a 128 byte message, a second 128 byte block of RAM needs to be allocated by the radio. This means that the user essentially has only two 128 large blocks of RAM available for allocation and they should be used for temporary operation such as being allocated to perform linear algebra routines. The second option is to dedicate a section of internal flash memory for storing configuration parameters and reading the configuration parameters into the local stack during module run-time. This is a preferred option because swapping in 256 bytes of data into the stack should only require around a tenth of a millisecond. This is a feasible option as long as the engineer utilizes the co-operative scheduler provided by *SOS*, and avoids interrupt service routines, and nested function calls which require large amounts of the stacks memory.

Being able to effectively manage the RAM and stack will allow *neclab* to support a much larger design space. For example, by gaining the ability to configure up to 256 bytes of data, the engineer can begin to develop four-by-four full-state observers. The following sections of configuring.h illustrate both the *networking configuration* and *control standard input* interface. The *networking configuration* is defined by the configuring_message_type. The *control standard input* is handled using the standard_io_message_t and indicated by the MSG_CONTROL_IO message id.

```
#define MSG_CONFIGURING         MOD_MSG_START
#define MSG_CONFIGURING_ENABLE  (MOD_MSG_START + 1)
#define MSG_CONFIGURING_DISABLE (MOD_MSG_START + 2)
/* #define MSG_*_IO */
#define MSG_STANDARD_IO         (MOD_MSG_START + 3)
#define MSG_ERROR_IO            (MOD_MSG_START + 4)
#define MSG_CONTROL_IO          (MOD_MSG_START + 5)
#define MSG_ROUTING_IO          (MOD_MSG_START + 6)
/* #define MSG_*_IO */
#define CONFIGURING_SOURCE        (1<<0)
#define CONFIGURING_SINK          (1<<1)
#define CONFIGURING_FILTER        (1<<2)
#define CONFIGURING_ENABLED       (1<<3)
#define CONFIGURING_CONFIGURED    (1<<4)
#define CONFIGURING_RESERVED      (1<<5)
```

```
#define CONFIGURING_TIMER_0_BIT_0 (1<<6)
#define CONFIGURING_TIMER_0_BIT_1 (1<<7)
#define CONFIGURING_TIMER_0       0
#define SIZE_OF_MULTI_HOP_PACKET  SOS_MSG_HEADER_SIZE
#define SIZE_OF_MSG_SMALL_BLOCK   (32)
#define SIZE_OF_MSG_LARGE_BLOCK   (128)
typedef struct destination_type{
    uint16_t daddr;  /* Destination address  */
    sos_pid_t did;   /* Destination pid      */
    uint8_t  type;   /* Message type to send */
} __attribute__ ((packed))
destination_t;

#define CONFIGURING_N_DESTINATIONS 2
typedef struct configuring_type{
    uint8_t flag;  /* source, sink, etc. */
    uint8_t size;  /* Size of user data  */
    sos_pid_t pid; /* pid of user module */
    uint8_t pad;   /* pad for alignment  */
    /* destinations takes up to 8 bytes */
    destination_t destinations[CONFIGURING_N_DESTINATIONS];
    uint8_t       data[8]; /* 8 bytes for user to store
                             user specific data */
} __attribute__ ((packed))
configuring_t;/* 20 bytes leaving 12 bytes for 6 function
```

```
                 pointers in the app_state_t structure
                 = 32 bytes = SIZE_OF_SMALL_BLOCKS
                 for the avr processor */

typedef struct configuring_message_type{
    int32_t interval; /* The ticks/counts you want to place
                         for the timer if timer < 0 the
                         timer is disabled by definition! */
    configuring_t conf;
} __attribute__ ((packed))
configuring_message_t; /* 24 byte packet leaving 8 bytes
                         (8 required) for multi-hop routing
                          worst case */

typedef enum {UINT8_T, INT8_T, UINT16_T, INT16_T,
              UINT32_T, INT32_T, FLOAT_T,
              USER_STRUCT_T} standard_io_types;

typedef struct standard_io_message_type{
    /* Engineer can send a vector of data_types */
    uint8_t data_r;
    /* Engineer can send an array of structures
       as long as the size_of is specified */
    uint8_t data_size_of;
    /* One of standard_io_types */
    uint8_t data_type;
    /* sid of sender of standard io message */
    sos_pid_t sid;
    uint8_t  data[MAX_STANDARD_IO_LENGTH]; /*32-8-4=20*/
} __attribute__ ((packed))
standard_io_message_t;
```

Refer to the *neclab* documentation for additional details.

5.4 Conclusion

In developing *neclab* many challenging design issues were addressed and solved. *neclab* first addressed the need to have a consistent tool chain, such as cross-compilers, for the MICA2 motes. This design problem was solved using *neclab*'s *build utilities*. The next design issue to be addressed was automating all the tasks associated with loading software on to the MICA2 motes. This was addressed by developing *sos utilities*. These utilities assisted in generating new modules, loading modules on to the network, building and installing new kernels on to the MICA2 motes. The next design issue was to create a mechanism to track the type of kernels and properties of each MICA2 mote on the network. This was accomplished, using the *necroot* design. Building on the tools developed from *build utilities*, the network.conf file provides a simple interface to manage each motes kernel, static-modules, radio configuration, location, group, and address properties. Next by showing a well-designed configuring module interface combined with *necroot* we demonstrated an efficient way to establish routes using a routing.conf file. Finally, all these components allow us to build a set of *FreeMat utilities* to enable control-engineers to develop *wnecs*, using a MATLAB like interface.

Using the routing.conf file, we also illustrated how control systems naturally create their own routes. The sensor is a *networking source*, creating a time-base for the system and generating messages to be routed to either a *networking filter*, or *networking sink*. The controller is a *networking filter*, receiving messages from the sensor, modifying the data and sending an appropriate message to either a *networking filter*, or *networking sink*. In the ball and beam example the controller sent a command message to the motor actuator. The motor actuator behaved as a *networking sink*, receiving the messages from either a *networking source* or *networking filter* and applied the appropriate input to the motor. We also showed in the ball and beam example how by utilizing a redundant controller we can accomplish the reprogramming on the fly. Specifically, the second controller, initially configured as a *networking sink*, can be reconfigured with new control parameters and then switched to become a *networking filter* while configuring the initial controller to be a *networking sink*. This shows that if a control system requires extensive time to reconfigure itself, the parallel architecture just described, can safely do so without having to risk having the system go unstable. This is a valuable feature, which may be used for control, reconfiguration and fault diagnostics and identification, switching control, etc.

Working with the memory constrained MICA2 and MICAz motes, we identified a unique way to increase our module configuration space. State-space control applications, in particular, those which require a state-observer, require large amounts of memory to describe a model of the system. In order to describe the system and make it re-configurable, we have identified that a section of internal flash can be reserved for reading in module control parameters into the stack during module run-time. Using *SOS* with it's co-operative scheduler, an engineer can wirelessly send new control parameters to the internal flash section in order to reconfigure and tune a state-observer. The current implementation is slower, however, due to the increased memory read times from the internal flash section.

Although, *neclab*, is in its initial release, it has become a fairly advanced piece of software. One limitation, is related to the limited message routing infrastructure. Presently, *neclab* relies on the built in *SOS* message routing structure, which is quite advanced; however, configuration messages can only be established if the gateway mote can access an individual node directly on the network. We are particularly interested in adding a more advanced routing interface, to enable multiple-hop routing for initial configuration. We are going to be actively using *neclab* for designing *wnecs*, and hope the control-community will find *neclab* a useful tool for their own research.

5.5 Blink Example

As shown in Figure 5.3, a basic network has been created and a blink module has been loaded and enabled on each MICA2 mote in order to control the blink rate of the yellow, green, and red LEDs. Each mote is a member of group 2, and each mote name corresponds to its address (mote1 has an address of 1). The blink module utilizes the configuring module interface and a module specific LED structure. The LED structure contains the state of each LED, a basic clock structure for each LED, and a mask to selectively filter the state of a given LED. Mote1 serves as a *networking source* in which its timer is configured to repeat every 1024 counts or one second. Once configured, and enabled by a MSG_CONFIGURING_ENABLE message, mote1's blink module will change the status (on/off) of the red LED every 4 seconds, green LED every 2 seconds, and the yellow LED every 8 seconds. Each time the led_state changes a MSG_STANDARD_IO message is delivered to mote2's blink module. Figure 5.3 indicates that mote1 currently has the yellow and green LEDs on. The message received by mote2 contains the new led_state of mote1. Mote2, configured as a *network-*

ing filter masks the received led_state such that only the red and green led state will pass. As a result only mote2's green LED is indicated as on. In this example, we chose to let the interval timer for mote2's blink module update the new status of the led state on the physical blink display so there could be a delay up to half a second. This is done to illustrate that we can isolate the handling of asynchronous events with a synchronous task. Last, the filtered led_state of mote2 is delivered to mote3 as a MSG_STANDARD_IO message. Mote3, configured as a *networking sink* updates the newly received led_state and displays the lit green LED with a lag of no greater than half a second. Note that modules on the network can be enabled/disabled by broadcasting a MSG_CONFIGURING_{ENABLE,DISABLE} message or motes can be enabled/disabled individually. Configuring messages are delivered reliably to individual motes, using the SOS_MSG_RELIABLE interface. As a result, configuring messages which are broadcast can not currently be acknowledged as being reliably sent; however, may still be deemed useful for a quick initial shutdown of a system.

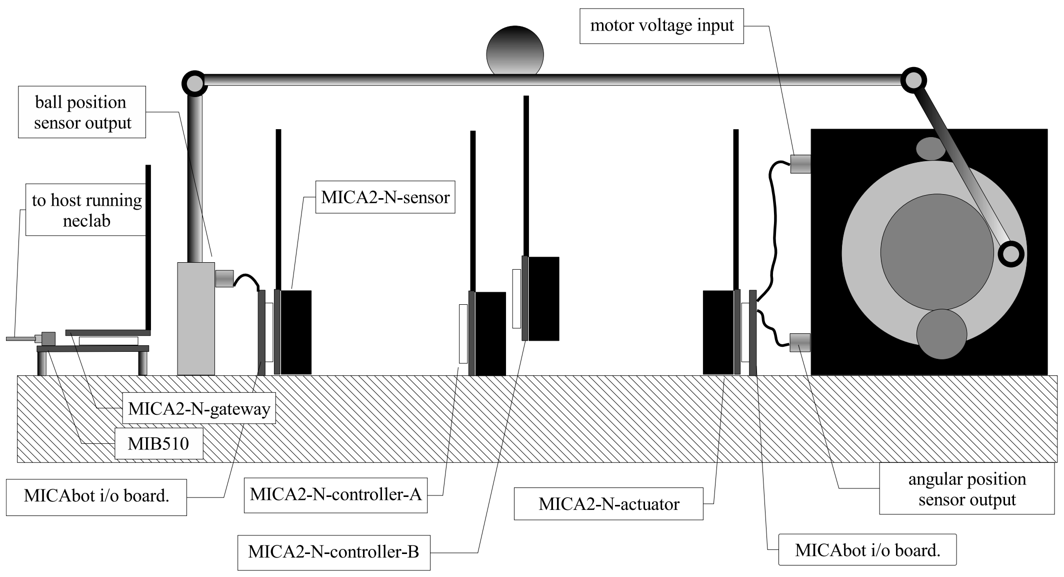

Figure 5.1. Ball and Beam Network Embedded Control System.

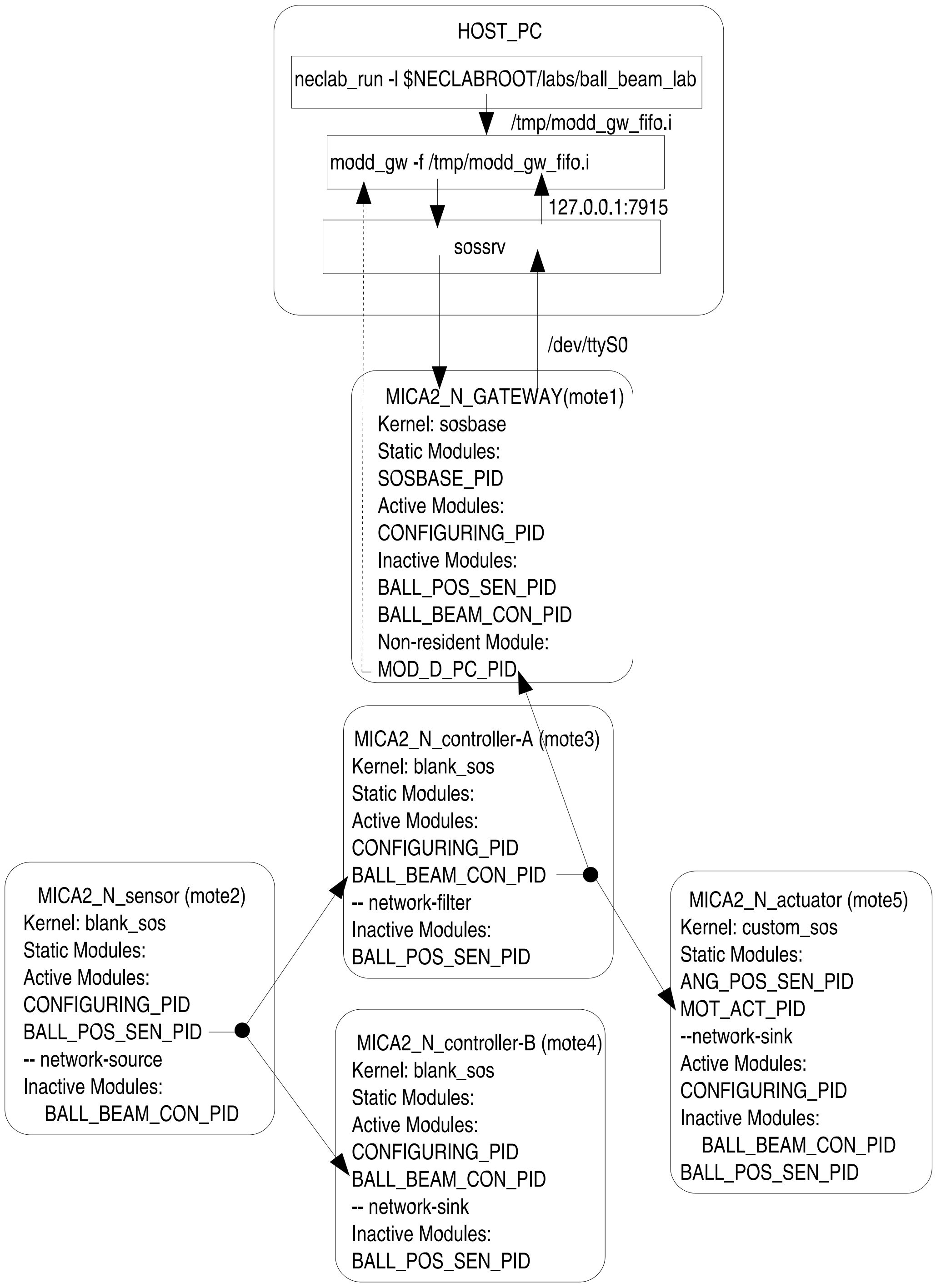

Figure 5.2. *neclab* block diagram for ball and beam control.

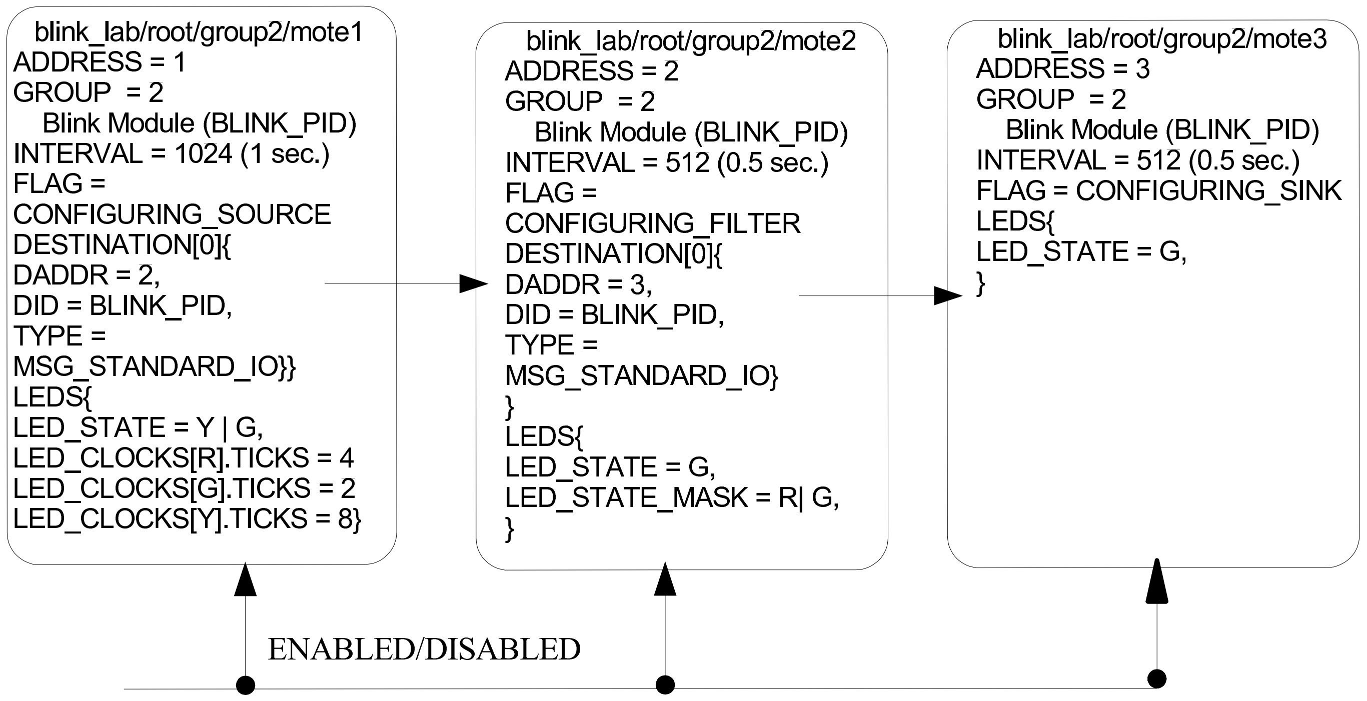

Figure 5.3. blink_lab network routing diagram.

APPENDIX A

NETWORKED EMBEDDED CONTROL SUBJECT TO RANDOM DELAYS

A.1 Strictly Positive Real and Strictly Input/Output Passive *LTI* Systems

It is not clear that anyone has formally stated that if a *LTI* system is *strictly-input passive* it is also *strictly-output passive*. Possibly this was implicitly understood in the earlier literature for *LTI* systems [25, 133, 135] in which the definition for a *strictly-input passive* system (Definition 3) was termed *strictly passive*. A *strictly passive* (*strictly-input passive*) system with *finite* l^2*-gain* is *strictly-output passive*. There is an emphasis in the literature that *strictly passive* (*strictly-input passive*) systems may or may not have *finite* l^2*-gain*, however, there is no specific statement that a *strictly-input passive LTI* systems has *finite* l^2*-gain*. This is emphasized by the fact that both [59, Corollary 1] [60, Corollary 2] note that discrete *SPR LTI* systems are indeed stable.

Definition 12 *[60, 122] Let $H(s)$ be a square rational transfer matrix in s. $H(s)$ is said to be strictly-positive real (SPR) if*

a) All elements of $H(s)$ are analytic in $\mathtt{Re}(s) \geq 0$;

b) $H(j\omega) + H^{\mathsf{T}}(-j\omega) > 0, \forall \omega \in \mathbb{R}$

c) i. $\lim\limits_{\omega \to \infty} \omega^2[H(j\omega) + H^{\mathsf{T}}(-j\omega)] > 0$, if $|D + D^{\mathsf{T}}| = 0$

ii. $\lim\limits_{\omega \to \infty} [H(j\omega) + H^{\mathsf{T}}(-j\omega)] > 0$, if $|D + D^{\mathsf{T}}| \neq 0$

Definition 13 *[60, 122] Let $H_d(z)$ be a square rational transfer matrix in z. $H_d(z)$ is said to be SPR if*

a) All elements of $H_d(z)$ are analytic in $|z| \geq 1$

b) $H_d(e^{j\theta}) + H_d^{\mathsf{T}}(e^{-j\theta}) > 0, \forall\theta \in [0, 2\pi]$

Note that both definitions are slightly more restrictive than those given by [122], however they are consistent with previous statements relating *SPR* to *strictly-input passive* systems [25, 60, 133]. Thus by Definition 12 and Definition 13 continuous and discrete *SPR LTI* systems are stable which implies that *strictly-input passive* or *strictly-output passive* systems are also stable. It has already been shown that *strictly-output passive* systems have *finite l^2-gain*, it remains to be shown that *LTI strictly-input passive* systems also have *finite l^2-gain*.

Theorem 12 *[80] The L^2/l^2-gain of a LTI system described by a transfer matrix $H(p)$ equals the H_∞ norm of H defined by*

$$||H||_\infty = \sup_{p\in\Omega} ||H(p)|| \tag{A.1}$$

where Ω is the right half plane $\Omega = \mathbb{C}_0$ for the continuous time (CT) case, and the exterior of the unit circle $\Omega = \mathbf{D}_1$ in the discrete time (DT) case. Moreover, for rational transfer matrices with no poles in Ω (such as SPR systems), the supremum can be calculated on the boundary of Ω (the imaginary axis in the CT case and the unit circle in the DT case).

Therefore, from Theorem 12 a continuous/discrete *LTI strictly-input passive* system which is *SPR* has *finite L^2/l^2-gain* which implies the *LTI* system is *strictly-output passive*[127, Remark 2.3.5].

Corollary 10 *Every continuous/discrete LTI system which is strictly-input passive has finite L^2/l^2-gain, therefore it also strictly-output passive.*

A.2 Observer Simulation Equations

In order to simulate an observer for a continuous *LTI* plant in which the actual state space matrices for the actual passive plant are denoted
$\{\mathbf{A_a} \in \mathbb{R}^{\mathbf{n\times n}}, \mathbf{B_a} \in \mathbb{R}^{\mathbf{n\times p}}, \mathbf{C_a} \in \mathbb{R}^{\mathbf{p\times n}}, \mathbf{D_a} \in \mathbb{R}^{\mathbf{p\times p}}\}$. The actual discrete equivalent matrices for a passive system are computed appropriately as described by (2.47), (2.48), (2.49), (2.50), and (2.51), and denoted as $\{\mathbf{\Phi_{oa}}, \mathbf{\Gamma_{oa}}, \mathbf{C_{oa}}\}$. If the observer is implemented on the plant side for a *LTI strictly-input passive* or *strictly-output passive* plant as depicted in Fig. 2.6 and described by Corollary 6, then the system can be described by

$$\begin{aligned}
\begin{bmatrix} \hat{x}(k+1) \\ x(k+1) \end{bmatrix} &= \begin{bmatrix} \mathbf{\Phi_{efo}} & \mathbf{K_e C_{oa}} \\ -b\mathbf{\Gamma_{oa} C_{efo}} & \mathbf{\Phi_{oa}} \end{bmatrix} \begin{bmatrix} \hat{x}(k) \\ x(k) \end{bmatrix} \\
&+ \begin{bmatrix} \mathbf{\Gamma_{efo}} \\ \mathbf{\Gamma_{efoa}} \end{bmatrix} (\sqrt{2b} v_{op}(k) + r_{op}(k)) \\
\begin{bmatrix} \hat{f}_{op}(k) \\ p(k) \end{bmatrix} &= \begin{bmatrix} \mathbf{C_{efo}} & \mathbf{0} \\ \mathbf{0} & \mathbf{C_{oa}} \end{bmatrix} \begin{bmatrix} \hat{x}(k) \\ x(k) \end{bmatrix} \\
&+ \begin{bmatrix} \mathbf{D_{efo}} \\ \mathbf{0} \end{bmatrix} (\sqrt{2b} v_{op}(k) + r_{op}(k))
\end{aligned} \tag{A.2}$$

in which

$$\mathbf{\Gamma_{efoa}} = \mathbf{\Gamma_{oa}}(\mathbf{I} - b\mathbf{D_{efo}}). \tag{A.3}$$

Similarly, if we implement the observer for a continuous plant on the "controller side" (i.e. when the plant is more accurately depicted as having a flow input and corresponding effort output) as depicted in Fig. 2.6 and described by Corollary 7 then the system can be described by

$$\begin{aligned}\begin{bmatrix}\hat{x}(k+1)\\ x(k+1)\end{bmatrix} &= \begin{bmatrix}\mathbf{\Phi_{feo}} & \mathbf{K_e C_{oa}}\\ -\frac{1}{b}\mathbf{\Gamma_{oa} C_{feo}} & \mathbf{\Phi_{oa}}\end{bmatrix}\begin{bmatrix}\hat{x}(k)\\ x(k)\end{bmatrix}\\ &+ \begin{bmatrix}\mathbf{\Gamma_{feo}}\\ \mathbf{\Gamma_{feoa}}\end{bmatrix}(\sqrt{\frac{2}{b}}u_{oc}(k) + r_{oc}(k))\\ \begin{bmatrix}\hat{e}_{oc}(k)\\ p(k)\end{bmatrix} &= \begin{bmatrix}\mathbf{C_{feo}} & \mathbf{0}\\ \mathbf{0} & \mathbf{C_{oa}}\end{bmatrix}\begin{bmatrix}\hat{x}(k)\\ x(k)\end{bmatrix}\\ &+ \begin{bmatrix}\mathbf{D_{feo}}\\ \mathbf{0}\end{bmatrix}(\sqrt{\frac{2}{b}}u_{oc}(k) + r_{oc}(k))\end{aligned} \tag{A.4}$$

in which

$$\mathbf{\Gamma_{feoa}} = \mathbf{\Gamma_{oa}}(\mathbf{I} - \frac{1}{b}\mathbf{D_{feo}}). \tag{A.5}$$

APPENDIX B

MATHEMATICAL PRELIMINARIES: WIRELESS CONTROL SUBJECT TO ACTUATOR CONSTRAINTS

B.1 Discrete Markovian Jump Linear Systems

Much work has been accomplished, that allows us to determine if a random control system is stochastically stable [49, 50, 55, 108, 110]. Unfortunately stochastic definitions for stability such as *stochastic stable (ss)*, *mean square stable (mss)*, *exponentially mean square stable (ems)*, and *almost sure stable (as)* do not imply any form of deterministic stability such as *Lyapunov stability* [55]. One class of systems which has received much attention, is the (continuous/discrete) *MJLS*. *MJLS* are stochastic systems which have been shown to be equivalently *mean square stable (mss)*, *exponentially mean square stable (ems)*, and *almost sure stable (as)* [31, 50]. The *MJLS* model was initially used to justify the design of controllers of systems subject to discontinuous or abrupt changes in particular they are used to modeling exponential failure/repair distributions [16]. Most recently these systems have been used to model and simulate *wnecs* [68, 108–110].

Following the notations used by in [49], a discrete *MJLS* can be described as shown in (B.1).

$$\begin{aligned} x_{k+1} &= A(r_k)x_k + B(r_k)u_k \\ y_k &= C(r_k)x_k \end{aligned} \tag{B.1}$$

The discrete sequence $k \in (1 \dots N)$ in which N can be infinite, the internal process state x_k is governed by the discrete matrices $A(r_k), B(r_k)$ while the process output y_k is described by $C(r_k)$. The matrices $A(r_k), B(r_k), C(r_k)$ are linear discrete matrices dependent only on the Markov state r_k. The Markov state r_k is a discrete time, discrete state Markov chain which has distinct values in the finite set $\mathcal{S} = \{1, 2, \dots, s\}$. The transition probabilities, π_k, which can be recursively determined at step k by the following difference (B.2).

$$\pi_{k+1} = \Gamma' \pi_k \tag{B.2}$$

$\Gamma \in \Re^{sxs}$ is the state transition matrix such that each element $p_{ij} = Pr\{r_{k+1} = j | r_k = i\} \geq 0$ and that for all $i \in \mathcal{S}$ the following equality holds $\sum_{j=1}^{s} p_{ij} = 1$. It is interesting to note that these conditions do not require the Markov chain to be stationary; however, if all elements $p_{ij} > 0$, then the transition matrix Γ will be irreducible and aperiodic. This implies that a unique stationary distribution π^* will result, independent of the initial state r_0 this distribution can be determined from (B.3).

$$\Gamma' \pi^* = \pi^* \tag{B.3}$$

It should be apparent from (B.3) that π^* is the unit eigenvector of the unit eigenvalue which can be calculated by solving for the null space of $(\Gamma' - I)$. A *wnecs*, which relies on feedback over an iid Bernoulli channel is an irreducible and aperiodic Markov process. Defining the probability of a successful radio transmission as p and defining

$r_k = 1$ for a successful transmission, the state transition matrix Γ has the following form.

$$\Gamma = \begin{bmatrix} p & 1-p \\ p & 1-p \end{bmatrix} \tag{B.4}$$

A fundamental result for these systems is the following Theorem 13 [49].

Theorem 13 *System (B.1) with $u = 0$ is said to be stochastically stable if and only if for a given set of symmetric matrices $\{W_i > 0 : i \in \mathcal{S}\}$, there exists a set of symmetric solutions $\{M_i > 0 : i \in \mathcal{S}\}$ of the following coupled matrix equations*

$$\sum_{j=1}^{s} p_{ij} A_i' M_j A_i - M_i = -W_i \tag{B.5}$$

B.2 Memoryless Nonlinearities

Two general classes of memoryless nonlinearities are to be reviewed. The first class of these nonlinearities is known as **sector** nonlinearities $\phi(u(x))$ in which $y(x) = \phi(u(x))\ \forall x$ in which x can be continuous time t or discrete time i.

Definition 14 *[25, Definition I-1.1] Let $\phi : \mathbb{R} \rightarrow \mathbb{R}$ with $\phi(0) = 0$. We say that $\phi \in \mathbf{sector}\ (k_1, k_2)$ iff $k_1 u^2 < u\phi(u) < k_2 u^2$, $\forall u \in \mathbb{R}$ with $u \neq 0$.*

Many equivalent statements can be made in regards to a memoryless scalar sector nonlinearity.

Theorem 14 *[25, Theorem I-1.2] Let $k_1, k_2 \in \mathbb{R}$ with $k_1 \leq k_2$. Let $\phi : \mathbb{R} \rightarrow \mathbb{R}$ with $\phi(0) = 0$. Such that the following statements are equivalent:*

i) $k_1 \leq \frac{\phi(u)}{u} \leq k_2,\ \forall u \neq 0$

ii) $k_1 u^2 \leq u\phi(u) \leq k_2 u^2,\ \forall u \in \mathbb{R}$

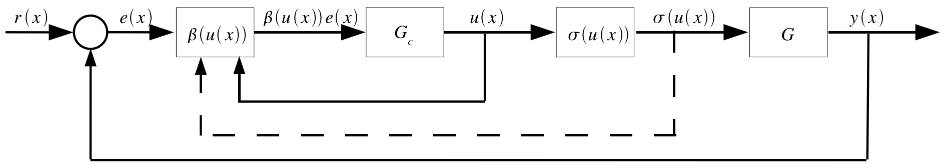

Figure B.1. Nonlinear controller arrangement studied in [15, 41, 42].

iii) $[\phi(u) - k_1 u][\phi(u) - k_2 u] \leq 0, \ \forall u \in \mathbb{R}$

iv) $|\phi(u) - cu|^2 \leq r^2 |u|^2, \ \forall u \in \mathbb{R}$

where

$$c \triangleq \frac{1}{2}(k_1 + k_2), \ r \triangleq \frac{1}{2}(k_2 - k_1), \ k_1 k_2 = c^2 - r^2$$

When the input u is a vector, such a set of equivalent statements are difficult to make. However, the following definition is fairly general and applies to a large class of sector nonlinearities. Note also that these sector nonlinearities can be typically treated as time varying if desired.

Definition 15 *[53, Definition 10.1] A memoryless nonlinearity $\phi : \mathbb{R}^m \to \mathbb{R}^m$ is said to satisfy a global* **sector** *condition belonging to sector $[K_1, K_2]$ if*

$$[\phi(u) - K_1 u]^\mathsf{T} [\phi(u) - K_2 u] \leq 0, \ \forall t \geq 0, \ \forall u \in \mathbb{R}^m \tag{B.6}$$

holds for some real matrices K_1 and K_2, where $K = K_2 - K_1$ is a positive definite symmetric matrix. If (B.6) holds with strict inequality, then $\phi(\cdot)$ is said to belong to a sector (K_1, K_2).

In [15, 41, 42] the following nonlinear control block $\beta(u(x))$ has been shown to effectively create stable control systems which are subject to an even broader class of memoryless nonlinearities, $\sigma(u(x))$, as depicted in Figure B.1. The controller has been studied when $e(x) = r(x) - y(x), x = t$ in which G and G_c are (*strictly*)-*positive real*[15]. Later it was studied when $e(x) = r(x), x = t$ ($x = i$) and G and G_c form a *passive-exponentially* (*geometrically*) *passive* pair [41, 42]. The only constraints on these memoryless nonlinearities is that $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ and $\forall i \in \{1, 2, \ldots, m\}$, if $u_i(x) = 0$ then $\sigma_i(u(x)) = 0$. Therefore, $\beta(u(x)) = \mathsf{diag}(\beta_1(u(x)), \ldots, \beta_m(u(x)))$, where

$$\beta_i(u(x)) = \begin{cases} \frac{\sigma_i(u(x))}{u_i(x)}, \text{ if } u_i(x) \neq 0 \\ 1, \text{ if } u_i(x) = 0 \end{cases} \tag{B.7}$$

We note that the latter case when $u_i(x) = 0$ has been defined as being arbitrary, however, we chose to use 1 because when $u_i(x) = 0$ then $\sigma_i(u(x)) = u_i(x)$ by assumption, hence, $\frac{\sigma_i(u(x))}{u_i(x)} = \frac{u_i(x)}{u_i(x)} = 1$ when $u_i(x) = 0$. Therefore,

$$\beta(u(x))u(x) = \sigma(u(x)) \tag{B.8}$$

and we also note that

$$\beta^{\mathsf{T}}(u(x)) = \beta(u(x)) \tag{B.9}$$

in which $x \in \{i, t\}$.

Furthermore this allows us to propose the following definition:

Definition 16 *A memoryless nonlinearity $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ is said to satisfy a global* **sector** *condition belonging to sector $[k_1, k_2]$ if*

$$k_1 u^{\mathsf{T}} u \leq u^{\mathsf{T}} \sigma(u) \leq k_2 u^{\mathsf{T}} u, \ \forall u \in \mathbb{R}^m \tag{B.10}$$

holds any $k_1, k_2 \in \mathbb{R}$. If (B.10) holds with strict inequality, then $\sigma(\cdot)$ is said to belong *to a sector* (k_1, k_2).

Theorem 15 *Let* $k_1, k_2 \in \mathbb{R}$ *with* $k_1 \leq k_2$*. Let* $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ *with* $\sigma_i(u_i = 0) = 0,\ i \in \{1, \ldots, m\}$ *and* $\beta(u)$ *satisfies (B.7). Such that the following statements are equivalent:*

i) $(k_1 I - \beta(u)) \leq 0$ *and* $(\beta(u) - k_2 I) \leq 0,\ \forall u \in \mathbb{R}^m$

ii) $k_1 u^\mathsf{T} u \leq u^\mathsf{T} \sigma(u) \leq k_2 u^\mathsf{T} u,\ \forall u \in \mathbb{R}^m$

Proof 20 *i)* $\rightarrow$ *ii) first we prove the left half of the inequality holds for both definitions.*

$$u^\mathsf{T}(k_1 I - \beta(u))u \leq 0,\ \forall u \in \mathbb{R}^m$$

$$k_1 u^\mathsf{T} u \leq u^\mathsf{T}(\beta(u)u),\ \forall u \in \mathbb{R}^m \tag{B.11}$$

$$k_1 u^\mathsf{T} u \leq u^\mathsf{T}\sigma(u),\ \forall u \in \mathbb{R}^m \tag{B.12}$$

Note that the simplification from (B.11) to (B.12) is a direct result of application of the definition for $\beta(u)$*. The proof for the right half of the inequality is as follows:*

$$u^\mathsf{T}(\beta(u) - k_2 I)u \leq 0,\ \forall u \in \mathbb{R}^m$$

$$u^\mathsf{T}(\beta(u)u) \leq k_2 u^\mathsf{T} u,\ \forall u \in \mathbb{R}^m \tag{B.13}$$

$$u^\mathsf{T}\sigma(u) \leq k_2 u^\mathsf{T} u,\ \forall u \in \mathbb{R}^m \tag{B.14}$$

.

ii) $\rightarrow$ *i) is fairly obvious when we substitute* $\sigma(u) = \beta(u)u$ *and* $u^\mathsf{T} u = u^\mathsf{T} I u$ *in to (B.10) which yields*

$$k_1 u^\mathsf{T} I u \leq u^\mathsf{T}\beta(u)u \leq k_2 u^\mathsf{T} I u,\ \forall u \in \mathbb{R}^m \tag{B.15}$$

in which

$$u^{\mathsf{T}}(k_1 I - \beta(u))u \leq 0,\ \forall u \in \mathbb{R}^m \tag{B.16}$$

$$(k_1 I - \beta(u)) \leq 0, \forall u \in \mathbb{R}^m \tag{B.17}$$

similarly,

$$u^{\mathsf{T}}(\beta(u) - k_2 I)u \leq 0,\ \forall u \in \mathbb{R}^m \tag{B.18}$$

$$(\beta(u) - k_2 I) \leq 0, \forall u \in \mathbb{R}^m \tag{B.19}$$

and completes the proof.

Next we provide the following definitions and theorem which relates how the bounds on the achievable maximum and minimum singular values for $\beta(u)$ relate to the sector $[k_1, k_2]$ bounds.

Definition 17 *The maximum achievable singular value squared $\sigma_{MAX}(\beta(u))^2$ is*

$$\sigma_{MAX}(\beta(u))^2 \stackrel{\triangle}{=} \max(\sigma_M(\beta(u))^2),\ \forall u \in \mathbb{R}^m \tag{B.20}$$

in which $\sigma_M(\beta(u))$ denotes the maximum singular value of the resulting matrix $\beta(u)$ for a given u.

Definition 18 *The minimum achievable singular value squared $\sigma_{MIN}(\beta(u))^2$ is*

$$\sigma_{MIN}(\beta(u))^2 \stackrel{\triangle}{=} \min(\sigma_m(\beta(u))^2),\ \forall u \in \mathbb{R}^m \tag{B.21}$$

in which $\sigma_m(\beta(u))$ denotes the minimum singular value of the resulting matrix $\beta(u)$ for a given u.

Theorem 16 *For a given sector $[k_1, k_2]$ nonlinearity:*

i) $\sigma_{MIN}(\beta(u))^2 = 0$ *if either* $k_1 = 0$*, or* $k_2 = 0$*, or* $\sigma(u) \in C^1,\ k_1 < 0 < k_2$*.*

ii) $\sigma_{MAX}(\beta(u))^2 = \max(k_1^2, k_2^2)$*.*

Proof 21 *Since* $\beta(u)$ *is a diagonal matrix, then* $\sigma_M(\beta(u)) = \max(|\lambda_M(\beta(u))|, |\lambda_m(\beta(u))|)$ *in which* $\lambda_M(\cdot)$ *denotes the maximum eigenvalue and* $\lambda_m(\cdot)$ *denotes the minimum eigenvalue of* $\beta(u)$*.*

i) If $k_1 \geq 0$ *than* $\sigma_{MIN}(\beta(u))^2 = k_1^2$*, therefore only* $k_1 = 0$ *will satisfy* $\sigma_{MIN}(\beta(u))^2 = 0$ *for this case. Next if* $k_2 \leq 0$ *than* $\sigma_{MIN}(\beta(u))^2 = k_2^2$*, therefore only* $k_2 = 0$ *will satisfy* $\sigma_{MIN}(\beta(u))^2 = 0$ *for this case. Lastly, If* $\sigma(u) \in C^1$*, then if* $k_1 < 0 < k_2$*, then there exists a* u *such that* $\frac{\sigma_i(u)}{u_i} = 0,\ u_i \neq 0$ *which implies that* $\sigma_{MIN}(\beta(u))^2 = 0$*.*

ii) From Theorem 15-i, we see that $\min(\lambda_m(\beta(u))) = k_1, \forall u \in \mathbb{R}^m$ *and* $\max(\lambda_M(\beta(u))) = k_2, \forall u \in \mathbb{R}^m$*, therefore* $\sigma_{MAX}(\beta(u))^2 = \max(k_1^2, k_2^2)$*.*

APPENDIX C

MATHEMATICAL PRELIMINARIES: WIRELESS DIGITAL CONTROL OF CONTINUOUS PASSIVE PLANTS OVER TOKEN RING NETWORKS

C.1 Key Formulas For Finite Markov Chains

We summarize some key results given by [113] as they allow us to evaluate various random processes which can be described by a finite Markov Chain. We assume the chain can be described by a finite set of states $s \in \{1, \ldots, m\}$ and the state transition matrix P. If the chain is an absorbing chain, it will reach an absorbing state with probability 1. An absorbing state is one that once reached, it will not leave that state (i.e. $P_{(i,i)} = 1$). Hence, we are interested in the following processes:

1. u_j: The number of times the process is in the nonabsorbing state j before being absorbed.
2. v: The number of steps taken before absorption.
3. w: The number of different nonabsorbing states entered before absorption.
4. x: The state in which the process is absorbed.

Using the following notation:

1. $\mathrm{Pr}_i[p]$: The probability that p occurs when the process is started in state i.
2. $M_i[f]$: The mean value of the random variable f when the process started in state i.

3. $C_i[f,g]$: The covariance of f and g when the process is started in state i.

The transition matrix can be put in a canonical form P in which the absorbing states are placed first such that

$$P = \begin{bmatrix} I & \mathbf{0} \\ R & Q \end{bmatrix} \tag{C.1}$$

From which we can compute the following first and second moments based on the fundamental matrix $N = (I-Q)^{-1}$ as given in Table C.1.

TABLE C.1

FIRST AND SECOND MOMENT STATISTICS FOR AN ABSORBING MARKOV CHAIN [113]

First Moment	Second Moment
$N = \{M_i[u_j]\} = (I-Q)^{-1}$	$N_2 = \{M_i[u_j^2]\} = N(2N_{dg} - I)$
$\tau = \{M_i[v]\} = Ne$	$\tau_2 = \{M_i[v^2]\} = (2N - I)Ne$
$H = \{M_i[w]\} = NN_{dg}^{-1}e$	

In which N_{dg} is a matrix with the same diagonal elements as N and 0 elsewhere, and e is a column vector of ones. It is also given that $B = \{\mathrm{Pr}_i[x=j]\} = NR$. Denoting the covariance of v given i as $\sigma_v^2 = \{C_i[v,v]\}$ can be calculated as follows:

$$\sigma_v^2 = \tau_2 - \mathsf{diag}(\tau)\tau = [2N - I - \mathsf{diag}(Ne)]Ne = [2N - I - \mathsf{diag}(\tau)]\tau. \tag{C.2}$$

C.2 Formulas for Calculating Packet Error Rates

The bit error rate P_{ber} is the percentage of bits dropped per transmission of a bit, it depends on the signal to noise ratio i.e. $P_{ber} = f(\texttt{snr})$. The packet error rate P_{per} is the percentage of packets dropped per transmission attempt. If no coding scheme is used to correct for bit errors in a packet, which is typically done, and if we assume the bit errors are independent. Then the packet error rate is

$$P_{per} = 1 - (1 - P_{ber})^{n_p} \tag{C.3}$$

in which n_p is the number of bits in a packet. Therefore, we focus our discussion on how to obtain an estimate of the bit error rate.

The bit error rate is dependent on the received signal power P_r, the thermal noise power P_{tn}, the "noise" factor associated with the receiver F, and the average noise from every interferer P_{nI}. The bit error rate is typically calculated using the signal to noise ratio $\texttt{snr}$ or the signal to interferer noise ratio $\texttt{sinr}$.

$$\texttt{snr} = \frac{P_r}{F P_{tn}} \tag{C.4}$$

Typically, the $\texttt{snr}$ does not explicitly include the noise factor F associated with the receiver electronics (i.e. $F = 1$) however, unlike interference noise it has a multiplicative effect on reducing the effective $\texttt{snr}$.

$$\texttt{sinr} = \frac{P_r}{F(P_{tn} + P_{nI})} \tag{C.5}$$

Typically, the power is expressed in dBm in which 0 dBm represents 1 mW of power, and the signal to noise ratio and noise factor can be expressed in dB. Therefore we assume power is expressed in mW and denote

$$\mathtt{SNR} = 10\log_{10}\frac{P_r}{FP_{tn}} = P_R - NF - P_{TN} \tag{C.6}$$

in which $P_R = 10\log_{1o}(P_r)$dBm, $NF = 10\log_{1o}(F)$dB (denoted as the noise figure), and $P_{TN} = 10\log_{1o}(P_{tn})$dBm. When $F = 1,\ NF = 0$, we denote the corresponding $\mathtt{SNR}$ as $\mathtt{SNR}_1$. Similarly we denote

$$\mathtt{SINR} = 10\log_{10}\frac{P_r}{F(P_{tn}+P_{nI})} = P_R - NF - P_{TN+I} \tag{C.7}$$

in which $P_{TN+I} = 10\log_{1o}(P_{tn}+P_{nI})$dBm.

As the name suggests, the thermal noise power is dependent on the temperature of the environment T (Kelvin), and the bandwidth B (Hz) of the receiver. Denoting Boltzmann's constant as k in which $k = -198.6$ (dBm/degreesK-Hz), we can calculate the thermal noise power as

$$P_{TN} = -198.6 + 10\log_{10}T + 10\log_{10}B \text{ dBm.} \tag{C.8}$$

The received power is dependent on the transmitted power P_t, the receiver and transmitter antenna gains G_r, G_t, the distance between the receiver and transmitter d, the average unobstructed distance d_o, the path loss exponent n, and the wavelength of the transmission λ, $\lambda = \frac{c}{f}$. In which c is the velocity of light (3.0×10^8 m/s) and f is the carrier frequency. These relationships are described by the following formula:

$$P_r = \begin{cases} P_tG_tG_r(\frac{\lambda}{4\pi d})^2, \text{ if } d \le d_o; \\ P_tG_tG_r(\frac{\lambda}{4\pi d_o})^2(\frac{d}{d_o})^n, \text{ otherwise;} \end{cases} \tag{C.9}$$

which is the Friis free-space equation when $n = 2$, when transmissions occur with obstructions it is generally higher [99]. In terms of dB

$$P_R = \begin{cases} P_T + G_T + G_R - 20\log_{10}\frac{4\pi d}{\lambda}, \text{ if } d \leq d_o; \\ P_T + G_T + G_R - 20\log_{10}\frac{4\pi d_o}{\lambda} - 10n\log_{10}\frac{d}{d_o}, \text{ otherwise;} \end{cases} \tag{C.10}$$

in which $G_T = 10\log_{10}(G_t)$ and $G_R = 10\log_{10}(G_r)$. Which, in terms of f given in MHz f_{MHz} the received power can be written as

$$P_R = \begin{cases} P_T + G_T + G_R + 27.6 - 20\log_{10} f_{MHz} - 20\log_{10} d, \text{ if } d \leq d_o; \\ P_T + G_T + G_R + 27.6 - 20\log_{10} f_{MHz} - 20\log_{10} d_o - 10n\log_{10}\frac{d}{d_o}, \text{ otherwise;} \end{cases} \tag{C.11}$$

With the path loss defined as $P_L(d) = P_T - P_R$, it will have the following form:

$$P_L(d) = \begin{cases} -G_T - G_R - 27.6 + 20\log_{10} f_{MHz} + 20\log_{10} d, \text{ if } d \leq d_o; \\ -G_T - G_R - 27.6 + 20\log_{10} f_{MHz} + 20\log_{10} d_o + 10n\log_{10}\frac{d}{d_o}, \text{ otherwise;} \end{cases} \tag{C.12}$$

which agrees with [99, (16)] for the case when $d > d_o = 1$ meter and $G_T = G_R = 0.0$ dB. Table C.2 summarizes our discussion thus far.

The table lists a new term known as the receiver sensitivity S [99], which is typically provided by the manufacturer of the digital radio. The additional term $\texttt{SNR}_{min}$ denotes the minimum acceptable SNR such that the bit error rate (or packet error rate) achieves some minimal acceptable level.

TABLE C.2

KEY TERMS AND FORMULAS TO DETERMINE RECEIVED SNR.

Term	Symbol	Formula
receiver antenna gain (dB)	G_R	$10\log_{10}(G_r)$
transmitter antenna gain (dB)	G_T	$10\log_{10}(G_t)$
path loss (dB)	$P_L(d)$	(see. (C.12))
transmitted power (dBm)	P_T	$10\log_{10}(P_t)$
received power (dBm)	P_R	$P_T - P_L(d)$
thermal noise power (dBm)	P_{TN}	$-198.6 + 10\log_{10} T + 10\log_{10} B$
min. accept. signal to noise ratio (dB)	$\mathtt{SNR}_{min}$	
receiver sensitivity (dBm)	S	$SNR_{min} + P_{TN} + NF$
receiver noise figure (dB)	NF	$10\log_{10}(F)$
signal to noise ratio (dB)	$\mathtt{SNR}$	$P_T - P_L(d) - NF - P_{TN}$

C.2.1 Packet Error Rates for 802.15.4 As a Function of SNR and Transmission Distance d.

The 802.15.4 standard specifies the physical layer and the *MAC* layers for a wireless network [40]. The physical layer specifies the type of modulation and carrier frequencies which will be used to transmit data over a network. In particular, we will concern ourselves with the radio which uses a carrier frequency around 2450 MHz and each channel has a corresponding bandwidth of 2 MHz. The modulation uses a four bit offset quadrature phase-shift keying (O-QPSK), in which each of the 16 data symbols is mapped to a corresponding 32-chip pseudo-random noise (PN) sequence [40, pp. 45-47]. As such a significant, coding gain is achieved in transmitting individual bits. The

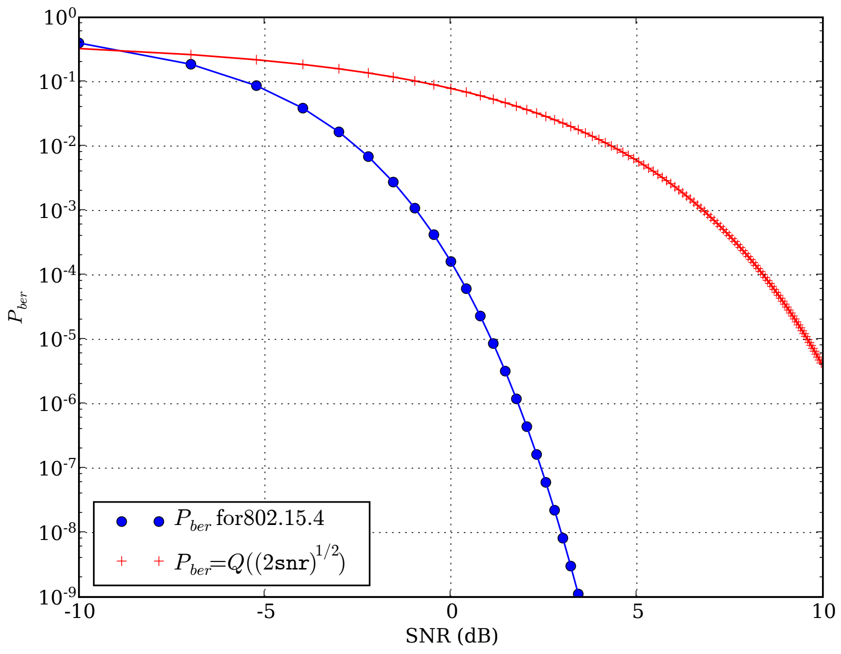

Figure C.1. Bit error rate as a function of SNR.

following formula in [40, p. 643] can be used to relate the P_{ber} as a function of the $\mathtt{snr}$

$$P_{ber}(\mathtt{snr}) = \frac{8}{15}\frac{1}{16}\sum_{k=2}^{16} -1^k \binom{16}{k} e^{(20\times\mathtt{snr}\times(\frac{1}{k}-1))}. \tag{C.13}$$

The corresponding $P_{ber}(\mathtt{SNR})$ is plotted in Figure C.1. Figure C.1 also includes the corresponding curve in which

$$P_{ber}(\mathtt{snr}) = Q(\sqrt{2\mathtt{snr}}),\ Q(x) = \frac{1}{2\pi}\int_x^\infty e^{-\frac{u^2}{2}} du. \tag{C.14}$$

Figure C.1 shows that (C.14) which was used in [111] to model the 802.15.4 bit error rate, fails to capture the significant coding gain for the respective modulation scheme.

The CC2420 is a true single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low power and low voltage wireless applications [48]. The published sensitivity is typically -95 dBm with a minimum allowable of -90 dBm. The sensitivity corresponds to a received power level in which the $P_{per} < 1\%$ for a 20 byte length packet ($n_p = 160$ bits). In order to achieve such a low P_{ber}, the $P_{ber} < [1 - (1 - P_{per})^{\frac{1}{n_p}}] = 6.28 \times 10^{-5}$ which corresponds to a $\mathtt{SNR}_{min} > 0.41$ dB. As such the noise figure NF is typically 15.44 dB which can be as high as 20.44 dB for some transceivers. With this information, and noting that each channel has a bandwidth of 2.0 MHz, we can generate the following figures which show P_{per} as a function of transmission distance: Figure C.2, Figure C.3, Figure C.4, and Figure C.5. As to be expected the free space transmission distances were longer for the same P_{per}, what was a bit surprising was the significant loss in transmission distance when the sensitivity was increased from -95 dBm to -90 dBm. We chose three different packet lengths 120 bits ($n_h + n_{fcs} = 120,\ n_d = 0$ bits), 248 bits ($n_d = 128$ bits), and 1080 bits ($n_d = 960$ bits). The largest amount equals the most data you can fit in the receive and transmit buffers for the CC2420.

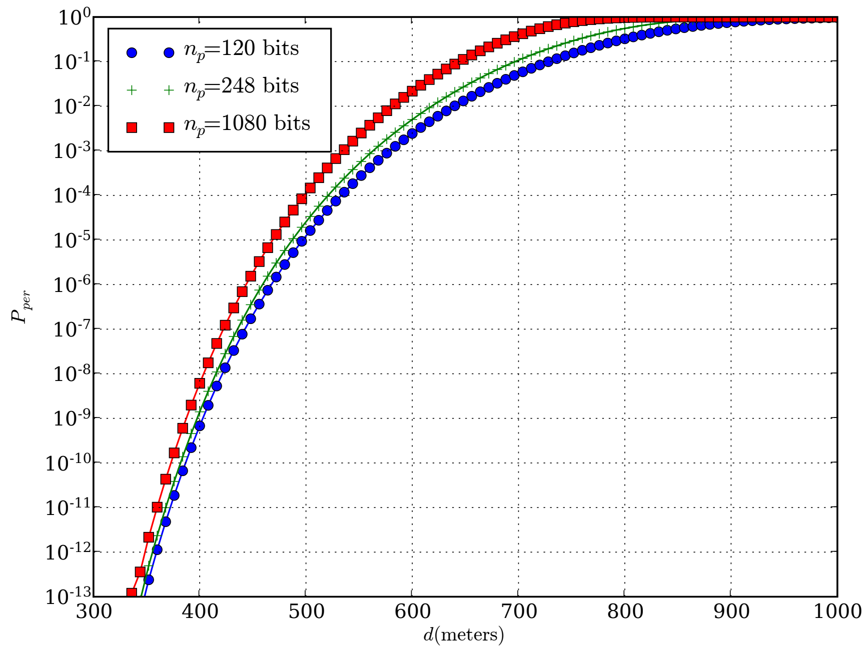

Figure C.2. Packet error rate ($P_T = 0$ dBm, $S = -95$ dBm) as a function of transmission distance d, packet length n_p in free space $n = 2$.

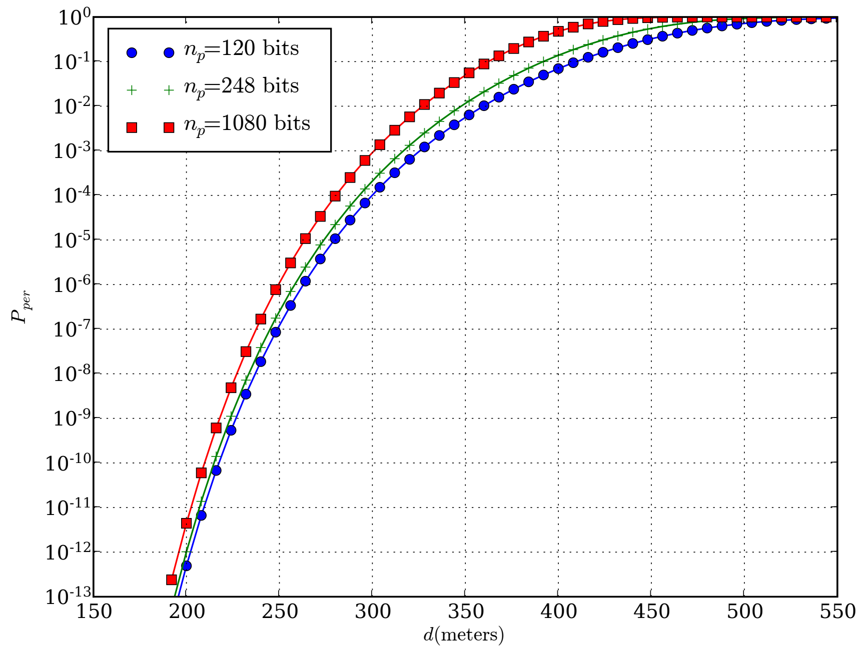

Figure C.3. Packet error rate ($P_T = 0$ dBm, $S = -90$ dBm) as a function of transmission distance d, packet length n_p in free space $n = 2$.

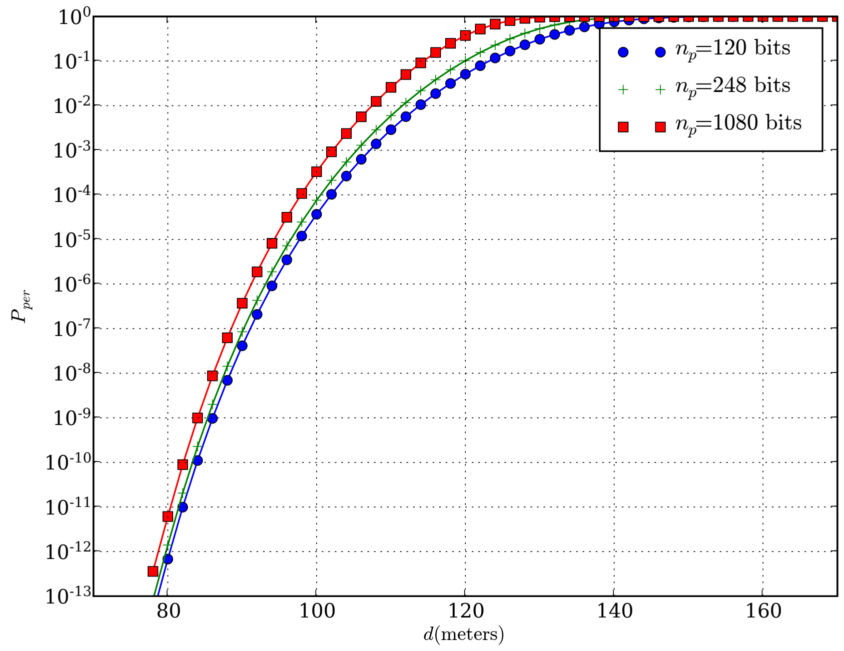

Figure C.4. Packet error rate ($P_T = 0$ dBm, $S = -95$ dBm) as a function of transmission distance d, packet length n_p not in free space $n = 3.3$, $d_o = 8$ meters.

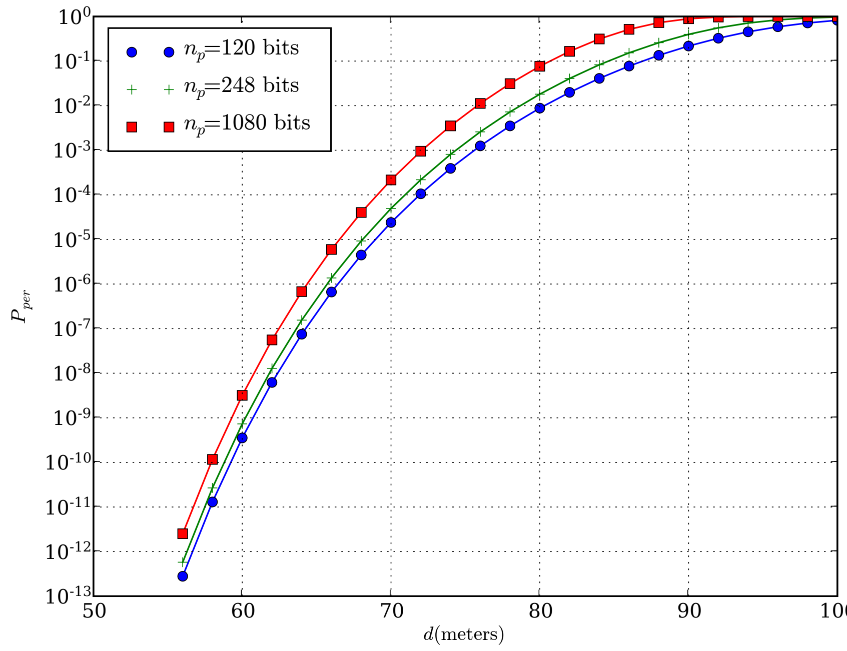

Figure C.5. Packet error rate ($P_T = 0$ dBm, $S = -90$ dBm) as a function of transmission distance d, packet length n_p not in free space $n = 3.3$, $d_o = 8$ meters.

BIBLIOGRAPHY

1. R. J. Anderson and M. W. Spong. Bilateral control of teleoperators with time delay. *Proceedings of the IEEE Conference on Decision and Control Including The Symposium on Adaptive Processes*, pages 167 – 173, 1988. ISSN 0191-2216. URL `http://dx.doi.org/10.1109/CDC.1988.194290`.

2. P. Antsaklis and J. Baillieul, editors. *Special Issue on Networked Control Systems*, volume 49 number 9 of *IEEE Transactions on Automatic Control*. IEEE, 2004.

3. P. Antsaklis and J. Baillieul, editors. *Special Issue: Technology of Networked Control Systems*, volume 95 number 1 of *Proceedings of the IEEE*. IEEE, 2007.

4. P. J. Antsaklis and A. Michel. *Linear Systems*. McGraw-Hill Higher Education, 1997. ISBN 0070414335.

5. Atmel. *ATmega128(L) Users Manual*. http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf.

6. S. Basu. *FreeMat*. http://freemat.sourceforge.net/.

7. J. S. Bay. *Fundamentals of linear state space systems*. WCB/McGraw-Hill, Boston, MA, USA, 1999. ISBN 0-256-24639-4.

8. D. M. Beazley. *Simplified Wrapper and Interface Generator*. http://www.swig.org/.

9. A. Behzad and I. Rubin. Multiple access protocol for power-controlled wireless access nets. *IEEE Transactions on Mobile Computing*, 3(4):307 – 316, 2004. ISSN 1536-1233. URL `http://dx.doi.org/10.1109/TMC.2004.44`.

10. P. Berestesky, N. Chopra, and M. Spong. Theory and experiments in bilateral teleoperation over the internet. In *Proceedings of the 2004 IEEE International Conference on Control Applications (IEEE Cat. No.04CH37596)*, volume Vol.1, pages 456 – 63, Taipei, Taiwan, 2004.

11. P. Berestesky, N. Chopra, and M. W. Spong. Discrete time passivity in bilateral teleoperation over the internet. *Proceedings - IEEE International Conference on Robotics and Automation*, 2004(5):4557 – 4564, 2004. ISSN 1050-4729.

12. Berkeley WEBS: Wireless Embedded Systems. *TinyOS Community Forum: Related*. http://www.tinyos.net/related.html.

13. B. Bernhardsson, J. Eker, and J. Persson. Bluetooth in control. In *Handbook of Networked and Embedded Control Systems*, pages 699–720. 2005.

14. D. S. Bernstein. *Matrix Mathematics*. Princeton University Press, 2005. ISBN 0691118027.

15. D. S. Bernstein and W. M. Haddad. Nonlinear controllers for positive real systems with arbitrary input nonlinearities. *IEEE Transactions on Automatic Control*, 39 (7):1513 – 1517, 1994. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/9.299647`.

16. J. D. Birdwell, D. A. Castanon, and M. Athans. On reliable control system designs. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-16(5):703 – 711, 1986. ISSN 0018-9472.

17. P. C. Breedveld. Port-based modeling of dynamic systems in terms of bond graphs. In I. Troch, editor, *5th Vienna Symposium on Mathematical Modelling, Vienna*, volume ARGESIM Report no. 30, page cd rom, Vienna, February 2006. ARGESIM and ASIM, Arbeitsgemeinschaft Simulation. ISBN 3 901608 30 3.

18. M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan. Control of large-scale irrigation networks. *Proceedings of the IEEE*, 95(1):75 – 91, 2007. ISSN 0018-9219. URL `http://dx.doi.org/10.1109/JPROC.2006.887289`.

19. J. L. Chen and L. Lee. Passivity approach to feedback connection stability for discrete-time descriptor systems. *Proceedings of the IEEE Conference on Decision and Control*, 3:2865 – 2866, 2001. ISSN 0191-2216. URL `http://dx.doi.org/10.1109/CDC.2001.980710`.

20. R. Costa-Castelló and E. Fossas. On preserving passivity in sampled-data linear systems. *2006 American Control Conference (IEEE Cat. No. 06CH37776C)*, pages 6 pp. –, 2006.

21. R. Costa-Castelló and E. Fossas. On preserving passivity in sampled-data linear systems by using state observers. *2007 European Control Conference*, pages 3350–3355, 2007.

22. T. Cover and A. E. Gamal. Capacity theorems for the relay channel. *Information Theory, IEEE Transactions on*, 25(5):572–584, 1979. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1056084`.

23. Crossbow Technology, Inc. *MPR-MIB Series Users Manual*. http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf.

24. R. D'Andrea and G. E. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9):1478 – 1495, 2003. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2003.816954`.

25. C. A. Desoer and M. Vidyasagar. *Feedback Systems: Input-Output Properties*. Academic Press, Inc., Orlando, FL, USA, 1975. ISBN 0122120507.

26. M. Dubberley, A. M. Agogino, and A. Horvath. Life-cycle assessment of an intelligent lighting system using a distributed wireless mote network. *IEEE International Symposium on Electronics and the Environment*, pages 122 – 127, 2004. URL `http://dx.doi.org/10.1109/ISEE.2004.1299700`.

27. M. Ergen, D. Lee, R. Sengupta, and P. Varaiya. Wtrp - wireless token ring protocol. *IEEE Transactions on Vehicular Technology*, 53(6):1863 – 81, 2004/11/. ISSN 0018-9545. URL `http://dx.doi.org/10.1109/TVT.2004.836928`.

28. X. Fan, K. Chandrayana, M. Arcak, S. Kalyanaraman, and J. T. Wen. A two-time scale design for detection and rectification of uncooperative network flows. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 1842–1847, December 12-15 2005.

29. L. Fang and P. J. Antsaklis. Information consensus of asynchronous discrete-time multi-agent systems. *Proceedings of the American Control Conference*, 3:1883 – 1888, 2005. ISSN 0743-1619.

30. J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465 – 1476, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2004.834433`.

31. X. Feng, K. A. Loparo, Y. Ji, and H. J. Chizeck. Stochastic stability properties of jump linear systems. *IEEE Transactions on Automatic Control*, 37(1):38 – 53, 1992. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/9.109637`.

32. A. Fettweis. Wave digital filters: theory and practice. *Proceedings of the IEEE*, 74(2):270 – 327, 1986/02/. ISSN 0018-9219.

33. O. Flärdh, K. H. Johansson, and M. Johansson. A new feedback control mechanism for error correction in packet-switched networks. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 488–493, December 12-15 2005.

34. G. F. Franklin, J. D. Powell, and M. Workman. *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., 1998. ISBN 0201820544.

35. J. Friedman, D. Lee, I. Tsigkogiannis, S. Wong, D. Chao, D. Levin, W. Kaiser, and M. Srivastava. RAGOBOT: A new platform for wireless mobile sensor networks. In *Distributed Computing in Sensor Systems: First IEEE International Conference: DCOSS 2005*, page 412. Springer-Verlag GmbH, July 2005. ISBN 3-540-26422-1.

36. FSF. *Host/Target specific installation notes for GCC*, . http://gcc.gnu.org/install/specific.html#avr.

37. FSF. *GNU Binutils*, . http://www.gnu.org/software/binutils/.

38. G. Golo, A. J. van der Schaft, P. Breedveld, and B. Maschke. Hamiltonian formulation of bond graphs. In *Nonlinear and Hybrid Systems in Automotive Control*, pages 351–372. Springer-Verlag, London, UK, 2003.

39. G. C. Goodwin, H. Haimovich, D. E. Quevedo, and J. S. Welsh. A moving horizon approach to networked control system design. *IEEE Transactions on Automatic Control*, 49(9):1427 – 1445, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2004.834132`.

40. I. P. W. Group. *802.15.4 IEEE Standard for Information technology – Telecommunications and information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, October 2003.

41. W. Haddad and V. Chellaboina. Nonlinear control of hammerstein systems with passive nonlinear dynamics. *IEEE Transactions on Automatic Control*, 46(10): 1630 – 1634, 2001. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/9.956062`.

42. W. M. Haddad and V. Chellaboina. Nonlinear controllers for nonlinear passive systems with arbitrary input nonlinearities. *Proceedings of the American Control*

Conference, 6:4149 – 4153, 2000. ISSN 0743-1619. URL http://dx.doi.org/10.1109/ACC.2000.877002.

43. O. Hajek. *Control theory in the plane*. Springer-Verlag New York, Inc., New York, NY, USA, 1991. ISBN 0-387-53553-5.

44. C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 163–176, New York, NY, USA, 2005. ACM Press. ISBN 1-931971-31-5. doi: http://doi.acm.org/10.1145/1067170.1067188.

45. J. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138 – 62, 2007. ISSN 0018-9219. URL http://dx.doi.org/10.1109/JPROC.2006.887288.

46. T. Hu and Z. Lin. *Control Systems with Actuator Saturation: Analysis and Design*. Birkhäuser Boston, 2001. ISBN 0-8176-4219-6.

47. J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-879-2. doi: http://doi.acm.org/10.1145/1031495.1031506.

48. T. Instruments. *CC2420: 2.4 GHz IEEE 802.15.4/ ZigBee-ready RF Transceiver - Data Manual (SWRS041B)*, 2007.

49. Y. Ji and H. Chizeck. Jump linear quadratic gaussian control. steady-state solution and testable conditions. *Control - Theory and Advanced Technology*, 6(3):289 – 319, 1990. ISSN 0911-0704.

50. Y. Ji, H. Chizeck, X. Feng, and K. Loparo. Stability and control of discrete-time jump linear systems. *Control - Theory and Advanced Technology*, 7(2):247 – 270, 1991. ISSN 0911-0704.

51. F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237 – 252, 1998. ISSN 0160-5682. URL http://dx.doi.org/10.1038/sj.jors.2600523.

52. J. Kepner. *Parallel Programming with MatlabMPI*. http://www.ll.mit.edu/MatlabMPI/.

53. H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, Inc., 1996. ISBN 0132280248.

54. N. Kottenstette and P. J. Antsaklis. neclab: The network embedded control lab. In *Workshop on Networked Embedded Sensing and Control*, Lecture Notes in Control and Information Sciences. Springer, October 2005.

55. F. KOZIN. Survey of stability of stochastic systems. *Automatica*, 5(1):95 – 112, 1969.

56. M. Kuschel, P. Kremer, S. Hirche, and M. Buss. Lossy data reduction methods for haptic telepresence systems. In *Proceedings. 2006 Conference on International Robotics and Automation (IEEE Cat. No. 06CH37729D)*, pages 2933 – 8, Orlando, FL, USA, 2006.

57. I. D. Landau, R. Lozano, and M. M'Saad. *Adaptive Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998. ISBN 354076187X.

58. K. K. Lee and S. T. Chanson. Packet loss probability for bursty wireless real-time traffic through delay model. *IEEE Transactions on Vehicular Technology*, 53(3): 929 – 938, 2004. ISSN 0018-9545. URL `http://dx.doi.org/10.1109/TVT.2004.825769`.

59. L. Lee and J. L. Chen. Strictly positive real lemma for discrete-time descriptor systems. *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, vol.4:3666 – 7, 2000//. URL `http://dx.doi.org/10.1109/CDC.2000.912277`.

60. L. Lee and J. L. Chen. Strictly positive real lemma and absolute stability for discrete-time descriptor systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 50(6):788 – 94, 2003/06/. ISSN 1057-7122. URL `http://dx.doi.org/10.1109/TCSI.2003.812617`.

61. N. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. Fratantoni, and R. Davis. Collective motion, sensor networks, and ocean sampling. *Proceedings of the IEEE*, 95(1):48 – 74, 2007. ISSN 0018-9219. URL `http://dx.doi.org/10.1109/JPROC.2006.887295`.

62. F. L. Lewis. Optimal control. In W. S. Levine, editor, *The Control Handbook*, pages 759–778. CRC Press, Boca Raton, FL, 1996.

63. K. Li and J. Baillieul. Robust quantization for digital finite communication bandwidth (dfcb) control. *IEEE Transactions on Automatic Control*, 49(9):1573 – 1584, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2004.834106`.

64. J. Linares-Flores and H. Sira-Ramirez. Dc motor velocity control through a dc-to-dc power converter. *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, Vol.5:5297 – 302, 2004//.

65. J. Linares-Flores, J. Reger, and H. Sira-Ramirez. A time-varying linear state feedback tracking controller for a boost-converter driven dc motor. *Mechatronics 2006. 4th IFAC Symposium on Mechatronic Systems. Preprints*, pages 926 – 31, 2006//.

66. Q. Ling. *Stability and Performance of Control Systems with Limited Feedback Information*. PhD thesis, University of Notre Dame, May 2005.

67. X. Liu and A. Goldsmith. Wireless medium access control in networked control systems. *Proceedings of the American Control Conference*, 4:3605 – 3610, 2004. ISSN 0743-1619.

68. X. Liu and A. J. Goldsmith. Cross-layer design of distributed control over wireless networks. In T. Basar, editor, *Systems and Control: Foundations and Applications*. Birkhauser, 2005.

69. X. Liu and M. Haenggi. Throughput analysis of fading sensor networks with regular and random topologies. *Eurasip Journal on Wireless Communications and Networking*, 2005(4):554 – 564, 2005. ISSN 1687-1472. URL `http://dx.doi.org/10.1155/WCN.2005.554`.

70. D. Lymberopoulos and A. Savvides. XYZ: A motion-enabled, power aware sensor node platform for distributed sensor network applications. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, April 2005. http://www.eng.yale.edu/enalab/publications/spots05_XYZ.pdf.

71. D. Marco, E. Duarte-Melo, M. Liu, and D. Neuhoff. the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data, 2003. URL `citeseer.ist.psu.edu/marco03manytoone.html`.

72. I. Maric and R. D. Yates. Cooperative multicast for maximum network lifetime. *IEEE Journal on Selected Areas in Communications*, 23(1):127 – 135, 2005. ISSN 0733-8716. URL `http://dx.doi.org/10.1109/JSAC.2004.837343`.

73. H. J. Marquez. *Nonlinear Control Systems: Analysis and Design*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2003.

74. Mathworks. *MATLAB*. http://www.mathworks.com/.

75. T. Matiakis, S. Hirche, and M. Buss. The scattering transformation for networked control systems. *2005 IEEE International Conference on Control Applications (CCA) (IEEE Cat. No.05CH37713)*, vol.1:705 – 10, 2005//.

76. T. Matiakis, S. Hirche, and M. Buss. Independent-of-delay stability of nonlinear networked control systems by scattering transformation. *2006 American Control Conference (IEEE Cat. No. 06CH37776C)*, pages 6 pp. –, 2006//.

77. A. S. Matveev and A. V. Savkin. Decentralized stabilization of linear systems via limited capacity communication networks. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 1155–1161, December 12-15 2005.

78. S. Mazumder, K. Acharya, and M. Tahir. ”wireless” control of spatially distributed power electronics. *Conference Proceedings - IEEE Applied Power Electronics Conference and Exposition - APEC*, 1:75 – 81, 2005. URL `http://dx.doi.org/10.1109/APEC.2005.1452889`.

79. M. B. McMickell, B. Goodwine, and L. A. Montestruque. MICAbot: A robotic platform for large-scale distributed robotics. In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, volume 2, pages 1600–1605, September 14-19 2003.

80. A. Megretski. 11.3 l2 gain of transfer matrix models. page 5, 2006. URL `http://web.mit.edu/6.241/www/images/L06L2gain.pdf`.

81. M. Michalkiewicz and M. Manterola. *AVR In-System Programmer*. http://savannah.nongnu.org/projects/uisp/.

82. M. Michalkiewicz, J. Wunsch, and E. Weddington. *AVR C Runtime Library*. http://savannah.nongnu.org/projects/avr-libc/.

83. L. A. Montestruque. *Model-Based Networked Control Systems*. PhD thesis, University of Notre Dame, November 2004.

84. L. A. Montestruque and P. Antsaklis. Stability of model-based networked control systems with time-varying transmission times. *IEEE Transactions on Automatic Control*, 49(9):1562 – 1572, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2004.834107`.

85. Y. Mostofi and R. M. Murray. On dropping noisy packets in kalman filtering over a wireless fading channel. *Proceedings of the American Control Conference*, 3: 4569–4600, 2005. ISSN 0743-1619.

86. Y. Mostofi and R. M. Murray. Receiver design principles for estimation over fading channels. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 464–469, December 12-15 2005.

87. G. Nair, F. Fagnani, S. Zampieri, and R. Evans. Feedback control under data rate constraints: an overview. *Proceedings of the IEEE*, 95(1):108 – 37, 2007. ISSN 0018-9219. URL `http://dx.doi.org/10.1109/JPROC.2006.887294`.

88. M. Nakamura, A. Sakurai, and S. H. B. Furubo. Collaborative processing in sensor/actuator networks for environment control. In *Proceedings of the 2005 IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 163–168, December 5-8 2005.

89. G. Niemeyer. *Using Wave Variables in Time Delayed Force Reflecting Teleoperation*. PhD thesis, Massachusetts Institute of Technology, September 1996.

90. G. Niemeyer and J.-J. E. Slotine. Telemanipulation with time delays. *International Journal of Robotics Research*, 23(9):873 – 890, 2004. ISSN 0278-3649. URL `http://dx.doi.org/10.1177/0278364904045563`.

91. G. Niemeyer and J.-J. E. Slotine. Towards force-reflecting teleoperation over the internet. *Proceedings - IEEE International Conference on Robotics and Automation*, 3:1909 – 1915, 1998. ISSN 1050-4729. URL `http://dx.doi.org/10.1109/ROBOT.1998.680592`.

92. U. of Notre Dame. A source of life and death, solving the combined sewer overflow problem. *SIGNATURES, Engineering Advances at the University of Notre Dame*, 6(2):12 – 19, 2005.

93. R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520 – 1533, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2004.834113`.

94. R. Ortega, A. Loria, P. J. Nicklasson, and H. Sira-Ramirez. *Passivity-Based Control of Euler-Lagrange Systems*. Springer-Verlag London Limited, Great Britain, 1998. ISBN 1852330163.

95. J. Ousterhout. *Tcl/Tk*. http://www.tcl.tk/software/tcltk/.

96. C. Papadopoulos and J. Heidemann. Using ns in the classroom and lab. In *Proceedings of the ACM SIGCOMM Workshop on Computer Networking: Curriculum Designs and Educational Challenges*, pages 45–46, Pittsburgh, PA, USA, August 2002. ACM. URL `http://www.isi.edu/~johnh/PAPERS/Papadopoulos02a.html`.

97. X. Qi, M. V. Salapaka, P. G. Voulgaris, and M. Khammash. Structured optimal and robust control with multiple criteria: A convex solution. *IEEE Transactions*

on Automatic Control, 49(10):1623 – 1640, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2004.835357`.

98. Quanser. *WinCon 5.0*. http://www.quanser.com/english/html/solutions/fs_soln_software_wincon.html.

99. S. Rao. Estimating the zigbee transmision-range ism band. *EDN*, pages 67 – 72, May 24, 2007. URL `http://www.edn.com/article/CA6442439.html`.

100. J. Rosen and B. Hannaford. A doc at a distance [remotely-controlled surgical robots]. *IEEE Spectrum*, 43(10):34 – 9, 2006/10/. ISSN 0018-9235. URL `http://dx.doi.org/10.1109/MSPEC.2006.1705774`.

101. T. P. Ruggaber and J. W. Talley. Detection and control of combined sewer overflow events using embedded sensor network technology. *Proceedings - ASCE World Water and Environmental Resources Congress*, 2005.

102. J.-H. Ryu, D.-S. Kwon, and B. Hannaford. Stable teleoperation with time-domain passivity control. *IEEE Transactions on Robotics and Automation*, 20(2):365 – 73, 2004/04/. ISSN 1042-296X. URL `http://dx.doi.org/10.1109/TRA.2004.824689`.

103. J.-H. Ryu, Y. S. Kim, and B. Hannaford. Sampled- and continuous-time passivity and stability of virtual environments. *IEEE Transactions on Robotics*, 20(4):772 – 6, 2004/08/. ISSN 1042-296X. URL `http://dx.doi.org/10.1109/TRO.2004.829453`.

104. T. Samad, J. Bay, and D. Godbole. Network-centric systems for military operations in urban terrain: the role of uavs. *Proceedings of the IEEE*, 95(1):92 – 107, 2007. ISSN 0018-9219. URL `http://dx.doi.org/10.1109/JPROC.2006.887327`.

105. J. S. Sandhu, A. M. Agogino, and A. K. Agogino. Wireless sensor networks for commercial lighting control: Decision making with multi-agent systems. In *Proceedings of the AAAI-04 Workshop on Sensor Networks*, pages 88–92, 2004.

106. S. V. Sarma and M. A. Dahleh. Synthesis of simple feed-forward networks: a first-order example. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 1131–1136, December 12-15 2005.

107. C. Secchi, S. Stramigioli, and C. Fantuzzi. Digital passive geometric telemanipulation. *Proceedings - IEEE International Conference on Robotics and Automation*, 3:3290 – 3295, 2003. ISSN 1050-4729. URL `http://dx.doi.org/10.1109/ROBOT.2003.1242098`.

108. P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. *Proceedings of the American Control Conference*, 2:1491 – 1496, 2001. ISSN 0743-1619. URL `http://dx.doi.org/10.1109/ACC.2001.945935`.

109. P. Seiler and R. Sengupta. A bounded real lemma for jump systems. *IEEE Transactions on Automatic Control*, 48(9):1651 – 1654, 2003. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2003.817010`.

110. P. Seiler and R. Sengupta. An h infinity approach to networked control. *IEEE Transactions on Automatic Control*, 50(3):356 – 364, 2005. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2005.844177`.

111. S. Y. Shin, S. Choi, H. S. Park, and W. H. Kwon. Lecture notes in computer science: Packet error rate analysis of ieee 802.15.4 under ieee 802.11b interference. In *Lecture Notes in Computer Science*, volume 3510, pages 279 – 288, Xanthi, Greece, 2005.

112. J. J. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

113. J. L. Snell. Finite markov chains and their applications. *The American Mathematical Monthly*, 66(2):99–104, February 1959.

114. E. D. Sontag. Algebraic approach to bounded controllability of linear systems. *International Journal of Control*, 39(1):181 – 188, 1984. ISSN 0020-7179.

115. N. Spurrier. *Pexpect – a Pure Python Expect-like module*. http://pexpect.sourceforge.net/.

116. T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical Report CENS-TR-30, University of California, Los Angeles, Center for Embedded Networked Computing, November 2003. URL `http://www.isi.edu/~johnh/PAPERS/Stathopoulos03b.html`.

117. S. Stramigioli. *Modeling and IPC Control of Interactive Mechanical Systems: A Coordinate-free Approach*. Springer-Verlag London Limited, Great Britain, 2001. ISBN 1852333952.

118. S. Stramigioli, C. Secchi, A. van der Schaft, and C. Fantuzzi. A novel theory for sampled data system passivity. *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (Cat. No.02CH37332C)*, vol.2:1936 – 41, 2002//. URL `http://dx.doi.org/10.1109/IRDS.2002.1044039`.

119. S. Stramigioli, C. Secchi, A. J. van der Schaft, and C. Fantuzzi. Sampled data systems passivity and discrete port-hamiltonian systems. *IEEE Transactions on Robotics*, 21(4):574 – 587, 2005. ISSN 1552-3098. URL `http://dx.doi.org/10.1109/TRO.2004.842330`.

120. A. Suri, J. Baillieul, and D. V. Raghunathan. Control using feedback over wireless ethernet and bluetooth. In *Handbook of Networked and Embedded Control Systems*, pages 677–698. 2005.

121. N. A. Tanner and G. Niemeyer. Practical limitations of wave variable controllers in teleoperation. *2004 IEEE Conference on Robotics, Automation and Mechatronics*, pages 25 – 30, 2004.

122. G. Tao and P. Ioannou. Necessary and sufficient conditions for strictly positive real matrices. *IEE Proceedings G (Circuits, Devices and Systems)*, 137(5):360 – 6, 1990/10/. ISSN 0956-3768.

123. S. Tatikonda, A. Sahai, and S. Mitter. Stochastic linear control over a communication channel. *IEEE Transactions on Automatic Control*, 49(9):1549 – 1561, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2004.834430`.

124. UCLA NESL. *SOS Operating System*. http://nesl.ee.ucla.edu/projects/sos/.

125. University of Notre Dame Department of Electrical Engineering. *Ball and Beam Balancing Problem*, . http://www.nd.edu/˜eeuglabs/ee455/lab/lab4/lab4_2003.pdf.

126. University of Notre Dame Department of Electrical Engineering. *Model identification of Ball and Beam Plant*, . http://www.nd.edu/˜eeuglabs/ee455/lab/lab3/lab3_2003.pdf.

127. A. van der Schaft. *L2-Gain and Passivity in Nonlinear Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999. ISBN 1852330732.

128. G. van Rossum. *Python Programming Language*. http://www.python.org/.

129. L. Wall, J. Orwant, and T. Christiansen. *Programming Perl*, chapter 1. O'Reilly, 3rd edition, July 2000.

130. J. T. Wen and M. Arcak. A unifying passivity framework for network flow control. *IEEE Transactions on Automatic Control*, 49(2):162 – 174, 2004. ISSN 0018-9286. URL `http://dx.doi.org/10.1109/TAC.2003.822858`.

131. G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, April 2005. http://www.eecs.harvard.edu/˜mdw/papers/motelab-spots05.pdf.

132. R. C. Whaley and A. Petitet. Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, February 2005. http://www.cs.utsa.edu/˜whaley/papers/spercw04.ps.

133. J. Willems. Mechanism for the stability and instability in feedback systems. *Proceedings of the IEEE*, 64(1):24 – 35, 1976/01/. ISSN 0018-9219.

134. X. Wu and C. G. Cassandras. A maximum time optimal control approach to routing in sensor networks. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 1137–1142, 2005.

135. L. Xie, M. Fu, and H. Li. Passivity analysis and passification for uncertain signal processing systems. *IEEE Transactions on Signal Processing*, 46(9):2394 – 403, Sept. 1998. ISSN 1053-587X. URL `http://dx.doi.org/10.1109/78.709527`.

136. M. Xie and M. Haenggi. Delay-reliability tradeoffs in wireless networked control systems. In *Workshop on Networked Embedded Sensing and Control*, Lecture Notes in Control and Information Sciences. Springer, October 2005.

137. M. Xie and M. Haenggi. Statistical delay analysis of TDMA and ALOHA in wireless multihop networks. *submitted IEEE Transactions on Networking*, 2006.

138. M. Xie and M. Haenggi. A study of the correlations between channel and traffic statistics in multihop networks. *To appear IEEE Transactions on Vehicular Technology*, 2007.

139. M. Xie and M. Haenggi. Towards an end-to-end delay analysis of wireless multihop networks. In *submitted Ad Hoc Networks*. Elsevier, 2007.

140. V. Yodaikien. *FSMLabs Lean POSIX for RTLinux*. http://www.fsmlabs.com/fsmlabs-lean-posix-for-rtlinux.html.

141. J. Zheng and M. J. Lee. Will IEEE 802.15.4 make ubiquitous networking a reality?: A discussion on a potential low power, low bit rate standard. *IEEE Communications Magazine*, 27(6):23–29, 2004.

This document was prepared & typeset with LaTeX 2_ε, and formatted with NDdiss2_ε *classfile (v3.0[2005/07/27]) provided by Sameer Vijay.*