MANAGING INFORMATION IN NETWORKED AND MULTI-AGENT CONTROL SYSTEMS

Thesis by

Michael S. Epstein

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2007

(Submitted November 16, 2007)

Acknowledgements

I wish to extend my most sincere thanks to my advisor, Prof. Richard M. Murray. Over the course of my graduate studies Richard has been a constant source of inspiration with his passion and approach to research. Whenever I felt frustrated from a lack of motivation or progress I could count on a meeting with Richard that would leave me invigorated and excited with insights to approach whatever problem I was stuck on, or with new ideas for avenues of research to pursue. Richard gave me the freedom to explore many different topics over the years and to pursue those I found interesting. My graduate experience was a wonderfully enriching endeavor thanks in no small part to Richard.

I would like to thank the remainder of my thesis committee: Prof. Joel Burdick, Prof. K. Mani Chandy, and Dr. Doug MacMynowski. They all provided me with useful insights and help with research and in preparing this thesis.

I have many fellow graduate students and researchers to thank. First I would like to give special attention to Ling Shi. It was through a collaboration with Ling that I first was introduced to the field of Networked Control Systems, the topic of this thesis. Over the years we collaborated on several challenging and interesting problems. Working with a fellow researcher who is so positive, energetic, and intelligent made the long processes of research seem easier. Another early collaborator in the area of networked control systems from whom I learned a great deal about how to narrow down and stay focused on a research topic was Abhishek Tiwari. I had the opportunity to work with Prof. Kevin Lynch and Prof. Karl Johannsson while they were visiting Caltech. This collaboration was in the area of hierarchical consensus and appears in this thesis. I also worked with Stefano Di Cairano while he was visiting Caltech on the topic of input buffering in networked control systems, work that appears in this thesis. Other collaborators I had the pleasure of doing research with include: Stephen Waydo, Sawyer Fuller, Will Dickson, Andrew Straw, and Prof. Michael Dickinson on insect flight control; John Carson, Dr. Doug MacMynowski, and Mark Milam on applying receding horizon control techniques to the Entry Descent and Landing problem; Prof. K. Mani Chandi and Prof. John Ledyard on market-based control of the electric power network; and Dr. Zoran Nenadic on decoding neuronal spike trains. Though most of these works were outside the area of my thesis research, they were a welcome and most fascinating diversion.

Other members of the Caltech community with whom I shared many coffee breaks, lunches around campus, laughs over the mundane issues of life, and many other ups and downs of being a graduate student that I would be remiss to acknowledge: Feras Habbal, Vaughan Thomas, Jeff Hanna, Andy Fong, Mary Dunlop, Julia Braman, Zhipu Jhin, Tamer El Sayed, Sean Humbert, Ebraheem Fontaine, Tomonori Honda, Farncesco Cucci, Michael Wolf, Nick Hudson, Vijay Gupta, William Dunbar, and many others. I must also mention the wonderful assistance provided to us by the administrative staff of the Mechanical Engineering and Control and Dynamical Systems departments who helped with everything from arranging travel accommodations during conference trips to making sure we received our paychecks Gloria Bain, Chris Silva, and others.

I must certainly extended my gratitude to the American Society for Engineering Education, the United States Department of Defense, and the Air Force Office of Scientific Research for the National Defense Science and Engineering Graduate (NDSEG) Fellowship I was generously awarded to support years two through four of my graduate studies. This helped give me the freedom to pursue a course of research in topics I was interested in. I should also thank my colleagues at the Lawrence Livermore National Laboratory, where I spent some of my off-time from Caltech working on very interesting research problems: Todd Decker, Monika Witte, Randy Hill, Bill Craig, and many others. I would be remiss to not show appreciation to the controls faculty at UCLA: Professors Steve Gibson, T-C. Tsao, Robert M'Closky, and Jeff Shamma. During my undergraduate studies they were responsible for my initial interest in the topic of feedback control systems, through both classroom instruction and my involvement in their research projects. I would also like to thank Dr. Naveed Hussein, who taught my first undergraduate class in dynamics and helped me obtain an internship at Boeing Satellite Systems working in the attitude control department.

Lastly, but certainly not least, I must extended a great deal of love and appreciation to all my friends and family outside of the Caltech community who have helped me along in my life. Special thanks are in order for my parents and sisters who always pushed me to achieve. To my mother Cecile for recognizing and cultivating my interest and abilities in science and math at such an early age. To my father Jeffrey for providing an example of how hard work in school can reap rewards. To my sister Danielle for helping me choose an undergraduate major, and to my sister Amy whose own interest in Caltech kept graduate school in my mind. Finally, I would like to thank Leilanie for being such a large part of my early adult life and always having faith in my abilities and inspiring me to seek out my passions in school and life.

Abstract

Traditional feedback control systems give little attention to issues associated with the flow of information through the feedback loop. Typically implemented with dedicated communication links that deliver nearly precise, reliable, and non-delayed information, researchers have not needed to concern themselves with issues related to quantized, delayed, and even lost information. With the advent of newer technologies and application areas that pass information through non-reliable networks, these issues cannot be ignored. In recent years the field of Networked Control Systems (NCS) has emerged to describe situations where these issues are present. The research in this field focuses on quantifying performance degradations in the presence of network effects and proposing algorithms for managing the information flow to counter those negative effects. In this thesis I propose and analyze algorithms for managing information flow for several Networked Control Systems scenarios: state estimation with lossy measurement signals, using input buffers to reduce the frequency of communication with a remote plant, and performing state estimation when control signals are transmitted to a remote plant via a lossy communication link with no acknowledgement signal at the estimator. Multi-agent coordinated control systems serve as a prime example of an emerging area of feedback control systems that utilize feedback loops with information passed through possibly imperfect communication networks. In these systems, agents use a communication network to exchange information in order to achieve a desired global objective. Hence, managing the information flow has a direct impact on the performance of the system. I also explore this area by focusing on the problem of multi-agent average consensus. I propose an algorithm based on a hierarchical decomposition of the communication topology to speed up the time to convergence. For all these topics I focus on designing intuitive algorithms that intelligently manage the information flow and provide analysis and simulations to illustrate their effectiveness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Scope of Thesis

Traditionally, the areas of feedback control and communication networks are decoupled from each other, as they have many different underlying assumptions. Whereas the quality and reliability of information transmitted in a communication network is typically the primary focus, feedback control systems study the performance of manipulating a dynamical system with information that is generally assumed to be passed through ideal channels. Control engineers generally assume perfect transmission of information within the closed loop and that data processing is done with zero time delay. On the other hand, in communication networks, data packets that carry the information can be dropped, delayed, or even reordered due to the network traffic conditions. As new technologies and applications emerge, the fields are coming closer together, and we are indeed witnessing that the "next phase of the information technology revolution is the convergence of communication, computing, and control" [47].

With this convergence of the fields, feedback loops are now being implemented with information passing through communication networks. New applications that utilize these types of feedback loops include future battlefield systems, urban search and rescue, internet-based control, "smart homes," sensor networks, unmanned air vehicles, multi-vehicle systems, and many more [47]. Introducing communication networks in the feedback loops gives several advantages, including modularity and reconfigurability of system components, simple and fast implementations, powerful system diagnosis tools, etc. Of course, a main advantage of Networked Control Systems is that they allow for the use of control systems with spatially distributed components, i.e., actuators, sensors, processing units, and plants that do not need to be physically collocated. There are of course potential issues that arise when closing the loop around imperfect communication links, including data dropout, delays, and quantization effects. Researchers have addressed many of these issues, see [1, 25] for recent surveys, with the management of the information flow through the loop becoming equally important as the design of the controller. These types of feedback loops, called Networked Control Systems

(NCS), have become a fast growing area of research, and managing the information flow in these loops has direct consequence on the system performance.

Multi-agent coordinated control systems serve as a prime example of an emerging area of feedback control systems that utilize feedback loops, with information passed through possibly imperfect communication networks. In these systems, agents must communicate information amongst themselves to achieve a desired global objective. Since the agents are not physically collocated, and there are typically a large number of agents, the information exchange takes place through a communication network. The quality of the information thus again directly impacts the performance of these systems. Hence, managing the information flow in multi-agent systems is also of great importance.

As mentioned above, there are a multitude of emerging applications for Networked Control Systems and multi-agent control systems, such as mobile sensor networks [55] and unmanned aerial vehicles [68]. To illustrate the types of potential design choices and specific choices for these systems, consider as a motivating example those applications which are similar to the Caltech Multi Vehicle Wireless Testbed (MVWT) [29], where remote vehicles with limited computational capability are sent trajectory and/or control commands from remote processing units. A schematic of the MVWT is shown in Fig. 1.1. The MVWT is a good laboratory example for any application involving the feedback control of a plant that is not physically collocated with the sensors and some of the processing units, or any multi-vehicle coordination application.

The vehicles sit in an arena with a goal to perform some desired task, typically defined by a trajectory to track. Overhead vision cameras capture images of the vehicles in the arena, and these images are processed by a remote computer to determine the position and orientation of the vehicles. The remote computer can communicate with the vehicles through a wireless communication channel, subject to quantization, delay, and random loss effects. The inter-vehicle communications occur through this same network. Here is where the Networked Control Systems architecture questions and information management play a role. Should the raw position and orientation measurements determined at the remote computer be transmitted across the network to the vehicles where they can then be used to determine a control action on the vehicles' computers? Alternatively, an estimator can be placed at the remote computer, where the control actions can then be determined and transmitted to the vehicles. The type of communication protocol to be used for either case is also a decision to be made. In what manner should the vehicles communicate amongst themselves based on the specific task? These and many more questions arise when designing feedback loops in settings where the information is sent through imperfect communication channels. This thesis attempts to tackle some of these issues by considering some specific architectural designs and network types and implementing algorithms for these settings.

Issues related to quantized and delayed information, while important in certain Networked Control Systems settings, are ignored in this thesis. With the observation that most Networked Control

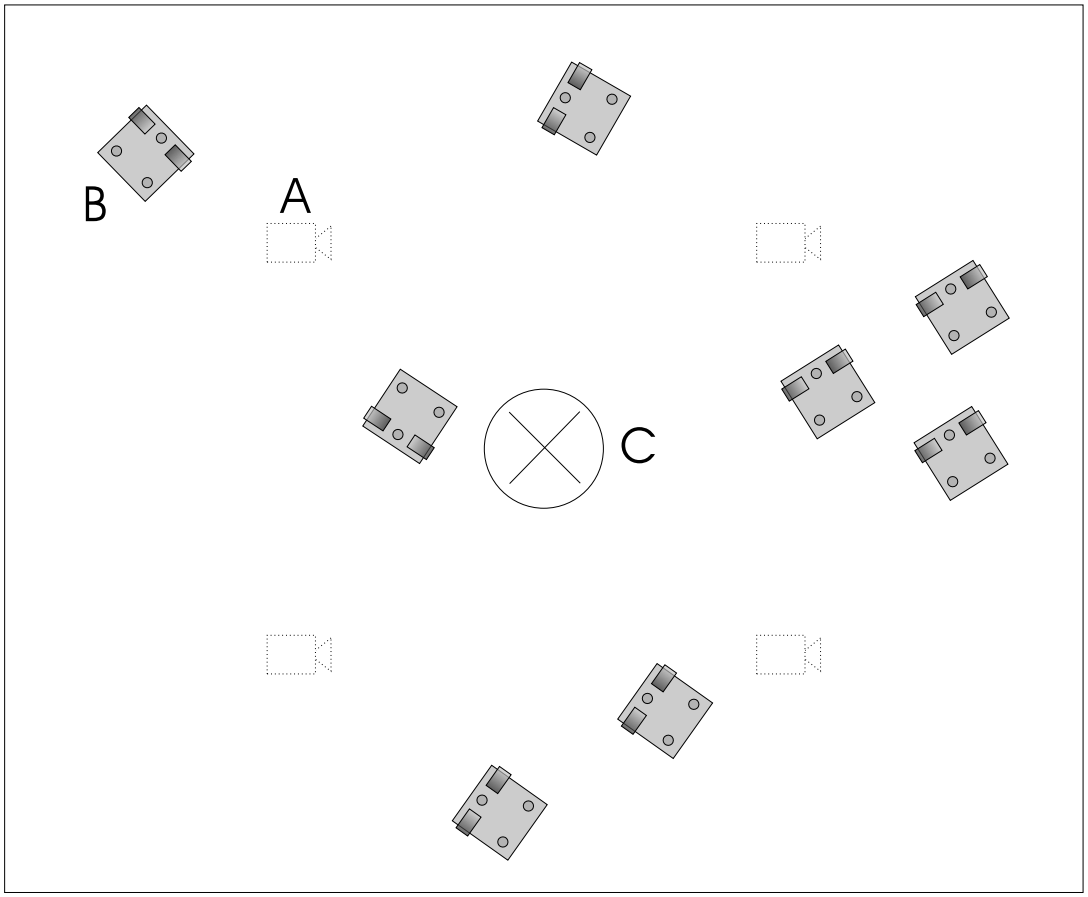

Figure 1.1: A schematic diagram of the Multi Vehicle Wireless Testbed (MVWT) at Caltech.

Systems networks have sufficient bandwidth to transmit information in the form of packets [32], this thesis simply ignores issues related to quantization effects due to limited bandwidth networks. Likewise, as old information is not as useful as newer or current information, delays are ignored by simply considering any delayed information as being lost. Utilizing the delayed information may lead to improved performance, so ignoring the delayed information can yield conservative results (especially as this effectively increases the percentage of dropped packets); nonetheless the algorithms are simpler and useful for a wide variety of problems. Therefore, this thesis only considers designing algorithms for settings where packets of information might be lost, as well as situations where the goal is simply to utilize network resources more effectively to improve performance.

The focus of this thesis is to propose and analyze algorithms for managing information flow in particular Networked Control Systems and multi-agent settings: (i) probabilistic performance characterization of state estimation across a lossy network with buffered measurements; (ii) using actuation buffers to send less frequent but more informative control packets across a network; (iii) designing state estimators when control signals are sent across a lossy network with no acknowledge-

ment signal at the estimator; and (iv) hierarchically decomposing a given communication topology to speed up the time to convergence in multi-agent consensus.

1.2 Background and Relevant Works

In recent years, networked control problems have gained much interest in the research community. An attempt is made here to only create a snapshot of recent results. This section is not intended to be an exhaustive recap of all Networked Control Systems and multi-agent research. For more detailed descriptions, readers are referred to a few recent survey papers of these fields [1, 25, 57].

The related works in Networked Control Systems research focus on estimation and feedback control problems with networks inserted between the dynamical elements of the loop. Networked Control Systems feedback loops are used for the same purpose as standard feedback loops, namely state estimation and closed loop stabilization and performance of a plant. The goals of the Networked Control Systems research are to characterize performance degradations due to the network effects and design algorithms to compensate for those effects. For a thorough comparison of the different types of network technologies available for use in Networked Control Systems see [32]. Despite the specific choice of network type, they all suffer similar effects that degrade the feedback loop performance. The network effects most commonly considered are bandwidth constraints of the communication channel, sampling and delay issues, and information loss due to dropped packets. Architecture questions, e.g., what types of networks to use and where they should be placed, are also prevalent in the literature.

One of the first areas of research investigated with the advent of using digital networks is that of quantization. Quantization converts regions of real numbers (analog signals) into discrete points (digital signals) via a finite set of integers. Some of the information of the signal is lost due to this quantization process, and it affects the closed loop system. Delchamps [10] studies stability when closing the loop with quantized measurement signals. Quadratic stabilization of a sampled-data system with quantization effects is analyzed in [28]. The quantization schemes can be tailored for feedback control applications. In [11], logarithmic-based quantization methods are used. Variable-step quantization algorithms that evolve with time are the focus in [24, 6]. In [7] the authors design the least destabilizing quantizer by reducing it to the multicenter problem from locational optimization.

Related to quantization is the problem of designing feedback loops with finite bandwidth communication channels. One of the first introductions to the problem of state estimation and stabilization of a linear time invariant(LTI) system over a digital communication channel which has a finite bandwidth capacity is by Wong and Brockett [85, 86]. Here they give stability conditions for encoder-decoder algorithms that relate the bandwidth of the communication channel to the system

dynamics. The works by Nair and Evans [49, 51, 50] study the stochastic stability of feedback control systems with limited data rate and show the relation to quantization theory. In addition to the limited data rate effect, they introduce system process and measurement noise into their model. Taking an information-theoretic point of view, the thesis by Sahai [66] derives stability conditions based on anytime information which quantifies the "time value" of data bits. Tatikonda's thesis [81] analyzes the necessary data rate and coding schemes to stabilize a plant across a noisy channel. A general extension of Bode's integral inequality is provided in [38] to assess the performance limitations of feedback control over finite capacity memory-less channels. Researchers also consider such issues as nonlinear systems [52], robustness to plant uncertainties [60], and disturbance attenuation [37].

Due to the nature of transmitting signals across networks, the information passed through the feedback loop can be delayed. In [5], the authors analyze the influence of the sampling rate and network delay on system stability. Nilsson's thesis [54] analyzes delays that are either fixed or random according to a Markov chain. He solves the LQG optimal control problem for the different delay models. Luck and Ray [36] compensate for delays by placing observers throughout the loop. In [33, 34], the authors use a stochastic approach to study time-varying delays. Montestruque and Antsaklis [45] study the stability in the presence of time varying delays that could be driven by an underlying Markov chain. In [42, 43, 44], they determine the maximum time between samples to ensure stability for periodic sampling.

Network scheduling and sampling is related to the problem of delay. Research in this area is concerned with designing when different components of a feedback loop should have access to a shared network resource. Walsh et al. [84] propose the maximum error first scheduling algorithm for systems where sensor nodes share access to a communication network. Their algorithm is based on transmitting data from the node that is most different from the data previously transmitted from that node. They introduce the notion of maximum allowable transfer interval between successive transmissions such that the overall system is stable. Nesic and Teel [53] also consider the effect of sampling times on stability with nonlinear plants and obtain a tighter bound on the time between samples. In [91], the authors reduce the frequency of communication, i.e., sampling, by placing estimators throughout the loop and only transmitting information when the estimated and true values differ by more than a given amount.

In addition to sampling and delay issues associated with Networked Control Systems, lost information is of critical concern to feedback loops. Packets of information can be lost due to network congestion as well as timeout features on the receive side that simply ignore data that is too old. Recently, researchers have been examining estimation and control problems in the presence of information loss. In [73], Sinopoli et al. discuss how packet loss can affect state estimation using Kalman filters. They show that there exists a certain threshold of the packet loss rate above which the state estimation error diverges in the expected sense, i.e., the expected value of the error covariance

matrix becomes unbounded as time goes to infinity. They provide lower and upper bounds of the threshold value which depend on how unstable the process is that is being estimated. Following this spirit, Liu and Goldsmith [35] extended the idea to the case where there are multiple sensors, and the packets arriving from different sensors are dropped independently. They provide similar bounds on the packet loss rate for a stable estimate, again in the expected sense. The minimum variance estimation problem with packet losses is also studied in [39]. Modeling the problem as a jump linear system, Smith and Seiler [77] construct a jump linear estimator to cope with losses.

Sinopoli et al. [74] look at closing the loop across lossy networks and solving an optimal control problem. They insert a lossy network between the controller/estimator and the actuators/plant and attempt to solve the LQG problem when sensor and actuator packets are lost. In this work they make the implicit assumption that the estimator/controller has direct knowledge about the fate of the control packet sent to the plant by way of an acknowledgement signal, i.e., a TCP-like protocol. They use this assumption to show that a separation principle holds and that the optimal LQG control is linear with a bounded cost when the percentage of loss is below a threshold. In [67] they consider using a UDP-like protocol, where no acknowledgement signal is present, and show that the separation principle only exists for a very specific instance. In general, the optimal control problem does not have a closed form solution. Imer et al. obtain similar results in [27]. In [23], the authors also solve the LQG problem with sensor measurements transmitted across a packet-dropping link, but they consider any arbitrary drop pattern, whereas most of the other works focus on i.i.d. or markov dropping networks. Seiler et al. [70] also use jump linear systems theory to solve the H_∞ problem in the presence of packet loss.

In addition to NCS, another research area that uses communication networks to pass information between dynamical elements, which is considered in this thesis, is that of multi-agent control systems. These settings, like NCS, require information be exchanged between dynamical elements. As a result, the information flow has a direct impact on the system performance. There are many emerging applications for multi-agent systems including: consensus [58], behavior of swarms [87], multi-vehicle formation control [19], sensor fusion [78], load balancing [8], and many others. In this thesis the area considered is that of multi-agent consensus, where a collection of agents must reach consensus/agreeement on a value of interest using a distributed algorithm. The most typical problem is the average consensus problem, where the consensus value is the average of the agents' initial conditions. To reach consensus, the agents must communicate their values to other agents, but they can only communicate with some subset of the other nodes. Given certain conditions on the topology of this communication network, and using an update rule that changes their value in the direction of the aggregate value of the nodes they communicate with, consensus can be achieved.

The early roots of the consensus problem studied today can be found in works studying distributed computation over networks, namely the thesis by Tsitsiklis [82] and work by Borkar and

Varaiya [4] as well as Tsitsiklis et al. [83] on the asynchronous asymptotic agreement problem. Much of the current work uses tools from graph theory [22] to represent the information sharing topology and aid in the analysis in consensus problems. Particularly, the graph Laplacian is the matrix representing the evolution of the agent's values based on the communication topology, and the properties of this matrix directly relate to the behavior of the consensus algorithm. The focus of the literature today is on providing conditions on the topology such that consensus can be reached, as well as designing different algorithms to improve the performance of the consensus loop.

The consensus algorithms can include communication schemes that are either continuous [58] or discrete [63] in nature. Asynchronous communication patterns are considered in [3, 40]. Issues such as communication delays and changes in the communication topology over time have been examined, see [58] and [62]. Moreau [46] and Olfati-Saber et al. [58] also consider using communication links that are not bidirectional. Consensus using quantized information is analyzed in [30]. Dynamic consensus algorithms, where the agents' measured values and hence the average consensus value change over time, is studied in [79, 20]. All these works focus on proving conditions such that consensus can be reached. Invariably, the conditions relate to connectivity requirements on the communication topology, e.g., that the union of the graph Laplacians over time be jointly connected for random graphs.

In addition to simply proving convergence can be achieved, another key area of research is speeding up the time to reach consensus. The key factor in determining the time to convergence has been shown [57] to be the second smallest eigenvalue of the graph Laplacian, represented as λ_2. Olfati-Saber [56] introduces additional communication links into the network to create small world networks. In [90], Yang et al. attempt to speed up the time to convergence, while trading-off robustness, by optimally choosing how much relative weight each node should give to the other nodal values in the update rule in order to maximize the ratio $\lambda_2/\lambda_{\max}$. A version for the discrete time consensus is given in [88].

1.3 Summary of Contributions and Overview of Thesis

The contributions of this thesis are to analyze different scenarios of networked control and multi-agent systems and to provide algorithms suitable for these scenarios. The scenarios and algorithms investigated are:

- *(Chapter 2)* Probabilistic performance of state estimation with buffered measurements sent across a lossy network.
- *(Chapter 3)* Analyzing the use of actuation buffers to reduce the frequency of communication when transmitting control signals to a remote plant.

- *(Chapter 4)* Estimation algorithms when transmitting control packets across a lossy network with no acknowledgement signal at the estimator.

- *(Chapter 5)* Using hierarchical decomposition to speed up the time to reach consensus in multi-agent average consensus problems.

The first scenario, explored in Chapter 2, is state estimation with measurements transmitted across a lossy network. A linear dynamical system with known dynamics and Gaussian noise is measured by sensors that transmit measurements to a remote estimator. The transmitted measurement packets are assumed to be subject to random losses, so they may not always be available when updating the estimator. The work by Sinopoli et al. [73] introduces the problem of performing Kalman filtering in the presence of lost measurement packets. They use a modified Kalman filter that performs the prediction and correction steps when the measurement is available, and only the prediction step when the measurement is lost. They show that there exists a relationship between the probability of receiving the measurements and the expected value of the estimation error covariance going unstable. There are two novel ideas presented in Chapter 2 that improve upon the work of [73]:

- To consider a probabilistic performance description of the estimation error covariance.

- To insert a buffer at the sensor which allows transmission of a finite number of past measurements along with the current measurement.

As a result of the random measurement packet loss, the estimation error covariance becomes a random quantity. In [73], estimator stability is given in terms of the expected value of the error covariance $\mathbf{E}[P_k]$. While this is a reasonable performance metric to start with, it can obscure the fact that in some cases the expected value grows unbounded due to the extremely low probability event of receiving no packets. To give a performance metric that better characterizes the estimation performance, I consider determining the probability that the error covariance is below a certain bound $\mathbf{Pr}[P_k \leq M] \leq 1 - \epsilon$. In addition to giving a more complete description of the estimator performance, the analysis can be used for situations where even if $\mathbf{E}[P_k]$ is unstable, it may turn out that the error covariance is below a given value for the vast majority of the time, and this might be considered acceptable performance. This work also shows the benefit of buffering measurements at the sensor and transmitting a packet that contains a sequence of previous measurements.

Keeping with the idea of using buffers to help mitigate effects of missing information, in Chapter 3 I consider using an actuation buffer when transmitting control signals to a remote plant. In this setting it is assumed that the goal is to stabilize a remote plant while at the same time reduce the frequency with which the control signals are transmitted to the plant. Reducing the frequency of transmission could reduce the bandwidth usage when packets have a fixed overhead and minimum

space allocated for data, which can allow better sharing of a network resource amongst multiple users. Thus, the goal is to design a method of transmitting less frequent but more informative packets.

The algorithm consists of a state-feedback controller that computes the control value for the current time-step as well as a prediction of control values for a finite amount of time in the future using the available plant model. The newly computed buffer of control signals is compared to the corresponding predicted control values of the buffer that was last transmitted to the plant. The two differ since the previously computed value uses a prediction of the state, while the newly computed value uses a more recent measured value of the state, and these differ due to the model uncertainties. If the difference between these two control signals is above a certain threshold, then the new buffer is transmitted otherwise it is discarded. I show how the length of the buffer and threshold value combine with the plant dynamics and noise characteristics to affect the transmission frequency and closed loop performance, thus providing tools for the system designer to choose these parameters.

In Chapter 4 I again consider the case where control signals are sent to a remote plant across a network. Here the signal is transmitted every time-step, but the network can drop packets. If the control packet is received, then the corresponding value is applied to the plant. As opposed to the previous setting, here the packets contain only the current control value; thus if the packet is dropped, no control is applied, and the plant evolves open loop. In this chapter I focus on the problem of designing a stabilizing estimation scheme when the estimator has no knowledge of the fate of the control packet, and hence if the plant applied the control signal or evolved open loop. When the estimator does not know what control signal is applied, the standard separation principle for the design of the estimator and controller does not hold. As a result, nearly all prior work in the NCS field that has considered designing an estimator in this setting assumes an acknowledgement signal so that the estimator knows what control signal is applied to the plant. This type of communication protocol is termed "TCP-like" communication. The "UDP-like" communication protocols used here do not provide an acknowledgement signal but could be advantageous due to lower latency, less overhead in terms of packet length, and decreased software complexity.

In this chapter I present methods to perform state estimation for the purpose of designing stabilizing control laws when the control signal is sent across a lossy UDP-like network with no acknowledgment signal of the control packet fate at the estimator. I begin with a discussion of certain naive estimation schemes that can be used and show why these are ineffective. Instead, a novel algorithm is designed that uses the system measurements to reason whether or not the previous control packet was received. The algorithm consists of an estimator, made up of a mode detector and state observer, and state feedback implemented with one of two options:

- using an enlarged control value that guarantees proper detection of the control packet fate or

- using standard state feedback and tolerate possible misdetections of the control packet but achieve better closed loop performance.

I analyze both of these options and present stability conditions. I also compare these techniques with that of the unknown input observer.

The final scenario, studied in Chapter 5, of this thesis relates to the area of multi-agent consensus. The goal in the consensus problem is to find a distributed algorithm such that a collection of agents reaches agreement on some scalar quantity of interest, typically the average of the agents' "initial conditions." To do so the agents must communicate their values to other agents, but they can only communicate with some subset of the other nodes. This problem has been extensively studied lately, see [57] and references therein. The consensus algorithm that is commonly used updates the agents' local value according to the dynamics $\dot{x} = -Lx$, where x is a vector of all the agents' values, and L is a matrix representing the communication graph of the agents. The first works in this field are concerned with proving what conditions on the communication topology are required for the algorithm to converge to the average consensus value. More recent work focuses on improving the time to reach consensus. Many different approaches have been presented with the underlying feature of attempting to increase the second smallest eigenvalue, λ_2, of the matrix L. The contribution made in this chapter is to introduce a new approach to speed up convergence in consensus algorithms applicable to graphs that can be decomposed into a hierarchical graph.

The algorithm presented consists of hierarchically decomposing the given communication topology by splitting the overall graph into layers of smaller connected subgraphs. The consensus dynamics are performed within the individual subgraphs starting with those of the lowest layer of the hierarchy and moving upwards. Certain "leader" nodes bridge the layers of the hierarchy. By exploiting the larger λ_2 values of the smaller subgraphs, this scheme can achieve faster overall convergence than the standard single-stage consensus algorithm running on the full graph topology. Furthermore, using the consensus dynamics for the individual subgraphs endows them with the benefits associated with standard consensus algorithm, such as robustness to information perturbations, no need for a global planner within the subgraphs, etc. The contribution of this chapter is to extend the basic understanding of consensus algorithms to situations when the system may have a hierarchical structure. This hierarchical structure is typical in layered communication networks, where some nodes are gateways between clusters of local nodes and the rest of the network. A bound on the consensus error under this algorithm is presented and compared to the standard single-stage consensus algorithm run on the full communication topology.

Finally, in Chapter 6 I again summarize the main contributions of this thesis and discuss extensions for this research and overall directions of future work for managing information in NCS and multi-agent systems.

1.4 Mathematical Notation and Basic Definitions

This section provides a quick description of some of the mathematical notation and basic definitions that appear in the remainder of this thesis.

The transpose of a vector x is x'; likewise for a matrix X it is X'. The trace of a matrix X is denoted by $Tr(X)$. For a given vector $x \in \mathbb{R}^n$, all norms $\|x\| = \sqrt{x'x}$ are assumed to be the standard Euclidean norm (2-norm). For any matrix $X \in \mathbb{R}^{n \times n}$, the norm $\|X\|$ is the corresponding induced matrix norm unless otherwise explicitly stated. We also use the H-norm, defined for some positive definite matrix $H > 0$. For a matrix X we have $\|X\|_H = \|H^{\frac{1}{2}} X H^{-\frac{1}{2}}\|$, and for a vector x it is $\|x\|_H = \sqrt{x^T H x}$. For any positive definite matrix $X > 0$, denote the smallest and largest eigenvalues of X by $\underline{\lambda}(X)$ and $\overline{\lambda}(X)$, respectively. For any positive semi-definite matrix $X \geq 0$ with all real non-negative eigenvalues, let $\lambda_2(X)$ represent the second smallest eigenvalue of X and $\rho(X)$ its spectral radius, i.e., the magnitude of the largest eigenvalue.

Various notions relating to probability theory and stochastic dynamical systems appear in this thesis. The abbreviation i.i.d. is used for any random distribution that is independent and identically distributed. The probability of any random event occurring is given by $\mathbf{Pr}[\cdot]$. For any random variable x the mathematical expectation of this variable is given by $\mathbf{E}[x]$. A random sequence $\mathbf{x}_k$ is called *almost surely stable* if

$$\mathbf{Pr}\left[\lim_{k \to \infty} \|\mathbf{x}_k\| = 0\right] = 1 \, .$$

It is called *stable in the m^{th} moment* if

$$\sup_k \mathbf{E}\left[\, \|x_k\|^m \, \right] < \infty \, .$$

Chapter 2

Probabilistic Performance Characterization of State Estimation Across a Lossy Network

2.1 Introduction

The first area of networked control systems considered in this thesis is the problem of state estimation over a network. This is a subject that has been widely studied recently. The problem of state estimation and stabilization of a linear time invariant (LTI) system over a digital communication channel which has a finite bandwidth capacity is introduced by Wong and Brockett [85, 86] and further pursued by [6, 49, 81, 59]. In [73], Sinopoli et al. discuss how packet loss can affect state estimation. Due to the random packet drops, the estimation error covariance becomes a random quantity as well. They show there exists a certain threshold of the packet loss rate above which the state estimation error diverges in the expected sense, i.e., the expected value of the error covariance matrix becomes unbounded as time goes to infinity. They also provide lower and upper bounds of the threshold value. Following the spirit of [73], in [35], Liu and Goldsmith extend the idea to the case where there are multiple sensors and the packets arriving from different sensors are dropped independently. They provide similar bounds on the packet loss rate for a stable estimate, again in the expected sense.

The problem of state estimation of a dynamical system where measurements are sent across a lossy network is also the focus of this chapter. Despite the great progress of the previous researchers, the problems they studied have certain limitations. For example, in both [73] and [35], they assume that packets are dropped independently, which is certainly not true in the case where burst packets are dropped or in queuing networks where adjacent packets are not dropped independently. They also use the expected value of the error covariance matrix as the measure of performance. This can conceal the fact that events with arbitrarily low probability can make the expected value diverge,

and it might be better to ignore such events that occur with extremely low probability. For example, consider the simple unstable scalar system in [73]

$$\begin{aligned} x_{k+1} &= ax_k + w_k \\ y_k &= x_k + v_k \ , \end{aligned}$$

with $a = 2$. Let the packet arrival rate be given by $\gamma = 0.74 < 1 - 1/a^2$. According to [73], the expected value of the estimation error covariance, $\mathbf{E}[P_k]$, grows unbounded with time. This is easily verifiable by considering the event T where no packets are received in k time steps. Then, $\mathbf{E}[P_k] \geq \mathbf{Pr}[T]\mathbf{E}[P_k|T] \geq (0.26^k)4^k P_0 = 1.04^k P_0$. By letting k go to infinity, we see that this event with almost zero probability makes the expected error diverge.

The goal of the present work is to give a more complete characterization of the estimator performance by instead considering a probabilistic description of the error covariance. We show the covariance is bounded above by a given bound with a high probability, i.e.,

$$\mathbf{Pr}[P_k \leq M] = 1 - \epsilon \ . \tag{2.1}$$

The importance of this characterization lies in the fact that while the expected value of P_k may diverge due to events with very low probability, in fact the actual value of P_k can be below an acceptable limit for a vast majority of the time. For this expression to hold, it requires an estimator that has a finite upper bound whenever a measurement packet is received. We construct such an estimator by using a buffer of previous measurements. We also show how to determine the relationship between M and ϵ.

The rest of the chapter is organized as follows. In Section 2.2, the mathematical model of our problem is given, followed by an analysis of the Kalman filter using a buffer of measurements and an observer-based estimator. The estimation algorithm that provides an upper bound to the error covariance whenever a measurement packet arrives is described in Section 2.3. In Section 2.4, we show that M exists for any given ϵ for the expression in Eqn. (2.1). In Section 2.5, we give an explicit relationship between the bound and probability of the error covariance staying below the bound. In Section 2.6 we compare our metric with that of [73] by means of an example. The chapter concludes with a summary of our results and a discussion of the work that lies ahead. This chapter is a joint work with Ling Shi, Abhishek Tiwari, and Richard M. Murray. It is published in [72, 16].

2.2 Problem Set Up

2.2.1 Problem Setting

We consider estimating the state of a discrete-time LTI system

$$\begin{aligned} x_{k+1} &= Ax_k + w_k \\ y_k &= Cx_k + v_k. \end{aligned} \tag{2.2}$$

As usual, $x_k \in \mathbb{R}^n$ is the state vector, $y_k \in \mathbb{R}^m$ is the observation vector, and $w_k \in \mathbb{R}^n$ and $v_k \in \mathbb{R}^m$ are zero mean white Gaussian random vectors with $\mathbf{E}[w_k {w_j}'] = \delta_{kj} Q \geq 0$, $\mathbf{E}[v_k {v_j}'] = \delta_{kj} R > 0$, and $\mathbf{E}[w_k {v_j}'] = 0 \ \forall j, k$, where $\delta_{kj} = 0$ if $k \neq j$ and $\delta_{kj} = 1$ otherwise. We assume the pair (A, C) is observable and $(A, Q^{\frac{1}{2}})$ is controllable and, to make the estimation problem interesting, that A is unstable.

We assume the sensor measurements y_k are sent across a lossy network to the estimator, with no delay and negligible quantization effects. Thus, the estimator either receives a perfectly communicated measurement packet or none at all. It is assumed the network losses are random events. Let γ_k be the random variable indicating whether a packet is received at time k or not, i.e., $\gamma_k = 0$ if a packet is dropped, and $\gamma_k = 1$ if it is received.

In addition, we assume that the sensor has the ability to store measurements in a buffer. Therefore, each packet sent through the network contains a finite number of the previous measurements, and the network has significant bandwidth to transmit those measurements at each time instant. Figure 2.1 shows a schematic of the system set up. Note the measurement packet sent across the network $\tilde{y}_k$ consists of the previous $S + p$ measurements taken by the sensor.

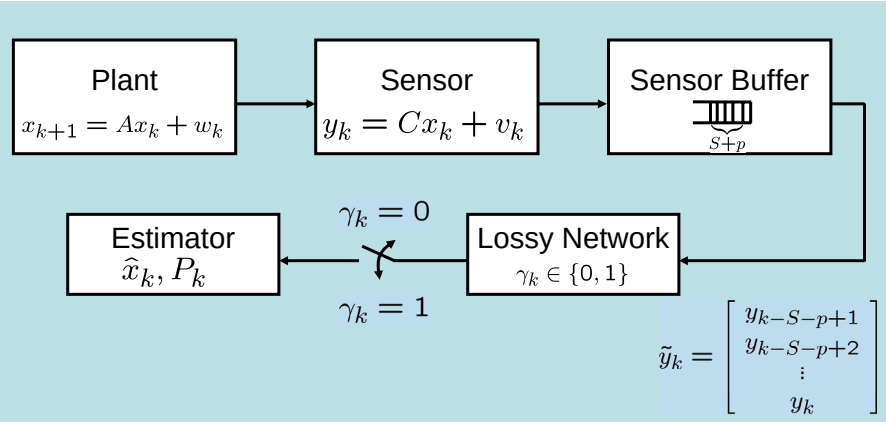

Figure 2.1: Schematic of NCS for state estimation across a lossy network.

2.2.2 Kalman Filtering Across a Lossy Network

Sinopoli et al. [73] consider this set up without a buffer, i.e., $S+p=1$. They show that the Kalman filter is still the optimal estimator in this setting. There is a slight change to the standard Kalman filter in that only the time update is performed when the measurement packets are dropped. When a measurement is received, the time and measurement update steps are performed. The filtering equations become

$$\begin{aligned}
\hat{x}_{k+1|k} &= A\hat{x}_{k|k} \,, && (2.3)\\
P_{k+1|k} &= AP_{k|k}A' + Q \,, && (2.4)\\
\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + \gamma_{k+1}K_{k+1}(y_{k+1} - C\hat{x}_{k+1|k}) \,, && (2.5)\\
P_{k+1|k+1} &= P_{k+1|k} - \gamma_{k+1}K_{k+1}CP_{k+1|k} \,, && (2.6)
\end{aligned}$$

where $\gamma_{k+1} \in \{0,1\}$ indicates if the measurement y_{k+1} is received, and $K_{k+1} = P_{k+1|k}C'(CP_{k+1|k}C' + R)^{-1}$ is the Kalman gain matrix. Note Eqn.s (2.3) – (2.4) are the Kalman Filter Time Update equations and Eqn.s (2.5) – (2.6) are the Measurement Update equations. The a priori and a posteriori error covariances are $P_{k+1|k}$ and $P_{k+1|k+1}$, respectively.

Unlike the standard Kalman filtering setting where the error covariance matrix is a deterministic quantity (given an initial value), the randomness of the lossy network makes the covariance a random variable as well. Nonetheless, the update equation for the a priori covariance can still be characterized as

$$P_{k+1} = AP_kA' + Q - \gamma_k AP_kC'[CP_kC' + R]^{-1}CP_kA' \,, \tag{2.7}$$

where we simply write $P_k = P_{k|k-1}$. For the case where γ_k is an i.i.d. variable with mean γ, Sinopoli et al. [73] show that there exists a critical value $\gamma_c = 1 - 1/\lambda_{\max}(A)^2$ which determines the stability of the expected value of the estimation error covariance $\mathbf{E}[P_k]$ as $k \to \infty$. As mentioned in Section 2.1, we are interested in a different metric to evaluate the estimator performance,

$$\mathbf{Pr}[P_k \leq M] = 1 - \epsilon \,. \tag{2.8}$$

The key to evaluate this expression is that the error covariance has an upper bound following a single measurement packet being received. A packet being received means the Kalman filter time and measurement updates are applied. As shown below, if $\text{rank}(C) = n$, an upper bound for the error covariance is achieved following a single measurement update, and the transmitted packet only needs to contain a single measurement. If $\text{rank}(C) < n$, we can still get an upper bound for the covariance, but only after several Kalman filter measurement update steps are applied, hence the transmitted packet must contain a sequence of previous measurements in this case.

Given the system parameters (A, C, Q, R), then for any positive semidefinite matrix $X \geq 0$, define the following functions:

$$h(X) = AXA' + Q \,, \tag{2.9}$$

$$g(X) = AXA' + Q - AXC'(CXC' + R)^{-1}CXA' \,. \tag{2.10}$$

Note that applying these functions to P_k yields P_{k+1} according to Eqn. (2.7)

$$P_{k+1} = \begin{cases} h(P_k), & \text{if } \gamma_k = 0 \\ g(P_k), & \text{if } \gamma_k = 1 \end{cases} \,. \tag{2.11}$$

We adopt the notation that $g^m(X)$ and $h^m(X)$ mean to apply the g and h functions m times starting from X, with $g^0(X) = h^0(X) = X$. Note that if we write $X = g(X)$ in Eqn. (2.10), this is equivalent to the discrete algebraic Riccati equation (DARE). Denote the solution to this equation by

$$\overline{P} = g(\overline{P}) \,, \tag{2.12}$$

which is also the steady state covariance if all measurements are received (i.e., if $\gamma_k = 1 \; \forall k$, then $\lim_{k\to\infty} P_k = \lim_{k\to\infty} g^k(P_0) = \overline{P}$ for any $P_0 \geq 0$). Next we establish an upper bound on $g(X)$ when $\text{rank}(C) = n$. First note that by using the matrix inversion lemma [26], we can rewrite $g(X)$ in Eqn. (2.10) as

$$g(X) = A\left(X^{-1} + C'R^{-1}C\right)^{-1} A' + Q \,. \tag{2.13}$$

Lemma 2.1 *If $X \geq Y \geq 0$, then $g(X) \geq g(Y)$ and $h(X) \geq h(Y)$.*

Proof: See [73] Appendix A. ■

Lemma 2.2 *For any positive semidefinite matrix $X \geq 0$, if $\text{rank}(C) = n$, then*

$$g(X) \leq A\left(C'R^{-1}C\right)^{-1} A' + Q \,. \tag{2.14}$$

Proof: From Lemma 2.1 we see that $g(X)$ is an increasing function. Then from the functional form of $g(X)$ in Eqn. (2.13) and taking the limit as the argument goes to infinity we see that

$$\lim_{X\to\infty} g(X) = \lim_{X\to\infty} A\left(X^{-1} + C'R^{-1}C\right)^{-1} A' + Q = A\left(C'R^{-1}C\right)^{-1} A' + Q \,.$$

Since it is an increasing function, this is an upper bound for $g(X)$, and the result holds. Note that $\text{rank}(C) = n$ is required so that $\text{rank}(C'R^{-1}C) = n$, and this quantity is invertible. ■

Corollary 2.3 *If C is square and invertible the upper bound becomes*

$$g(X) \leq AC^{-1}RC'^{-1}A' + Q \; . \tag{2.15}$$

Proof: This is a direct result of Lemma 2.2 with $\left(C'R^{-1}C\right)^{-1} = C^{-1}RC'^{-1}$ since C is square and invertible. ■

Thus we see that if $\text{rank}(C) = n$, the covariance can be upperbounded after a measurement is received and applying a single measurement update, since if $\gamma_k = 1$ we have $P_{k+1} = g(P_k) \leq A\left(C'R^{-1}C\right)^{-1}A' + Q$. If $\text{rank}(C) < n$, we still seek a bound on the Kalman filter updates. It turns out the bound exists, but not given a single measurement update. Rather, it requires a sequence of measurement updates be processed for a bound on the covariance to be obtained. To find this bound, we begin with some preliminary results to be used later on.

Similar to the definition of $g(X)$, for any positive semidefinite matrix $X \geq 0$ and system matrices (A, C, Q, R) and matrix T of appropriate dimension such that

$$Q - TR^{-1}T' \geq 0 \; , \tag{2.16}$$

define the following function,

$$\widetilde{g}(X, A, C, Q, R, T) = AXA' + Q - (AXC' + T)\left(CXC' + R\right)^{-1}\left(T' + CXA'\right) \; . \tag{2.17}$$

If the matrix $T = 0$, then we have $\widetilde{g}(X, A, C, Q, R, 0) = g(X)$. The lemma below provides a bound for this function given (A, C, Q, R, T).

Lemma 2.4 *For any positive semidefinite matrix $X \geq 0$, if* $\text{rank}(C) = n$*, then*

$$\widetilde{g}(X, A, C, Q, R, T) \leq \left(A - TR^{-1}C\right)\left(C'R^{-1}C\right)^{-1}\left(A - TR^{-1}C\right)' + Q - TR^{-1}T' \; . \tag{2.18}$$

Proof: First we expand the expression for $\widetilde{g}(X, A, C, Q, R, T)$ in Eqn. (2.17),

$$\begin{aligned} \widetilde{g}(X, A, C, Q, R, T) \;\; = \;\; & AXA' + Q - AXC'\left(CXC' + R\right)^{-1}CXA' - T\left(CXC' + R\right)^{-1}T' \\ & -T\left(CXC' + R\right)^{-1}CXA' - AXC'\left(CXC' + R\right)^{-1}T' \; . \end{aligned} \tag{2.19}$$

If we make the following substitutions for the A and Q matrices,

$$\begin{aligned} A - TR^{-1}C \;\; & \rightarrow \;\; A \\ Q - TR^{-1}T' \;\; & \rightarrow \;\; Q \; , \end{aligned} \tag{2.20}$$

evaluate $g(X)$ in Eqn. (2.10), and denote this by $\widehat{g}(X)$, it is equivalent to

$$\begin{aligned}
\widehat{g}(X) &= (A-TR^{-1}C)X(A-TR^{-1}C)' + (Q-TR^{-1}T') \\
&\quad -(A-TR^{-1}C)XC'\left(CXC'+R\right)^{-1}CX(A-TR^{-1}C)' \\
&= AXA' + Q - AXC'\left(CXC'+R\right)^{-1}CXA' \\
&\quad +T\left(R^{-1}CXC'R^{-1} - R^{-1} - R^{-1}CXC'\left(CXC'+R\right)^{-1}CXC'R^{-1}\right)T' \\
&\quad +T\left(R^{-1}CXC'\left(CXC'+R\right)^{-1} - R^{-1}\right)CXA' \\
&\quad +AXC'\left(\left(CXC'+R\right)^{-1}CXC'R^{-1} - R^{-1}\right)T' \\
&= AXA' + Q - AXC'\left(CXC'+R\right)^{-1}CXA' - T\left(CXC'+R\right)^{-1}T' \\
&\quad -T\left(CXC'+R\right)^{-1}CXA' - AXC'\left(CXC'+R\right)^{-1}T' \,, \qquad (2.21)
\end{aligned}$$

where the last line uses the matrix inversion lemma [26]. Thus we see that Eqn. (2.19) equals Eqn. (2.21) when $g(X)$ is evaluated with the substitutions in Eqn. (2.20), i.e., $\widetilde{g}(X,A,C,Q,R,T) = \widehat{g}(X)$. Furthermore, since $\widehat{g}(X)$ is simply $g(X)$ evaluated with the substitutions in Eqn. (2.20), then $\widehat{g}(X)$ is upper bounded the expression in Eqn. (2.14) with the same substitutions as in Eqn. (2.20). Hence,

$$\widetilde{g}(X,A,C,Q,R,T) = \widehat{g}(X) \leq \left(A-TR^{-1}C\right)\left(C'R^{-1}C\right)^{-1}\left(A-TR^{-1}C\right)' + Q - TR^{-1}T' \,.$$

■

Corollary 2.5 *If C is square and invertible, the upper bound becomes*

$$\widetilde{g}(X,A,C,Q,R,T) \leq AC^{-1}RC'^{-1}A' + Q - AC^{-1}T' - TC'^{-1}A' \,. \qquad (2.22)$$

Proof: This is a direct result of Lemma 2.4 with $\left(C'R^{-1}C\right)^{-1} = C^{-1}RC'^{-1}$ since C is square and invertible. ■

We are now ready to proceed with finding an upper bound for the estimation covariance using a sequence of measurements and the Kalman filter with C noninvertible, i.e., $\text{rank}(C) < n$. To find this bound, begin by defining

$$\mathcal{O}(r) = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{r-1} \end{bmatrix} \,, \qquad (2.23)$$

for any positive integer $r \geq 1$. Next define S to be the smallest integer such that the matrix is rank n, i.e.,

$$S = \min \{r \geq 1 : \text{rank}\,(\mathcal{O}(r)) = n\} \; . \tag{2.24}$$

Since (A, C) is observable, S is guaranteed to exist, and $S \leq n$. Thus, by concatenating the previous S consecutive measurements, the augmented observation vector $\mathcal{O}(S)$ is full rank. For notational convenience we simply let $\mathcal{O}$ represent $\mathcal{O}(S)$ for the remainder of the chapter.

Assume that each measurement packet received at the estimator contains S previous measurements, i.e., the measurement sequence $y_{k-S+1}, y_{k-S+2}, \ldots, y_k$. Then, an estimation algorithm can be run that performs S Kalman filter time and measurement updates starting with stored values of $\hat{x}_{k-S+1|k-S+1}$ and P_{k-S+1} and using the measurement sequence $y_{k-S+1}, y_{k-S+2}, \ldots, y_k$. We call this algorithm the S-step Kalman filter. Following the S update steps, the filter produces an a priori estimate $\hat{x}_{k+1|k}$ with an associated a priori covariance given by $P_{k+1} = g^S(P_{k-S+1})$. In the theorem below we find an upper bound for $g^S(X)$.

Theorem 2.6 *For any positive semidefinite matrix $X \geq 0$, with S defined by Eqn. (2.24), the following upper bound holds,*

$$g^S(X) \leq A\left(\left(A^{S-1} - \widetilde{T}\widetilde{R}^{-1}\mathcal{O}\right)\left(\mathcal{O}'\widetilde{R}^{-1}\mathcal{O}\right)^{-1}\left(A^{S-1} - \widetilde{T}\widetilde{R}^{-1}\mathcal{O}\right)' + \widetilde{Q} - \widetilde{T}\widetilde{R}^{-1}\widetilde{T}'\right)A' + Q \; , \tag{2.25}$$

with

$$\widetilde{Q} \triangleq \mathbf{E}\left[\widetilde{w}_k\widetilde{w}_k'\right] \tag{2.26}$$

$$\widetilde{R} \triangleq \mathbf{E}\left[\widetilde{v}_k\widetilde{v}_k'\right] \tag{2.27}$$

$$\widetilde{T} \triangleq \mathbf{E}\left[\widetilde{w}_k\widetilde{v}_k'\right] \tag{2.28}$$

where

$$\widetilde{w}_k = \sum_{i=0}^{S-2} A^i w_{k-i-1} \tag{2.29}$$

$$\widetilde{v}_k = \begin{bmatrix} v_{k-S+1} \\ Cw_{k-S+1} + v_{k-S+2} \\ \vdots \\ C\left(\sum_{i=0}^{S-2} A^i w_{k-i-1}\right) + v_k \end{bmatrix} . \tag{2.30}$$

Proof: To prove this, we first construct a linear estimator that uses the previous a priori estimate $\hat{x}_{k-S+1|k-S+1}$ and covariance P_{k-S+1} and the sequence of measurements $y_{k-S+1}, y_{k-S+2}, \ldots, y_k$

to produce an estimate of x_k and associated covariance that is shown to be upperbounded. The estimate and covariance are equivalent to using the S-step Kalman filter with the same information set. Then, using this estimate and the corresponding variance upper bound to predict forward to time step $k+1$ we have an a priori covariance that is equivalent to $g^S(P_{k-S+1})$, with corresponding upper bound.

At time k, given the sequence of measurements $y_{k-S+1}, y_{k-S+2}, \ldots, y_k$ and a previous a priori estimate $\hat{x}_{k-S+1|k-S}$ and covariance P_{k-S+1}, we can construct an estimate of the state according to the linear estimator

$$\tilde{x}_k = A^{S-1}\hat{x}^-_{k-S+1} + \tilde{K}_k\left(\tilde{y}_k - \mathcal{O}\,\hat{x}^-_{k-S+1}\right) \ , \tag{2.31}$$

where we use the notation $\hat{x}^-_{k-S+1} = \hat{x}_{k-S+1|k-S}$, and

$$\tilde{y}_k \quad = \quad \begin{bmatrix} y_{k-S+1} \\ y_{k-S+2} \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} Cx_{k-S+1} + v_{k-S+1} \\ Cx_{k-S+2} + v_{k-S+2} \\ \vdots \\ Cx_k + v_k \end{bmatrix} \tag{2.32}$$

for this measurement update step. Note that for $j \geq k-S+1$ we can write

$$y_j \quad = \quad C\left(A^{j-k+S-1}x_{k-S+1} + \sum_{i=0}^{j-k+S-2} A^i w_{j-i-1}\right) + v_j \ .$$

The term in the parenthesis is x_j, and note we have separated the expression in terms of dependence on the state at time $k-S+1$ and the noise sequence from $k-S+1$ to $j-1$. Thus, the measurement packet can be represented as

$$\tilde{y}_k = \mathcal{O}x_{k-S+1} + \widetilde{v}_k \ . \tag{2.33}$$

Note the state at time k can be written as

$$\begin{aligned} x_k \quad &= \quad A^{S-1}x_{k-S+1} + \sum_{i=0}^{S-2} A^i w_{k-i-1} \\ &= \quad A^{S-1}x_{k-S+1} + \widetilde{w}_k \ . \end{aligned} \tag{2.34}$$

Combining Eqn.s (2.33) and (2.34) in Eqn. (2.31) we get that the a posteriori estimation error $\tilde{e}_k = \tilde{x}_k - \hat{x}_k$ is

$$\tilde{e}_k = \left(A^{S-1} - \tilde{K}_k\mathcal{O}\right)e^-_{k-S+1} + \widetilde{w}_k - \tilde{K}_k\widetilde{v}_k \ ,$$

where $e^-_{k-S+1} = x_{k-s+1} - \hat{x}^-_{k-S+1}$ is the a priori estimation error from time $k-S+1$. Note that since e^-_{k-S+1} is an a priori estimation error, it is independent of $\tilde{w}_k$ and $\tilde{v}_k$. Calculating the variance

of the estimation error, $\tilde{P}_k = \mathbf{E}[\tilde{e}_k \tilde{e}_k']$, yields

$$\tilde{P}_k = \left(A^{S-1} - \tilde{K}_k \mathcal{O}\right) P_{k-S+1} \left(A^{S-1} - \tilde{K}_k \mathcal{O}\right)' + \widetilde{Q} + \tilde{K}_k \widetilde{R} \tilde{K}_k' - \tilde{K}_k \widetilde{T}' - \widetilde{T} \tilde{K}_k' \,. \tag{2.35}$$

This variance can be minimized by taking the derivative of the trace of $\tilde{P}_k$ with respect to $\tilde{K}_k$, setting that expression equal to zero and solving for $\tilde{K}_k$. Doing this yields the optimal gain

$$\tilde{K}_k = \left(\widetilde{T} + A^{S-1} P_{k-S+1} \mathcal{O}'\right) \left(\mathcal{O} P_{k-S+1} \mathcal{O}' + \widetilde{R}\right)^{-1} \,.$$

It can be shown that using this optimal gain in Eqn. (2.35) results in the expression

$$\begin{aligned} \tilde{P}_k = {} & A^{S-1} P_{k-S+1} A^{S-1\prime} + \widetilde{Q} \\ & - \left(A^{S-1} P_{k-S+1} \mathcal{O}' + \widetilde{T}\right) \left(\mathcal{O} P_{k-S+1} \mathcal{O}' + \widetilde{R}\right)^{-1} \left(A^{S-1} P_{k-S+1} \mathcal{O}' + \widetilde{T}\right)' \,. \end{aligned} \tag{2.36}$$

Note the right hand side of Eqn. (2.36) is exactly equivalent to $\widetilde{g}(P_{k-S+1}, A^{S-1}, \mathcal{O}, \widetilde{Q}, \widetilde{R}, \widetilde{T})$ from Eqn. (2.17). Therefore, we can apply Lemma 2.4 to get the upper bound

$$\tilde{P}_k \leq \left(A^{S-1} - \widetilde{T} \widetilde{R}^{-1} \mathcal{O}\right) \left(\mathcal{O}' \widetilde{R}^{-1} \mathcal{O}\right)^{-1} \left(A^{S-1} - \widetilde{T} \widetilde{R}^{-1} \mathcal{O}\right)' + \widetilde{Q} - \widetilde{T} \widetilde{R}^{-1} \widetilde{T}' \,. \tag{2.37}$$

Since the estimator described above is a linear estimator that minimizes the a posteriori covariance of the estimate at time k given the a priori estimate $\hat{x}_{k-S+1|k-S+1}$ and the measurement sequence $y_{k-S+1}, y_{k-S+2}, \ldots, y_k$, it must be identical to the Kalman filter that is implemented by starting with $\hat{x}_{k-S+1|k-S+1}$ and P_{k-S+1} and performing the S time and measurement updates according to Eqn.s (2.3) – (2.6), see [80]. This means the a posteriori Kalman filter estimate at time k is given by $\hat{x}_{k|k} = \tilde{x}_k$, and the a priori estimate at time $k+1$ is simply $\hat{x}_{k+1|k} = A\tilde{x}_k$ with variance $P_{k+1} = A\tilde{P}_k A' + Q$. Of course, using the definition of $g(X)$ we can also write $P_{k+1} = g^S(P_{k-S+1})$, therefore

$$g^S(P_{k-S+1}) = P_{k+1} = A\tilde{P}_k A' + Q \,. \tag{2.38}$$

Then, using Eqn. (2.37) we arrive at Eqn. (2.25). ■

Corollary 2.7 *If $\mathcal{O}$ is square and invertible, the upper bound becomes*

$$g^S(X) \leq A^S \mathcal{O}^{-1} \widetilde{R} \mathcal{O}'^{-1} A^{S\prime} + A\widetilde{Q}A' + Q - A^S \mathcal{O}^{-1} \widetilde{T}' A' - A\widetilde{T} \mathcal{O}'^{-1} A^{S\prime} \,. \tag{2.39}$$

Proof: This is a direct result of Theorem 2.6 with $\left(\mathcal{O}' R^{-1} \mathcal{O}\right)^{-1} = \mathcal{O}^{-1} R \mathcal{O}'^{-1}$ since $\mathcal{O}$ is square and invertible. ■

The variances $(\widetilde{Q}, \widetilde{R}, \widetilde{T})$ in Eqn.s (2.26) – (2.28) can be computed according to the following expressions:

$$\widetilde{R} = \text{diag}\left(\widetilde{R}_S\right) + U_S + U_S' \tag{2.40}$$

$$\widetilde{Q} = \widetilde{Q}_S \tag{2.41}$$

$$\widetilde{T} = \left[0_n^m, \widetilde{T}_2, \widetilde{T}_3, \cdots, \widetilde{T}_S\right] , \tag{2.42}$$

with

$$\widetilde{R}_S = \left[R, R + C\widetilde{Q}_2 C', R + C\widetilde{Q}_3 C', \cdots, R + C\widetilde{Q}_S C'\right]$$

$$\widetilde{Q}_i = \sum_{j=0}^{i-2} A^j Q A^{j'} \text{ , for } i = 2, \ldots, S$$

$$U_S = \begin{cases} \left[0_m^{S\cdot m'}, u_2', u_3', \cdots, u_{S-1}', 0_m^{S\cdot m'}\right]' & \text{, if } S \geq 2 \\ 0_m^{S\cdot m} & \text{, if } S = 1 \end{cases}$$

$$u_i = \left[0_m^{i\cdot m}, C\widetilde{Q}_i A' C', C\widetilde{Q}_i A^{2'} C', \cdots, C\widetilde{Q}_i A^{S-i'} C'\right] \text{ , for } i = 2, \ldots, S-1$$

$$\widetilde{T}_i = \sum_{j=S-i}^{S-2} A^j Q A^{j-S+i'} C' \text{ , for } i = 2, \ldots, S \text{ .}$$

The term 0_i^j is used to represent a matrix with i rows and j columns whose elements are all identically zero.

Theorem 2.6 provides an upper bound to $g^S(X)$ and, hence, an upper bound for the S-step Kalman filter. If each measurement packet that is received at the estimator contains S previous measurements, the S-step Kalman filter can be run by using the stored values of $\hat{x}_{k-S+1|k-S+1}$ and P_{k-S+1} and the measurement sequence $y_{k-S+1}, y_{k-S+2}, \ldots, y_k$ to produce an estimate with a bounded variance. With this upper bound, the probabilistic performance metric in Eqn. (2.8) can be evaluated. There is a difficulty in implementing the estimation algorithm in this manner, however. Every time-step a measurement packet is received, a total of S Kalman gains must be computed, and this can be a heavy computational burden. In addition, the bound provided here essentially assumes the a priori variance P_{k-S+1} is infinite. As this does not occur in practice, the bound is conservative.

To remove these issues, in the next section we introduce an observer-based estimator that also utilizes the previous S measurements to obtain an estimate with a bounded variance. It does not, however, have the computational burden of computing S Kalman gains every time-step a measurement packet is received, nor does it require the storage of the old estimate $\hat{x}_{k-S+1|k-S+1}$ and variance P_{k-S+1}.

2.2.3 Observer-Based Estimator

The observer-based estimator described in this section provides a state estimate by inverting out the known dynamics from a finite sequence of past measurements. Like the S-step Kalman filter from the previous section, the variance of the observer-based estimator has an upper bound, but it does not suffer the computational burden associated with computing S Kalman filter gains as mentioned above.

Recall from the definition of $\mathcal{O}$ that it is a matrix with $S{\cdot}m$ rows and n columns with $\text{rank}(\mathcal{O}) = n$; hence, it has a left inverse. Denote the left inverse by

$$\mathcal{O}^\dagger \triangleq (\mathcal{O}'\mathcal{O})^{-1}\mathcal{O}' \,. \tag{2.43}$$

If $\mathcal{O}$ is square and full rank, then note that $\mathcal{O}^\dagger = \mathcal{O}^{-1}$. At time k, given the measurement packet with the sequence of measurements $y_{k-S+1}, y_{k-S+2}, \ldots, y_k$, we can construct the observer-based estimator according to

$$\overline{x}_k^- = A^{S-1}\mathcal{O}^\dagger \tilde{y}_k \tag{2.44}$$

$$\overline{x}_{k+1}^+ = A\overline{x}_k^- \,, \tag{2.45}$$

with $\tilde{y}_k$ as in Eqn. (2.32). Note that $\overline{x}_k$ is the a posteriori estimate at time k, and $\overline{x}_{k+1}^+$ is the a priori estimate of the state at time $k+1$ since it is made using measurements up to time k. Define the estimation errors as $\overline{e}_k^- = x_k - \overline{x}_k^-$ and $\overline{e}_{k+1}^+ = x_{k+1} - \overline{x}_{k+1}^+$ with associated variances $\overline{P}_k^- = \mathbf{E}[\overline{e}_k^- \overline{e}_k^{-\prime}]$ and $\overline{P}_{k+1}^+ = \mathbf{E}[\overline{e}_{k+1}^+ \overline{e}_{k+1}^{+\prime}]$. In the previous sections the quantity of interest is the a priori covariance, so we also consider $\overline{P}_{k+1}^+$ here. The following theorem gives the value for this variance.

Theorem 2.8 *Using the observer-based estimator in Eqn.s (2.44) – (2.45), the a priori covariance is given by*

$$\overline{P}_{k+1}^+ = A^S\mathcal{O}^\dagger\widetilde{R}\mathcal{O}^{\dagger\prime}A^{S\prime} + A\widetilde{Q}A' + Q - A^S\mathcal{O}^\dagger\widetilde{T}'A' - A\widetilde{T}\mathcal{O}^{\dagger\prime}A^{S\prime} \,, \tag{2.46}$$

with $(\widetilde{Q}, \widetilde{R}, \widetilde{T})$ defined as before.

Proof: Using the expression for $\tilde{y}_k$ from Eqn. (2.33) we can then write the estimator in Eqn. (2.45) in terms of x_{k-S+1} as

$$\begin{aligned}
\overline{x}_{k+1}^+ &= A^S\mathcal{O}^\dagger\tilde{y}_k \\
&= A^S\mathcal{O}^\dagger(\mathcal{O}x_{k-S+1} + \tilde{v}_k) \\
&= A^S x_{k-S+1} + A^S\mathcal{O}^\dagger\tilde{v}_k \,,
\end{aligned}$$

Recall the state at time $k+1$ can be written as

$$\begin{aligned} x_{k+1} &= A^S x_{k-S+1} + \sum_{i=0}^{S-2} A^{i+1} w_{k-i-1} + w_k \\ &= A^S x_{k-S+1} + A\tilde{w}_k + w_k \ . \end{aligned}$$

Combining these, the estimation error for this estimator can then be easily seen to be

$$\overline{e}_{k+1}^{+} = A\tilde{w}_k + w_k + A^S \mathcal{O}^{\dagger} \tilde{v}_k \ . \tag{2.47}$$

Computing the variance of this expression, and noting that w_k is independent of both $\tilde{w}_k$ and $\tilde{v}_k$, we arrive at Eqn. (2.46). ■

Note that the expression in Eqn. (2.46) is an identity and not a bound. That is, $\overline{P}_{k+1}^{+}$ is a fixed quantity that depends only on A, C, Q, R and S, and it is independent of any prior estimate covariance. So, whenever S consecutive measurements are available, the observer-based estimator can produce an estimate with error covariance given by Eqn. (2.46). This simply requires that the sensor transmit the previous S measurements at each time-step, just as to have an upper bound for the Kalman filter requires S measurement updates. Of course implementing the observer-based estimator of Eqn.s (2.44) – (2.45) does not suffer from the computational burden of computing S Kalman gains. Notice that if $\mathcal{O}$ is square and invertible, the covariance from the observer-based estimator in Theorem 2.8 is equivalent to the upper bound of $g^S(X)$ from Corollary 2.7. The drawback of using the observer-based estimator, compared to the S-step Kalman filter, is that the covariance of the observer-based estimator in Eqn. (2.46) can be quite large and provides a tight bound on any estimation algorithm using this method. As a result, we develop an algorithm in the next section that uses S measurements to evaluate the observer-based estimator along with additional p measurements to decrease the bound.

2.3 Estimator Algorithm

As mentioned before, the key in evaluating the probabilistic performance description in Eqn. (2.8) is to have an estimator algorithm that provides an upper bounded whenever a measurement packet is received. In the previous section we presented two estimators that achieve this

- the S-step Kalman filter and
- the observer-based estimator.

Both of these estimators require S measurements be transmitted in every packet. The bound of the covariance from the S-step Kalman filter given in Theorem 2.6, which we denote by

$$\widetilde{S} \triangleq A\left(\left(A^{S-1}-\widetilde{T}\widetilde{R}^{-1}\mathcal{O}\right)\left(\mathcal{O}'\widetilde{R}^{-1}\mathcal{O}\right)^{-1}\left(A^{S-1}-\widetilde{T}\widetilde{R}^{-1}\mathcal{O}\right)'+\widetilde{Q}-\widetilde{T}\widetilde{R}^{-1}\widetilde{T}'\right)A'+Q\,, \quad (2.48)$$

is conservative since it is computed by taking the limit as the previous a priori covariance goes to infinity. The bound for error covariance of the observer-based estimator given in Theorem 2.8, which we denote by

$$\overline{S} \triangleq A^S\mathcal{O}^\dagger\widetilde{R}\mathcal{O}^{\dagger\prime}A^{S\prime}+A\widetilde{Q}A'+Q-A^S\mathcal{O}^\dagger\widetilde{T}'A'-A\widetilde{T}\mathcal{O}^{\dagger\prime}A^{S\prime}\,, \quad (2.49)$$

is tight since it is a fixed value. If $\mathcal{O}$ is square and invertible, we have $\overline{S}=\widetilde{S}$.

As to be expected, the values of $\widetilde{S}$ and $\overline{S}$ can be quite large since they rely on an infinite previous a priori covariance and inverting out the dynamics, respectively. Regardless of what estimator algorithm we use, let P_{k+1} represent the a priori estimate covariance at time $k+1$. Then, using the observer-based estimator means we have exactly $P_{k+1}=\overline{S}$, whereas using the S-step Kalman filter gives $P_{k+1}\leq\overline{S}$, and in fact P_{k+1} can be much smaller than this bound. Despite yielding a larger P_{k+1}, there are two main advantages to using the observer-based estimator rather than the S-step Kalman filter; the observer-based estimator does not require the computation of S Kalman gains, and no previous estimate needs to be stored. Furthermore, as we show below, even if we use the observer-based estimator, the bound on P_{k+1} can actually be made significantly smaller than $\overline{S}$ by including an additional p measurements in the buffer. For these reasons the estimator algorithm uses the observer-based estimator as described below.

Since the value of $\overline{S}$ can be quite large, as it relies on inverting out the dynamics, we seek a method to reduce the bound on the estimator covariance after a packet is received. This is accomplished by including a total of $S+p$ measurements in the transmitted packet. The first S measurements, $\{y_{k-S-p+1}, y_{k-S-p+1}, \ldots, y_{k-p}\}$, are used to construct the estimate $\overline{x}^\dagger_{k-p+1}$ according to Eqn. (2.45). The remaining measurements $\{y_{k-p+1}, \ldots, y_k\}$ are then used in running the Kalman filter time and measurement updates, Eqn.s (2.3) – (2.6), initialized with $\hat{x}_{k-p+1|k-p}=\overline{x}^+_{k-p+1}$ and $P_{k-p+1}=\overline{S}$. After running a total of p Kalman filter time and measurement updates we have an estimate $\hat{x}_{k+1|k}$ whose error covariance is given by

$$P_{k+1}=g^p(\overline{S}) \triangleq \overline{M}\,. \quad (2.50)$$

Note that $\overline{M}$ can be much smaller than $\overline{S}$, and in fact as p gets larger $\overline{M}$ approaches the error covariance of the steady state Kalman filter with all measurements received, $\overline{P}$. Note also that since the covariance of the observer-based estimator is a fixed quantity, $\overline{S}$, the subsequent p Kalman gains can be computed off-line and stored in advance; once again this pre-computing of the gains is not

possible if the first S measurements are used with the S-step Kalman filter. We call the estimator just described an observer-based estimator with p-step Kalman filter extension.

Using the observer-based estimator with p-step Kalman filter extension, the estimate covariance is given by Eqn. (2.50) whenever a measurement packet arrives, i.e., if $\gamma_k = 1$, then $P_{k+1} = \overline{M}$. We call the 1-step Kalman filter the algorithm that runs the Kalman filter time and measurement steps using the estimate and variance from the previous time-step and only using the current measurement y_k every time a measurement packet is received. Then, using the 1-step Kalman filter after a sequence of packet receives the covariance might be smaller than $\overline{M}$. Of course, after a sequence of drops, using the 1-step Kalman filter might not produce an estimate with covariance smaller than $\overline{M}$. The 1-step Kalman filter is easier to implement than the observer-based estimator with p-step Kalman filter extension. Therefore, we implement an estimation algorithm that simply performs the 1-step Kalman filter using only the current measurement y_k every time a packet arrives and only decides to run the observer-based estimator with p-step Kalman filter extension if the covariance from the 1-step Kalman filter is greater than $\overline{M}$. The algorithm provides an upper bound on the error covariance that holds whenever a measurement packet is received, i.e.,

$$P_{k+1} \leq \overline{M}, \text{ if } \gamma_k = 1 \,, \tag{2.51}$$

which we need to evaluate the performance metric in Eqn. (2.8). Of course, if no packet arrives $\gamma_k = 0$, then $P_{k+1} = h(P_k)$.

The estimator algorithm consists of running at every time-step k the time update of the Kalman filter, Eqn.s (2.3) – (2.4), using the a posteriori estimate $\hat{x}_{k-1|k-1}$ and variance $P_{k-1|k-1}$ from the previous time-step to produce $\hat{x}_{k|k-1}$ and P_k. If a measurement packet is not received, $\gamma_k = 0$, the algorithm is complete for this time-step. If a measurement packet is received, $\gamma_k = 1$, the Kalman filter measurement update step, Eqn.s (2.5) – (2.6), is run using the most recent measurement y_k to produce an a posteriori estimate $\hat{x}_{k|k}$ and variance $P_{k|k}$. From this, the a priori covariance for the next time step can be computed as $g(P_k)$. If this value is less than $\overline{M}$, then the algorithm is complete for this time-step. If $g(P_k) \not\leq \overline{M}$, then the observer-based estimator with p-step Kalman filter extension yields a better estimate, and the algorithm produces an estimate using this method instead. It uses the oldest S measurements in the packet to compute the $\overline{x}^+_{k-p+1}$ from the observer-based estimator in Eqn. (2.45). Then, it sets $\hat{x}_{k-p+1|k-p} \leftarrow \overline{x}^+_{k-p+1}$ and $P_{k-p} \leftarrow \overline{S}$ and runs the p-step Kalman filter extension with the measurement sequence $\{y_{k-p+1}, \ldots, y_k\}$ to produce the estimate $\hat{x}_k$ and variance $P_{k|k}$. The algorithm is described in Table 2.1.

Table 2.1: Algorithm for estimation scheme.

0) Given A, C, Q, R ;
- Determine $\overline{S}$ and $\overline{P}$;
- Choose the number of additional measurements, p, to buffer and transmit so that $\overline{M} = g^p(\overline{S})$ is as close to $\overline{P}$ as desired and so that $P_k \leq \overline{M}$ holds whenever a packet is received ;
- Initialize $\hat{x}_0$ and P_0 ;

1) Wait for packet at time k ;
- Kalman Filter Time Update ;
- If packet received at time k ;
  - Kalman Filter Measurement Update ;
  - If $P_k \not\leq \overline{M}$;
    - Compute $\overline{x}^+_{k-p+1}$ using Eqn. (2.45) ;
    - Set $\hat{x}_{k-p+1|k-p} \leftarrow \overline{x}^+_{k-p+1}$ and $P_{k-p+1} \leftarrow \overline{S}$;
    - Loop $j = 1$ to p ;
      - Kalman Filter Time and Measurement Updates using measurement y_{k-p+j} ;
    - EndLoop ;
  - EndIf ;
  - $k \leftarrow k + 1$;
- EndIf ;
- Goto 1 ;

2.4 Asymptotic Properties of Error Covariance Matrix

As the simple example in the introduction shows, some events with almost zero probability can make the expected value of the error covariance diverge. In practice, those rare events are unlikely to happen and hence should be ignored. Therefore, the expected value of the error covariance matrix may not be the best metric to evaluate the estimator performance. By ignoring these low probability events, we hope that the error covariance matrix is stable with arbitrarily high probability. This is precisely captured in the following theorem.

Theorem 2.9 *Assume the packet arrival sequences are i.i.d. Let π_g be the expected value of the packet arrival rate. If $\pi_g > 0$, then for any $0 < \epsilon < 1$, there exists $M(\epsilon) < \infty$ such that the error covariance matrix P_k is bounded by M with probability $1 - \epsilon$.*

Though we assume here that the packet drops occur independently, it is shown later when we determine the relationship between M and ϵ that the condition can be relaxed to include the case where the packet drops are described by an underlying markov chain. The theorem also suggests that for a given error tolerance $M > 0$, we can find $\min(\pi_g)$ such that the error covariance matrix P_k is bounded by M with any given specified probability. Before we prove the theorem, we introduce the following proposition.

Proposition 2.10 *Define*

$$\lambda_h(X) = \frac{\mathrm{Tr}(h(X))}{\mathrm{Tr}(X)}.$$

Then,

$$\lambda_h(X) \leq 1 + \lambda_n(A'A) \triangleq \bar{\lambda}_h$$

for all $X > 0$ *such that* $\mathrm{Tr}(X) \geq \mathrm{Tr}(Q)$*, where* $\lambda_n(A'A)$ *denotes the largest eigenvalue of* $A'A$*.*

Proof:

$$\begin{aligned}
\lambda_h(X) &= \frac{\mathrm{Tr}(AXA')}{\mathrm{Tr}(X)} + \frac{\mathrm{Tr}(Q)}{\mathrm{Tr}(X)} \\
&\leq 1 + \frac{\mathrm{Tr}(AXA')}{\mathrm{Tr}(X)} \\
&= 1 + \frac{\mathrm{Tr}(A'AX)}{\mathrm{Tr}(X)} \\
&= 1 + \frac{\mathrm{Tr}(P'A'APP'XP)}{\mathrm{Tr}(P'XP)} \\
&= 1 + \frac{\mathrm{Tr}(SY)}{\mathrm{Tr}(Y)},
\end{aligned}$$

where $S = P'A'AP$ is diagonal and $Y = P'XP > 0$ and has the same eigenvalues as X. Such P exists and $P' = P^{-1}$, as $A'A$ is real symmetric. Hence,

$$\begin{aligned}
\lambda_h(X) &\leq 1 + \frac{\mathrm{Tr}(SY)}{\mathrm{Tr}(Y)} \\
&= 1 + \frac{\sum_{i=1}^n \lambda_i(A'A) Y_{ii}}{\sum_{i=1}^n Y_{ii}} \\
&\leq 1 + \frac{\lambda_n(A'A) \sum_{i=1}^n Y_{ii}}{\sum_{i=1}^n Y_{ii}} \\
&= 1 + \lambda_n(A'A).
\end{aligned}$$

Notice that we implicitly used the fact that $Y_{ii} > 0$ for all i; this follows as

$$Y_{ii} = e_i' Y e_i > 0 \, .$$

■

Now we are ready to prove Theorem 2.9.

Proof: *[Theorem 2.9]* Without loss of generality, assume at time k the packet is not received, $\gamma_k = 0$, otherwise $P_k \leq \overline{M} \triangleq M(\epsilon)$ for any ϵ. Define $\pi_h = 1 - \pi_g$ and let $k' = \max\{s : s \leq k, \gamma_s = 1\}$. Then

$k - k' = N$ with probability $\pi_g \pi_h^N$. Further, define $M_0 = \mathrm{Tr}(P_0)$, $M_1 = \mathrm{Tr}(\overline{M})$, and $\alpha_N = \bar{\lambda}_h^N$. We discuss two cases for a given ϵ.

1. $0 < \epsilon \leq \pi_g$

   Solve the following equation for N,

   $$\pi_g \pi_h^N = \epsilon \, ,$$

   to get

   $$N = \left\lceil \frac{\log \epsilon - \log \pi_g}{\log \pi_h} \right\rceil ,$$

   where $\lceil x \rceil$ denotes the smallest integer that is larger than or equal to x. Assume first that $k \geq N$ so that $k' \geq 0$. Since $\gamma_{k'} = 1$, $P_{k'} \leq \overline{M}$. Therefore,

   $$P_k \leq \alpha_N M_1 I \triangleq M(\epsilon)$$

   with probability $1 - \epsilon$, where I is the identity matrix of appropriate dimension.

   Now consider the case $k < N$; it is easy to see

   $$P_k \leq \alpha_N M_0 I \triangleq M(\epsilon)$$

   with probability at least $1 - \epsilon$.

2. $\pi_g < \epsilon \leq 1$.

   Assume first $k \geq 2$. Let $N = 1$ so that $k' = k - 1$, i.e., , the previous packet is received and $P_{k-1} \leq \overline{M}$. Then,

   $$P_k \leq \alpha_1 M_1 I \triangleq M(\epsilon)$$

   with probability at least $1 - \epsilon$. When $k = 1$,

   $$P_k \leq \alpha_1 M_0 I \triangleq M(\epsilon)$$

   with probability at least $1 - \epsilon$.

■

2.5 Determining the M-ϵ Relationship

It is apparent that $M(\epsilon)$ given in Theorem 2.9 is very conservative, and we seek a tighter bound for the expression

$$\mathbf{Pr}[P_k \leq M] = 1 - \epsilon \, . \tag{2.52}$$

We begin by finding an upper bound on ϵ given M. Recall the bound on the error covariance after a packet is received is given by $\overline{M}$, as in Eqn. (2.51). Then, define $\epsilon_i(k)$ as the probability that at least the previous i consecutive packets are dropped at time k, i.e.,

$$\epsilon_i(k) = \Pr[N_k \geq i] \; , \tag{2.53}$$

with N_k the number of consecutive packets dropped at time k. Note that $N_k = (1-\gamma_k)(1+N_{k-1})$. Clearly $\epsilon_i \geq \epsilon_j$ for $i \leq j$. Next define

$$k_{\min} \triangleq \min \{k \in \mathbb{Z}^+ : h^k(\overline{M}) \not\leq M\} \; . \tag{2.54}$$

Lemma 2.11 *For $0 \leq M < \infty$, the quantity $k_{\min}$ always exists.*

Proof: To prove the existence of $k_{\min}$, note that for any $X > 0$, $\lim_{k\to\infty} \mathrm{Tr}(h^k(X)) = \infty$ if A is unstable. Thus, for any scalar $t > 0$ there exists a $k_{\min}$ such that $h^{k_{\min}}(\overline{M}) \not\leq tI$, and t can be chosen such that $tI \geq M$. This means $\lambda_n(h^{k_{\min}}(\overline{M})) > t$ and $\lambda_n(M) < t$, where λ_n is the maximum eigenvalue. Then, using Weyl's Theorem [26] we see $\lambda_n(h^{k_{\min}}(\overline{M}) - M) \geq \lambda_n(h^{k_{\min}}(\overline{M})) - \lambda_n(M) > 0$, which implies $h^{k_{\min}}(\overline{M}) \not\leq M$. ■

Theorem 2.12 *For unstable A, assume the initial error covariance matrix P_0 is given by $P_0 \leq \overline{M}$. Given a matrix bound $M \geq \overline{M}$, then we have the following lower bound*

$$\mathbf{Pr}[P_k \leq M] = 1 - \epsilon \geq 1 - \epsilon_{k_{\min}}(k) \; . \tag{2.55}$$

That is, the probability only depends on the number of consecutive packets dropped at the current time and is independent of the packet drop/receive sequence prior to the previous received packet.

Proof: Since $P_0 \leq \overline{M}$, then assuming the next k packets are dropped we have $P_k = h^k(P_0)$, and it is clear that $P_0 \leq \overline{M} \Rightarrow h^k(P_0) \leq h^k(\overline{M})$, so

$$P_k \leq h^k(\overline{M}) \; .$$

So, the necessary condition that $P_k \not\leq M$ is

$$h^k(\overline{M}) \not\leq M \; ,$$

but by definition $h^k(\overline{M}) \leq M \; \forall k < k_{\min}$. Thus, for $P_k \not\leq M$ it is necessary to drop at least the previous $k_{\min}$ consecutive packets.

Now assume a packet is not received until time $m > k_{\min}$, that is $\gamma_k = 0$ for $k = 0, \cdots, m-1$ and $\gamma_m = 1$, then $P_{m+1} \leq \overline{M}$ from Eqn. (2.51). Thus, for a packet received at time m we have

$$P_{m+1} \leq \overline{M} \; . \tag{2.56}$$

Regardless of how large m is, i.e., how long between packet receives, and how large the error covariance gets, Eqn. (2.56) holds. Hence, the analysis above can always be repeated with P_{m+1} replacing P_0, and the probability $P_k \nleq M$ depends only on the number of consecutive packets dropped and is independent of what happens prior to the last packet received. ■

Now we also establish an upper bound on $1 - \epsilon$ that is valid under certain conditions. Recall $\overline{P}$ is the solution to the Riccati equation, $g(\overline{P}) = \overline{P}$. The extra condition we require to establish a lower bound on ϵ is that the relation

$$\overline{P} < \overline{M} \tag{2.57}$$

holds. Now define

$$k_{\max} \triangleq \min \{k \in \mathbb{Z}^+ : h^k(\overline{P}) > M\} \; . \tag{2.58}$$

Lemma 2.13 *It is always true that $h(\overline{P}) \geq \overline{P}$, which implies $h^{k+1}(\overline{P}) \geq h^k(\overline{P})$.*

Proof: Since $\overline{P}$ is the solution to the DARE, we can write

$$\begin{aligned} \overline{P} &= g(\overline{P}) \\ &= A\overline{P}A' + Q - A\overline{P}C'(C\overline{P}C' + R)^{-1}C\overline{P}A' \\ &\leq A\overline{P}A' + Q \\ &= h(\overline{P}) \; . \end{aligned}$$

With $h(\overline{P}) \geq \overline{P}$, if we apply h to both sides k times, we get $h^{k+1}(\overline{P}) \geq h^k(\overline{P})$. ■

Lemma 2.14 *If A is purely unstable, i.e., all the eigenvalues of A have magnitude larger than 1, then $k_{\max}$ is guaranteed to exist.*

Proof: If A is purely unstable, then $\lim_{k \to \infty} \lambda_{\min}(h^k(X)) = \infty$. Thus, we can again pick any finite scalar $t > 0$ such that $tI > M$ and find a $k_{\max}$ such that $h^{k_{\max}}(\overline{P}) \geq tI > M$. ■

Lemma 2.15 *With the definitions above, if $k_{\min}$ and $k_{\max}$ both exist, then $k_{\min} \leq k_{\max}$.*

Proof: This can easily be shown by contradiction. Assume $k_{\min} > k_{\max}$. By assumption, $\overline{P} < \overline{M}$, implying $h^{k_{\max}}(\overline{P}) < h^{k_{\max}}(\overline{M})$, and if $k_{\min} > k_{\max}$, then $h^{k_{\max}}(\overline{M}) \leq M$. From the definition of $k_{\max}$, however, we see $h^{k_{\max}}(\overline{P}) > M$, which is a contradiction of the previous inequality. Hence it must be true that $k_{\min} \leq k_{\max}$. ■

Corollary 2.16 *If A is purely unstable and assuming $\overline{P} \leq P_0 \leq \overline{M}$, then we have the upper bound*

$$\mathbf{Pr}[P_k \leq M] = 1 - \epsilon \leq 1 - \epsilon_{k_{\max}}(k) \ . \tag{2.59}$$

Proof: Following the proof of Theorem 2.12, assume the first k packets are dropped so $P_k = h^k(P_0)$. A sufficient condition for $P_k \nleq M$ is then

$$h^k(\overline{P}) > M \ ,$$

since $P_k = h^k(P_0) \geq h^k(\overline{P})$. By definition, $h^k(\overline{P}) > M$ first holds when $k = k_{\max}$. Then, since $h^{k+1}(\overline{P}) \geq h^k(\overline{P})$, it also holds for $k > k_{\max}$. Thus, dropping at least the previous $k_{\max}$ consecutive packets guarantees $P_k \nleq M$. Now assume a packet is not received until time $m > k_{\max}$; then we know $P_m = h^m(P_0) \geq h^m(\overline{P}) > M$ and $\overline{P} \leq P_{m+1} \leq \overline{M}$, so the analysis is repeated with P_{m+1} replacing P_0 as before. ■

The following example can help visualize the concepts of the theorem.

Example 2.17 Consider the scalar system $A = 1.3$, $C = 1$, $Q = 0.5$, and $R = 1$. For this system we have $\overline{P} = 1.519$, and with $S = 1$ and picking $p = 0$ we get $\overline{M} = 2.19$. Picking $M = 6.25$, it is easy to show $k_{\min} = 2$ and $k_{\max} = 3$. Thus, there exists an X^* with $\overline{P} < X^* \leq M$ such that for all $\overline{P} \leq X < X^*$ it requires 3 consecutive packets to be dropped before the error covariance is greater than M, while for the region $X^* \leq X \leq \overline{M}$ it only requires 2 consecutive packets be dropped. In fact it can be easily shown that $X^* = 1.7174$.

Figure 2.2 shows the evolution of the error covariance for a particular sequence of packet drops. The sequence used is $hhhhggghhghhhgh(P_0)$. As can be seen, it requires at least 2 consecutive packets be dropped for the error covariance to rise above the bound.

Remark 2.18 With the definition of $\epsilon_i(k)$ in Eqn. (2.53) it is easy to see $\epsilon_i(k) = 0$ for all $k < i$, which leads to $\Pr[P_k \leq M] = 1$, $\forall k < k_{\min}$. In addition at least $S + p$ measurements are required for the estimator algorithm to be implemented and Eqn. (2.51) to hold. Therefore, we only consider time greater than $\max\{k_{\min}, S + p\}$.

For these results to be useful we need to calculate $\epsilon_i(k)$. Figure 2.3 shows all possible packet sequences at time k for a packet dropping network. From this it is clear to see that $\epsilon_i(k)$ is the sum of the probabilities of each of the instances with at least the previous i packets dropped occurring.

Corollary 2.19 *For $k > i$, any packet dropping network that is either i.i.d. or reaches a steady state (for example a Markov network), $\epsilon_i(k) = \epsilon_i$ is independent of k.*

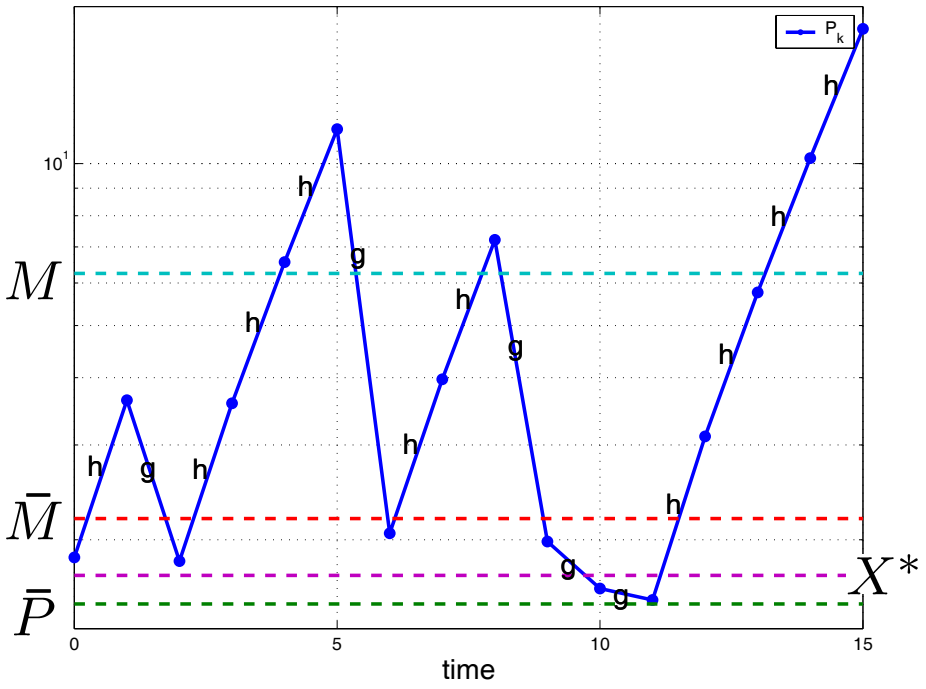

Figure 2.2: Error covariance (*log* scale) for Example 2.17.

The above corollary says the probability of dropping at least the previous i packets is the same for all time. To calculate $\epsilon_{k_{\min}}$ (or $\epsilon_{k_{\max}}$), we can make use of the Markov chain model in Figure 2.4. The states of the Markov chain represent the number of consecutive packets dropped at the current time, and the final state represents $k_{\min}$ or more consecutive packets dropped. The transition probability from state i to state j is given by $T_{i,j}$. It is clear $\epsilon_{k_{\min}} = \pi_{k_{\min}}$, the steady state probability of the Markov chain being in state $k_{\min}$. This is easily determined to be given by

$$\pi_{k_{\min}} = \frac{D}{D + T_{k_{\min},0} + T_{k_{\min},0} \sum_{l=1}^{k_{\min}-1} \prod_{j=0}^{l-1} T_{j,j+1}} , \tag{2.60}$$

with

$$D = 1 - T_{0,0} - \sum_{l=1}^{k_{\min}-1} T_{l,0} \prod_{j=0}^{l-1} T_{j,j+1} .$$

Note that $\pi_{k_{\min}}$ decreases as $k_{\min}$ increases. The same formula holds for $k_{\max}$ by replacing $k_{\min} \leftarrow k_{\max}$.

The $T_{i,j}$ are determined based on the type of network. For example, an i.i.d. network with packet arrival rate γ and drop rate $1-\gamma$ has $T_{j,0} = \gamma \ \forall j \geq 0$, $T_{j,j+1} = 1-\gamma \ \forall j \geq 0$, and $T_{k_{\min},k_{\min}} = 1-\gamma$. This leads to $\pi_{k_{\min}} = (1-\gamma)^{k_{\min}}$. A first order Markov network with transition probabilities T_{hh}, T_{hg}, T_{gh}, and T_{gg} leads to $\pi_{k_{\min}} = \frac{1-T_{gg}}{2-T_{hh}-T_{gg}} (T_{hh})^{k_{\min}-1}$. The probability $\pi_{k_{\min}}$

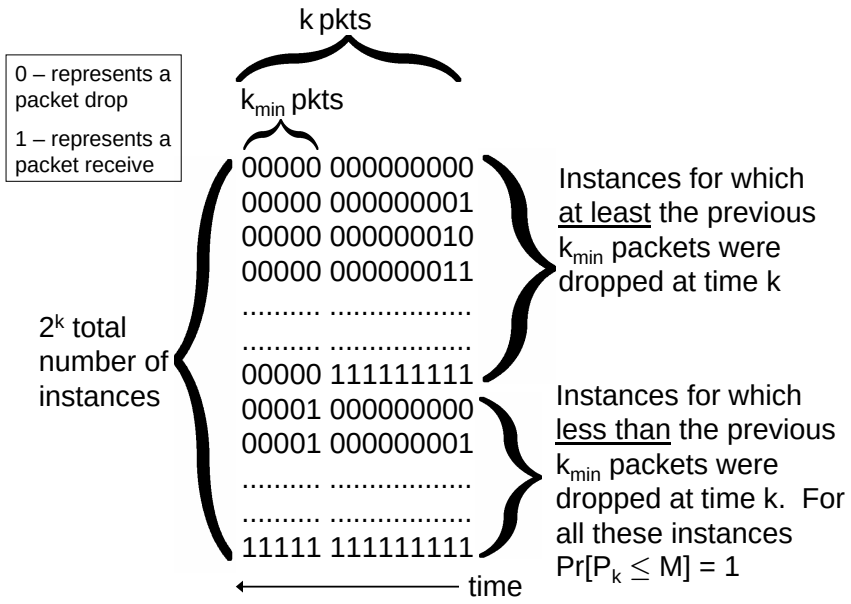

Figure 2.3: A binary representation of the possible packet sequences (i.e., drop/receive) at time k.

for any arbitrary order Markov network can be determined in this manner. All the equations above are valid for calculating $\epsilon_{k_{\max}}$ as well by simply replacing $k_{\min} \leftarrow k_{\max}$

Theorem 2.12 and Corollary 2.16 provide bounds on ϵ for a given M and the network properties, i.e., $\pi_{k_{\min}}$ and $\pi_{k_{\max}}$. It is also possible to determine bounds on M and $\pi_{k_{\min}}$.

Corollary 2.20 *With the same assumptions as Theorem 2.12 and given the transition probabilities $T_{i,j}$ of the Markov model in Figure 2.4 and a lower bound $1 - \epsilon_{k_{\min}}$, it is possible to determine a suitable M such that $Pr[P_k \leq M] \geq 1 - \epsilon_{k_{\min}}$. To do so, define*

$$k_M \triangleq \min\left\{k \in \mathbb{Z}^+ : \pi_k \leq \epsilon_{k_{\min}}\right\}, \tag{2.61}$$

with π_k given in Eqn. (2.60). Then the tightest such bound is

$$M = h^{k_M}(\overline{M}) \, . \tag{2.62}$$

Corollary 2.21 *Likewise, given M and a lower bound $1 - \epsilon_{k_{\min}}$ it is possible to determine limits on the transition probabilities $T_{i,j}$ of the Markov model in Figure 2.4 such that $Pr[P_k \leq M] \geq 1 - \epsilon_{k_{\min}}$. With $k_{\min}$ as defined in Eqn. (2.54), it is easy to see that we require*

$$\pi_{k_{\min}} \leq \epsilon_{max} \, . \tag{2.63}$$

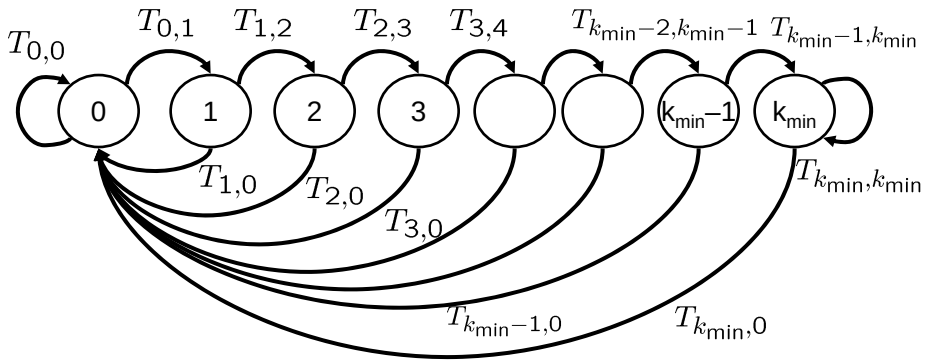

Figure 2.4: The states of the Markov chain represent the number of consecutive packets dropped at the current time, the final state represents $k_{\min}$ or more consecutive packets dropped.

For the i.i.d. network this reduces to $\gamma \geq 1 - \epsilon_{max}{}^{\frac{1}{k_{\min}}}$.

2.6 Simulation Example

Consider the linearized pendubot system in [67] with

$$A = \begin{bmatrix} 1.001 & 0.005 & 0.000 & 0.000 \\ 0.35 & 1.001 & -0.135 & 0.000 \\ -0.001 & 0.000 & 1.001 & 0.005 \\ -0.375 & -0.001 & 0.590 & 1.001 \end{bmatrix}, \; B = \begin{bmatrix} 0.001 \\ 0.540 \\ -0.002 \\ -1.066 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \; R = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix},$$

and

$$Q = qq', \; q = \begin{bmatrix} 0.003 \\ 1.000 \\ -0.005 \\ -2.150 \end{bmatrix},$$

and an i.i.d. network with packet arrival rate $\gamma = 0.75$. For this system we have $S = 2$, meaning $[C', A'C']'$ is full rank, and we need to transmit at least 2 measurements at each time-step. Using the analysis presented in this chapter, we can predict the probability that the error remains below certain bounds. The value of the trace of $\overline{M}$ as a function of the number of additional measurements to buffer, p, is shown in Figure 2.5; notice how $Tr(\overline{M})$ approaches $Tr(\overline{P})$ as $p \to \infty$. This curve can be used as a guide to pick the number of measurements to buffer. Note the error covariance if all

measurements were received, $\overline{P}$, has a trace of 16.27. For the simulations presented below we use $p = 7$ (so we transmit a total of $S + p = 9$ measurements), which gives $Tr(\overline{M}) = 16.99$.

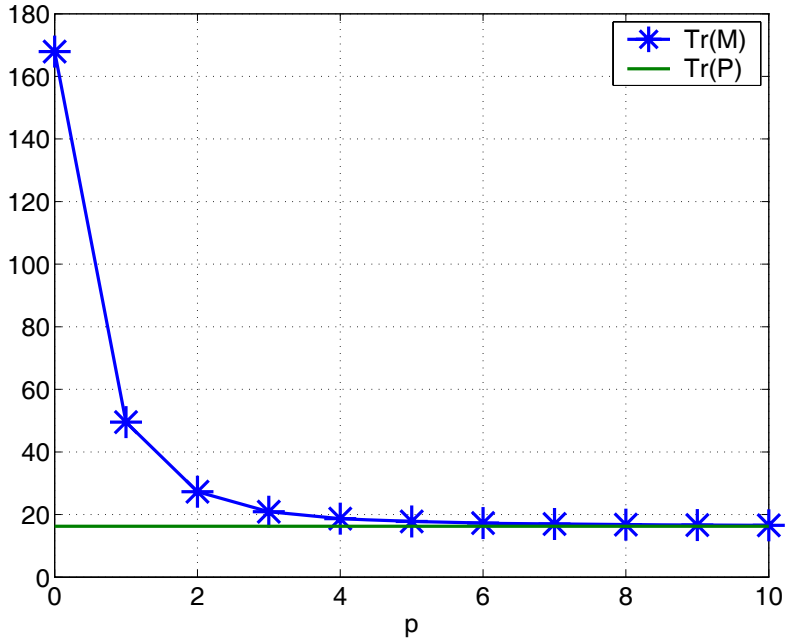

Figure 2.5: The trace of the $\overline{M}$ bound vs. p.

Figure 2.6 shows the M - ϵ relationship for this system. A total of 10,000 simulations are run with a random initial error covariance in the range $\overline{P} \leq P_0 \leq \overline{M}$ chosen for each simulation. The simulations are run for 500 time steps, and the $1-\epsilon$ value calculated from the simulations corresponds to the average over all simulations of the percent of time the error covariance was larger than the M bound. The staircase-like plot can be explained by the fact the probability bounds for $1 - \epsilon$ are given by $1 - \epsilon_{k_{\min}}$ and $1 - \epsilon_{k_{\max}}$, which exhibit sharp jumps, i.e., the staircase, as $k_{\min}$ and $k_{\max}$ change integer values.

2.7 Conclusions and Future Work

In this chapter the problem of state estimation where measurement packets are sent across a lossy network was analyzed. An estimator algorithm that relies on transmitting the current and several previous sensor measurements was designed to guarantee an upper bound on the estimation error covariance whenever a measurement packet is received.

We showed that with this upper bound, as long as the information gain is not exactly equal to zero, then for any given $0 < \epsilon < 1$ there exists an $M(\epsilon) < \infty$ such that the error covariance matrix P_k is bounded by M with probability $1 - \epsilon$. This analysis is independent of the probability distribution of packet drops. Next, we gave explicit relations for upper and lower bounds on the

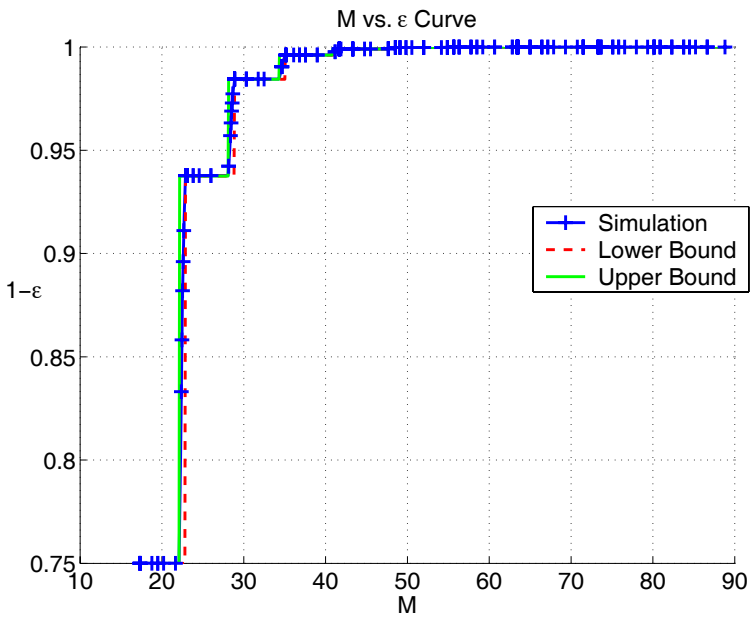

Figure 2.6: M bound vs. ϵ. The (blue) line with tick marks is the simulated $1-\epsilon$ and the dashed and solid (red and green) lines are the predicted $1-\epsilon_{\max}$ and $1-\epsilon_{\min}$.

probability $1-\epsilon_{\min} \leq \mathbf{Pr}[P_k \leq M] \leq 1-\epsilon_{\max}$. We observe that $P_k \not\leq M$ only if a large enough consecutive burst of packets are dropped before time k. The size of the required burst is dependent on M.

The new probabilistic performance description of the estimator is an area that can be pursued further. How this description of the estimation error couples into the closed loop performance with a controller using these estimates is an interesting topic. Considering how the estimation scheme could be adapted for other plant dynamics, such as nonlinear or uncertain systems, is another area to investigate.

Chapter 3

Using Actuation Buffers in Networked Control Systems to Reduce Transmission Frequency of Control Signals

3.1 Introduction

In the previous chapter, buffering sensor measurements was shown to aid in improving the performance of state estimation across a lossy network. In this chapter, once again the benefit of using buffers in NCS is considered, but the communication network is now assumed to reside between the controller and actuators. Thus, packets containing the control signals to be applied are transmitted across the network, and the use of actuation buffers is analyzed. Rather than considering a lossy network, however, the buffers are used in an attempt to decrease the frequency of communication between the controller and actuators by utilizing more informative communication packets.

Often the information communicated in NCS is in the form of packets [32]. These packets have a fixed header length and additional space to be used for data. Due to the overhead included in each transmission, if the frequency of transmission is decreased, there is a corresponding savings in bandwidth and decrease in network congestion. Thus, it is desired to send fewer but more informative packets. In this work we are concerned with how much extra information should be included in the packets, when they should be transmitted, and the impacts on closed loop performance.

We assume control values are transmitted across a communication network to the plant; a schematic of this situation is shown in Fig. 3.1. A motivating example for this type of system is similar to [29], where remote vehicles with limited computation are sent trajectory and/or control commands from remote processing units. To make the packets more informative we include not only the control value computed for the current time-step, but also predicted control values for the next N steps in the future. We investigate the closed loop performance as a function of this buffer length

under different communication schemes. First we consider the case that the controller transmits only when the buffer at the actuator is empty. We also introduce a communication protocol, which we call the Input Difference Transmission Scheme (IDTS), that calculates a new control sequence at every time-step but only transmits to the actuator when the difference between the new sequence and the sequence in the buffer is larger than a certain threshold. We show how this scheme can improve the closed loop performance and provide more flexibility for the system designer.

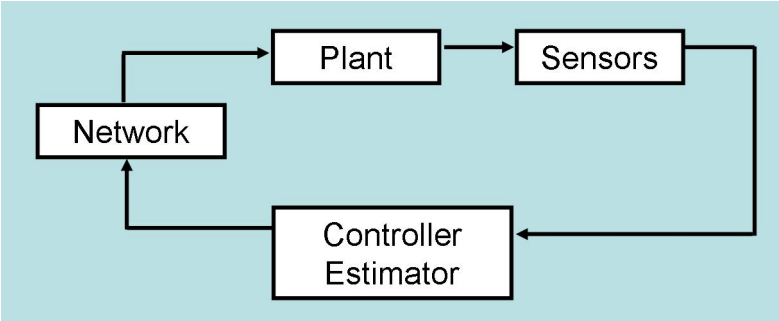

Figure 3.1: NCS feedback loop with control commands sent across the network.

Other researchers have studied ways to determine access to a shared communication medium by nodes of actuators and sensors. In [92] they provide conditions for a stabilizing communication sequence to exist and an algorithm to construct such a sequence. This uses a static scheduling protocol and assumes zero values when transmission does not occur, rather than incorporating some form of an estimator or buffer. A similar setting is considered in [53], but they include estimators on the receiving side of every network transmission to estimate the value of a signal when it is not transmitted. In addition to a static transmission policy, they study granting access to the nodes whose difference between estimated and true signal is largest. The problem setting is slightly different from the one considered here since they consider continuous plants and are not concerned with limiting transmission frequency. In [89] an optimal communication logic is developed to strike a balance between closed loop error and communication rate. The logic that results is similar to the IDTS in this chapter; data only transmits when an estimated state differs from a true state by more than a specified amount; however, they transmit state information rather than control values and thus make no use of a control buffer.

The notion of using a buffer with predicted control values in a NCS setting is not necessarily new, for example see [71, 64], however not much has been investigated relating the length of the buffer to the closed loop performance, especially in terms of reducing the communication frequency. In [21] the authors consider a similar setting, but their analysis of performance as a function of the buffer size is affected by the discretization sampling time of a continuous time plant. Furthermore, they do not consider the effect on the transmission frequency. In [31] the authors consider transmitting a

packet that contains future control signals, but they are not concerned with limiting the transmission frequency, only reducing the effect of communication losses.

The remainder of the chapter is organized as follows. In Section 3.2 a mathematical description of the problem setting is given. The communication protocol is introduced and analyzed in Section 3.3. A simulation example is shown in Section 3.4. Finally, the chapter concludes with a summary of the work and future directions in Section 3.5. This work is joint with Ling Shi, Stefano Di Cairano, and Richard M. Murray, published in [13].

3.2 Problem Set Up

We consider discrete time linear time invariant systems of the form

$$x_{k+1} = Ax_k + Bu_k + w_k \ , \tag{3.1}$$

where $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^r$ is the control input, and $w_k \in \mathbb{R}^n$ is an unknown but bounded disturbance. We also assume a bound on the initial condition so that

$$\begin{aligned} \|w_k\| &\leq \delta_w \ , \ \forall k \\ \|x_0\| &\leq \delta_x \ . \end{aligned}$$

Further, we assume A is unstable, the pair (A, B) is controllable, and that a feedback gain F is designed so that in the absence of the network the control signal would be $u_k = Fx_k$ and $(A+BF)$ is stable. We consider the case where the control signal u_k arriving at the actuators/plant is transmitted across a network from a remotely-located controller, as shown in Fig. 3.1. We ignore delay, quantization, and lost information effects of the network.

The information is transmitted in a packet containing a fixed amount of overhead. As discussed earlier, it is advantageous to put more information into a single packet and reduce the frequency of transmission. We use an anticipative controller that transmits a sequence of control steps each time a packet is sent to plant. In [48] the authors use an anticipative continuous-time controller as a way to reduce the negative effects introduced by network delays, while our aim is to use it to reduce transmission frequency. In addition, we propose different methods for determining when to transmit the packets. The overall performance of the system that we consider is the closed loop error and the frequency of transmission of the control packets, with a desire to keep both of these quantities low. The tradeoff is that lowering the transmission frequency can increase the error. We analyze how the closed loop performance varies with the length of the control sequences transmitted and by using different transmission protocols.

At every instance in time, a control signal for the current time and any future time can be computed based on the current state. Denote the control signal to be applied at time $k+j$ but computed at time k by $u_{k+j|k}$, $j = 0, 1, \ldots$. Note the information available to the controller when calculating $u_{k+j|k}$ is $\mathcal{I}_k = \{\mathcal{I}_{k-1}, x_k, u_{k-1}\}$. We consider a controller that at every time instant computes a control signal for the current time and $N \geq 0$ time steps in the future, i.e., $\{u_{k|k}, u_{k+1|k}, \ldots, u_{k+N|k}\}$. The information packet transmitted from the controller to the plant at time k is exactly this control sequence

$$\mathbf{U}_k = \{u_{k|k}, u_{k+1|k}, \ldots, u_{k+N|k}\} . \tag{3.2}$$

Denote the elements of the packet by $\mathbf{U}_k(j) = u_{k+j-1|k}$, $j = 1, 2, \ldots, N+1$. With this scheme each control packet contains $r \cdot (N+1)$ data points.

When the plant receives packet $\mathbf{U}_k$, it discards all previously-buffered commands and follows the current control sequence. If a packet is not received, the plant applies the corresponding control signal in the buffer from the last previously-received packet. For example, assume that at time $k+M$ the last previously-received packet was that from time k; then, the control signal applied to the plant would be $u_{k+M} = u_{k+M|k} = \mathbf{U}_k(M+1)$. Since we have assumed a finite packet length, $N < \infty$, we must decide what to do when $M > N$. In this case, the last previously-received packet only contains control signals up to time $u_{k+N|k}$, so we must choose what control signal the plant applies for time $u_{k+N+j}, j = 1, 2, \ldots$. There are two obvious possibilities: apply zero control, $u_{k+N+j} = 0$, or hold the control from the last command in the sequence, $u_{k+N+j} = u_{k+N|k}$.

The control applied to the plant at time $k+M$ assuming the last transmitted packet was sent at time k is

$$u_{k+M} = \begin{cases} u_{k+M|k} = \mathbf{U}_k(M+1) \text{ if } M \leq N \\ \lambda u_{k+N|k} = \lambda \mathbf{U}_k(N+1) \text{ otherwise} \end{cases} , \tag{3.3}$$

where $\lambda \in \{0, 1\}$ indicates if the choice is to use zero control ($\lambda = 0$) or hold the last command ($\lambda = 1$). If a state feedback controller is used with the future controls signals based on the predicted evolution of the system, then we see

$$\mathbf{U}_k = \{Fx_k, F(A+BF)x_k, \ldots, F(A+BF)^N x_k\} . \tag{3.4}$$

3.3 Transmit Protocol

To reduce the amount of traffic on the network, the controller does not transmit every control packet $\mathbf{U}_k$. There are several options for determining when to transmit the control packet; they are explored below.

3.3.1 Fixed Transmission Time

The simplest scheme to implement is that with a fixed transmission time. Given that the length of the control buffer is N, so that each control packet contains the current control signal and the next N predicted control signals, if the packet is transmitted at time k, then the control buffer is not empty until time $k+N+1$. Thus, if the control sequence is transmitted every $N+1$ time steps, the actuator always has a control signal to apply. We transmit the first control packet $\mathbf{U}_0$; thus, the packets to transmit are $\{\mathbf{U}_0, \mathbf{U}_{N+1}, \mathbf{U}_{2\cdot(N+1)}, \ldots\}$.

If the control sequence is transmitted more frequently, there would be some elements of the transmitted control buffer that would never be implemented, and these would be unnecessary to include. For example, if the packet that was sent at time k contained $u_{k+N+j|k}, j = 1, 2, \ldots$, these control signals would never be applied since at time $k+N+1$ a new packet is transmitted to the actuator containing $u_{k+N+1|k+N+1}$, and this is applied to the plant. Thus, we set the time between transmissions equal to the number of control signals included in each packet, $N+1$, and call this the Fixed Transmission Time scheme.

Lemma 3.1 *The fixed transmission scheme with buffer length N, using state feedback gain F, has closed loop performance bounded by*

$$\|x_k\| \leq \|(A+BF)^k\|\delta_x + g(A,B,F,N,k)\ \delta_w\ , \tag{3.5}$$

with the effect of the noise terms being accounted for in

$$g(A,B,F,N,k) = \sum_{j=0}^{h(k-1,N)} \left\|A^j\right\| + \left(\sum_{j=0}^{\lfloor \frac{k-1}{N+1} \rfloor - 1} \left\|(A+BF)^{s(j,k,N)}\right\|\right)\left(\sum_{j=0}^{N}\left\|A^j\right\|\right)\ , \tag{3.6}$$

where

$$\begin{aligned} h(k,N) &= mod(k, N+1) \\ s(j,k,N) &= (N+1)j + h(k-1,N) + 1\ , \end{aligned}$$

and $\lfloor\ \rfloor$ is the floor operator.

Proof: The control packet is transmitted every $N+1$ time steps, i.e., whenever $h(k,N) = 0$, the packet $\mathbf{U}_k$ is transmitted. This allows us to write the control applied to the plant as $u_k = F(A+BF)^{h(k,N)}x_{k-h(k,N)}$, and the closed loop evolution as

$$x_{k+1} = Ax_k + BF(A+BF)^{h(k,N)}x_{k-h(k,N)} + w_k\ . \tag{3.7}$$

It is not too difficult to use this to express the closed loop state, for $k \geq 1$, as

$$x_k = (A+BF)^k x_0 + \sum_{j=0}^{h(k-1,N)} A^j w_{k-j-1} + \sum_{j=0}^{\lfloor \frac{k-1}{N+1} \rfloor - 1} (A+BF)^{s(j,k,N)} \, d(j,k,N) \tag{3.8}$$

where

$$d(j,k,N) = \sum_{i=0}^{N} A^i w_{k-s(j,k,N)-i-1} \; .$$

The first term in Eqn. (3.8) accounts for the initial condition. The second term is from the noise acting on the system in the time since the last transmitted control sequence; this noise cannot be compensated for by the current control sequence. The last term accounts for all the noise prior to the last transmitted packet; this is compensated for by the current control sequence through the use of $x_{k-h(k,N)}$. From Eqn. (3.8) and the properties of the norm and bound on $\|w_k\|$, we arrive at the upper bound on $\|x_k\|$ in Eqn. (3.5). ■

Note that whenever a packet is transmitted, the control sequence is a function of the initial condition and the previous noise sequences and is only able to compensate for those terms. For example, consider the control packet that is sent at time $k-N$, $\mathbf{U}_{k-N}$. The control signals from this packet are applied between time $k-N$ and time k, and they compensate for the noise sequence prior to time $k-N$ which are manifest in x_{k-N} and, hence, $\mathcal{I}_{k-N}$. The noise terms $\{w_{k-N+1}, \ldots, w_k\}$ are not compensated by the control sequence, and their effect can be amplified by the open loop dynamics; this is the second term on the right hand side of Eqn. (3.8).

The fixed transmit scheme is simple to implement, and it is easy to see that as the buffer length, and hence transmission interval, is increased, the transmission frequency decreases but the bound on the closed loop error increases. This gives a design tradeoff as desired. The potential downside with the fixed communication scheme is that it might not be utilizing the network very efficiently. The control packets are transmitted at fixed points in time regardless if their transmission has a significant impact on the closed loop error. We seek a scheme that can choose whether or not to transmit online and utilize the network resources more efficiently.

3.3.2 Input Difference Transmission Scheme (IDTS)

In this section, we propose a simple algorithm that at each time-step determines if the control packet should be sent. The algorithm computes a sequence of controls at every time step but only transmits this sequence to the plant if the difference between the newly computed control sequence and the last sequence sent to the plant is larger than a certain threshold. This threshold becomes a design parameter, and we analyze its impact on the closed loop performance.

To formalize the scheme, at time $k+M$ the computed sequence of commands is $\mathbf{U}_{k+M}$, and let the last packet sent to the plant be the one sent at time k, $\mathbf{U}_k$. The criterion that determines

whether or not to transmit packet $\mathbf{U}_{k+M}$ is based on the norm of the difference between the two control signals. Define

$$\Delta U_k^{k+M}(j) = U_{k+M}(j) - \begin{cases} U_k(M+j) \text{ if } M+j \leq N+1 \\ \lambda U_k(N+1) \text{ otherwise} \end{cases} \tag{3.9}$$

for $j = 1, \ldots, N$, and as before, λ indicates if using the hold ($\lambda = 1$) or zero ($\lambda = 0$) option when the buffer is exhausted. Let $\alpha_j \geq 0$ be a scaling factor, and define the weighted norm to be

$$\|\Delta U_k^{k+M}\|_{(\infty,\alpha_j)} = \max_j \alpha_j \|\Delta U_k^{k+M}(j)\| \ . \tag{3.10}$$

Next, pick a scalar $\overline{U}$; the controller only transmits the packet $\mathbf{U}_{k+M}$ to the plant if

$$\|\Delta U_k^{k+M}\|_{(\infty,\alpha_j)} > \overline{U} \ . \tag{3.11}$$

Remark 3.2 The 1-norm can easily replace the ∞-norm with slight modification to the results below.

Based on this scheme, the controller transmits if the newly computed control command differs from the command sequence currently in the plant's buffer. In essence, this only utilizes the network resources when the transmission of a control packet has an impact on the system compared to not sending the packet. Additionally, this scheme can be combined with a force send feature, which forces transmission if the buffer at the plant has been exhausted, i.e., the time since the last transmit is greater than the length of the control sequence. The α_j coefficients and $\overline{U}$ value are also available design choices. The new transmission scheme is illustrated in Table 3.1. Next we analyze the performance of this transmission scheme.

Lemma 3.3 *With the IDTS using the state feedback controller so that* $\mathbf{U}_k$ *is given by Eqn. (3.4), then if* $\alpha_1 = 1$*, the norm of the state is bounded according to*

$$\|x_k\| \leq \|(A+BF)^k\|\delta_x + \sum_{j=0}^{k-1} \|(A+BF)^j\| \left(\|B\|\overline{U} + \delta_w\right) \ . \tag{3.12}$$

Proof: Consider that at time k the last packet transmitted was at time $k-M$. The control applied to the plant at time k can be written as

$$u_k = u_{k|k} - (1-\gamma_k) \cdot \Delta U_{k-M}^k(1) \ , \tag{3.13}$$

Table 3.1: Input Difference Transmission Scheme (IDTS).

0)
- Chose design parameters
  - $F, N, \overline{U}, \alpha_j, \lambda$, FORCE_SEND
- Set
  - time step $k \leftarrow 0$
  - last sent indicator to $k^* \leftarrow -N-1$;

1) Measure x_k ;
2) Compute $\mathbf{U}_k = \{u_{k|k}, u_{k+1|k}, \ldots, u_{k+N|k}\}$;
3) Determine $\|\Delta U^k_{k^*}\|_{(\infty,\alpha_j)}$;
4) If ($\|\Delta U^k_{k^*}\|_{(\infty,\alpha_j)} > \overline{U}$)
   OR
   (FORCE_SEND $= 1$ AND $k - k^* > N$) ;
   - Transmit $\mathbf{U}_k$;
   - Set $k^* \leftarrow k$;

   EndIf ;
5) Set
   $k \leftarrow k+1$;
6) Goto 1 ;

where $\gamma_k \in \{0,1\}$ is used to indicate if the control packet $\mathbf{U}_k$ is transmitted or not.

Since we are using a state feedback controller, we have $u_{k|k} = Fx_k$, and we can write Eqn. (3.13) as

$$\begin{aligned} u_k &= Fx_k + z_k \\ z_k &= (1-\gamma_k) \cdot \Delta U^k_{k-M}(1) \; . \end{aligned}$$

If $\gamma_k = 1$, then $\mathbf{U}_k$ is transmitted, and $z_k = 0$. If $\gamma_k = 0$, then the packet is not transmitted; however, using IDTS with $\alpha_1 = 1$ if the packet is not transmitted then we are guaranteed to have $\|z_k\| = \|\Delta U^k_{k-M}(1)\| \leq \overline{U}$. Thus, regardless of the value of γ_k, i.e., independent of the transmission status of $\mathbf{U}_k$, we get that $\|z_k\| \leq \overline{U}$.

The closed loop system can now be rewritten as

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ &= (A+BF)x_k + (Bz_k + w_k) \; , \end{aligned}$$

which is a stable system with a bounded disturbance term $\|Bz_k + w_k\| \leq \|B\|\overline{U} + \delta_w$. The state can then be written as

$$x_k = (A+BF)^k x_0 + \sum_{j=0}^{k-1} (A+BF)^{k-j-1}(Bz_j + w_j) \; ,$$

from which the bound in Eqn. (3.12) directly follows. ■

Remark 3.4 The bound in Eqn. (3.12) is independent of the buffer length N. It is also a worst case analysis, but it allows comparison with the worst case analysis for the fixed transmission scheme. These worst case bounds are conservative due to the use of the norm properties as well as assuming the worst case noise in each step; nonetheless, they can be useful guides.

In addition to computing this upper bound on the state error, we also want to characterize the transmission frequency using IDTS since the number of time-steps between transmissions is no longer fixed. In Lemma 3.3 the condition $\alpha_1 = 1$ was imposed. In fact, if we set $\alpha_j = 0 \; \forall j \geq 2$, the result is unaffected, and the transmit criterion in Eqn. (3.11) becomes

$$\|\Delta U_k^{k+M}(1)\| > \overline{U} \; .$$

This means the decision to transmit the control packet depends only on the difference between the control signal for the current time step and not the future control signals, though they are still transmitted in the packet. Since the condition to transmit is checked at every time step, removing the dependence on the future control signals is less critical and simplifies the analysis. Likewise, we assume a zero control scheme if the buffer runs out, i.e., $\lambda = 0$, which also simplifies the analysis below. Thus, for the remainder of this section we assume

- $\alpha_1 = 1$ and $\alpha_j = 0$ for $j \geq 2$ and
- $\lambda = 0$,

though similar results can be obtained without these assumptions.

In the analysis below we make use of the following quantity. With $m \geq 1$ a positive integer, define

$$L(m,k,N) = \sum_{j=0}^{m-1} A^{m-j-1} w_{k+j} + \delta(m-N-1) A^{m-N-1} (A+BF)^{N+1} x_k \tag{3.14}$$

with

$$\delta(j) = \begin{cases} 0 \text{ , if } j \leq 0 \\ 1 \text{ , if } j > 0 \end{cases} \; .$$

Proposition 3.5 *Given the last transmission occurred at time k, the next transmission is at time $k+M$, where*

$$M = \min \left[\frac{N+1}{\beta}, \min_{m>0} \left\{ m \in \mathbb{Z}^+ : \|F \cdot L(m,k,N)\| > \overline{U} \right\} \right] , \tag{3.15}$$

and β is used to indicate if using the force send feature ($\beta = 1$) or not ($\beta = 0$).

Proof: The packet transmitted at time k is given by Eqn. (3.4). We are interested in the time when the next packet is sent, so we can write the closed loop evolution based on no packet being sent

between k and $k+m$. For $1 \leq m \leq N+1$, the applied control signal is $u_{k+m-1} = F(A+BF)^{m-1}x_k$, and the state is given by

$$x_{k+m} = (A+BF)^m x_k + \sum_{j=0}^{m-1} A^{m-j-1} w_{k+j} \; .$$

No control is applied starting at time $k+N+1$, i.e., $u_{k+i} = 0$ for $i = N+1, N+2, \ldots, m-1$. Thus, for $m > N+1$ the state can be written as

$$x_{k+m} = A^{m-N-1}(A+BF)^{N+1} x_k + \sum_{j=0}^{m-1} A^{m-j-1} w_{k+j} \; .$$

From this and the fact that $u_{k+M|k+M} = F x_{k+M}$, it is easy to see that $\|\Delta U_k^{k+m}(1)\| = \|F \cdot L(m,k,N)\|$. The next transmission occurs at the first instance that $\|\Delta U_k^{k+m}(1)\| > \overline{U}$, or if force send is in effect it occurs at time $k+N+1$ if $m > N+1$. This is exactly the expression captured in Eqn. (3.15). ■

With the IDTS using the force send feature and state feedback control, the number of time-steps between successive transmissions is simply a function of the realization of the noise sequence and the buffer length, it is independent of the state. That is, with the force send feature it is not possible to wait longer than $N+1$ time steps, so we only need to evaluate $\|F \cdot L(m,k,N)\|$ for $m \leq N+1$, and from Eqn. (3.14) we see the x_k term disappears, so it is only a function of the open loop dynamics and realization of the noise sequence from time k. Without the force send feature it is possible to wait longer than $N+1$ time-steps, so we check all $m \geq 1$, and for $m > N+1$ and the state x_k appears in the expression for $L(m,k,N)$, meaning in this case the number of time steps between transmissions depends on the value of the state at the last transmission.

As seen in Proposition 3.5, the design parameter $\overline{U}$ affects the time between transmissions. In fact, it is possible to show how the minimum time between transmissions depends on $\overline{U}$ and δ_w.

Lemma 3.6 *Define*

$$m^* = \min_{m>0} \left\{ m \in \mathbb{Z}^+ : \|F\| \cdot \delta_w \cdot \sum_{j=0}^{m-1} \|A^j\| > \overline{U} \right\} . \tag{3.16}$$

Then, using IDTS with state feedback control, the lower bound on the time between transmissions is given by

$$M^* = \min\left[N+1, m^*\right] . \tag{3.17}$$

Proof: As seen in Proposition 3.5, the key to determining the time between transmissions is the expression

$$\|F \cdot L(m,k,N)\| > \overline{U} \ .$$

For $m \leq N+1$, the second term in Eqn. (3.14) drops out, and we can write

$$\begin{aligned} \|F \cdot L(m,k,N)\| &= \left\| F \cdot \sum_{j=0}^{m-1} A^{m-j-1} w_{k+j} \right\| \\ &\leq \|F\| \cdot \delta_w \cdot \sum_{j=0}^{m-1} \|A^j\| \ . \end{aligned}$$

Thus, $\|F\| \cdot \delta_w \cdot \sum_{j=0}^{m-1} \|A^j\| > \overline{U}$ is a necessary condition for $\|F \cdot L(m,k,N)\| > \overline{U}$, i.e., it requires at least m^* time steps before the noise alone *could* trigger the transmit criterion. If $m^* > N+1$, then the value of the state x_k affects the value of $\|\Delta U_k^{k+m^*}\|$; however, with force send the packet automatically is sent after $N+1$ time steps, and without force send the lower bound of $N+1$ still holds. Hence we arrive at Eqn. (3.17). ■

Remark 3.7 With force send, the transmission time is in the interval $[M^*, N+1]$. Without force send, it is in the interval $[M^*, \infty)$. Thus, if $M^* = N+1$, meaning $m^* \geq N+1$, then with force send the time between transmissions is exactly fixed at $N+1$, i.e., it recovers the fixed transmission scheme.

Remark 3.8 When $N+1 \gg M^*$, the transmission rate is qualitatively the same whether or not force send is used. This behavior is expected since the only difference is that without force send the transmission time can lie in the interval $[N+2, \infty)$. With $N+1 >> M^*$, however, this rarely occurs, and it is more likely to be in $[M^*, N+1]$ for both schemes.

3.4 Example

We consider the plant in Eqn. (3.1) with

$$A = \begin{bmatrix} 1.2 & 0.4 & 1.2 & 1.5 \\ -0.2 & -0.4 & -0.2 & -0.4 \\ 0.1 & -0.2 & 1.6 & 2.0 \\ -0.2 & 0.4 & 1.1 & 1.3 \end{bmatrix}, \ B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} .$$

This has open loop eigenvalues of $[2.79, 1.25, 0.37, 0.092]$. The disturbance is bounded according to $\delta_w = 2$. The state feedback gain is chosen to place the closed loop eigenvalues at $[0.267, 0.234, 0.15, 0.12]$.

A total of $10,000$ simulations of 100 time-steps each are used to generate the noise sequences and initial conditions. The closed loop system is then simulated with the different communication schemes and various buffer lengths.

We use buffer lengths of $N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20\}$ and bounds of $\overline{U} = \{20, 500\}$. With $\overline{U} = 20$ we have $m^* = 2$, i.e., without force send the IDTS always skips at least 1 time step in between transmissions, and with $\overline{U} = 500$ we have $m^* = 5$. In Fig. 3.2 we plot the transmit properties. The top plot shows the percent of time the control packet is transmitted to the plant. The transmit rate is the same for IDTS with force send and the fixed transmission scheme when $N + 1 < m^*$. For IDTS, as $N \gg M^*$ we see that the percent of time transmitting is roughly the same with or without force send and only depends on $\overline{U}$. The controller does not transmit as often as the transmit criterion $\overline{U}$ is increased . The bottom plot is the percent of time no control is applied and the plant evolves completely open loop, i.e., $u_k = 0$. This occurs only when the controller does not transmit a packet and the time since the last transmit is greater than $N + 1$. This can not happen with the force send, so we only plot the cases without force send.

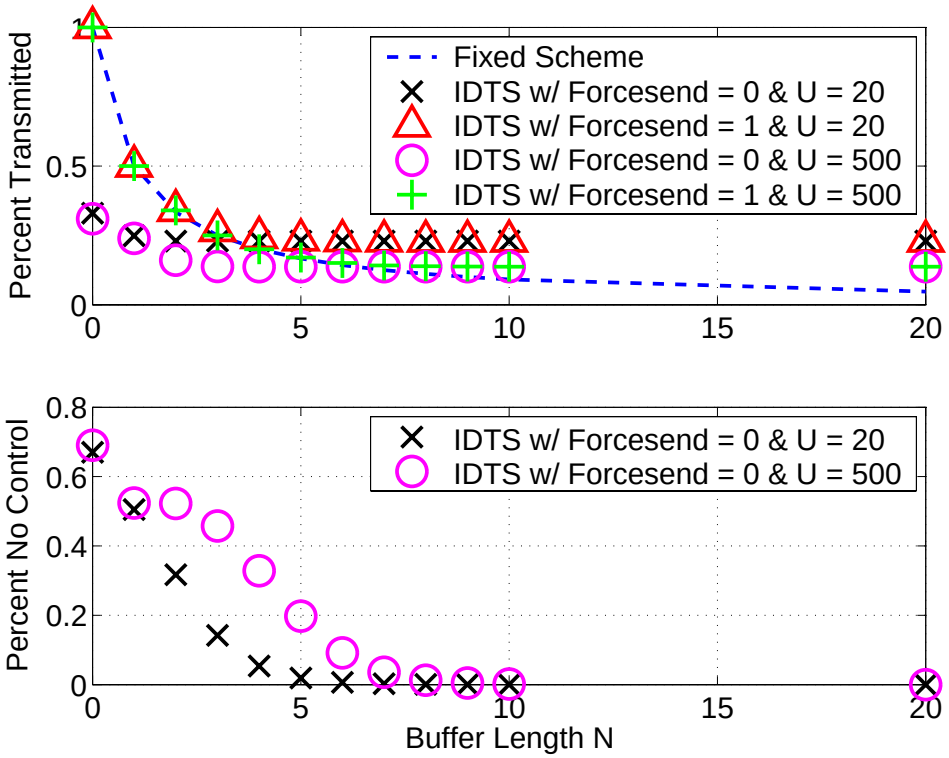

Figure 3.2: Simulation transmission properties.

In Fig. 3.3 the state errors are plotted. The maximum error over all simulation time steps and the theoretical upper bounds are plotted. Again notice that for $N \gg M^*$ the IDTS with and without force send exhibit the same closed loop error characteristics. Notice that for very similar transmit

rates, the IDTS has smaller error compared to the fixed transmission scheme. This is evidence that the IDTS makes more effective use of the network by transmitting the packets when it is deemed important.

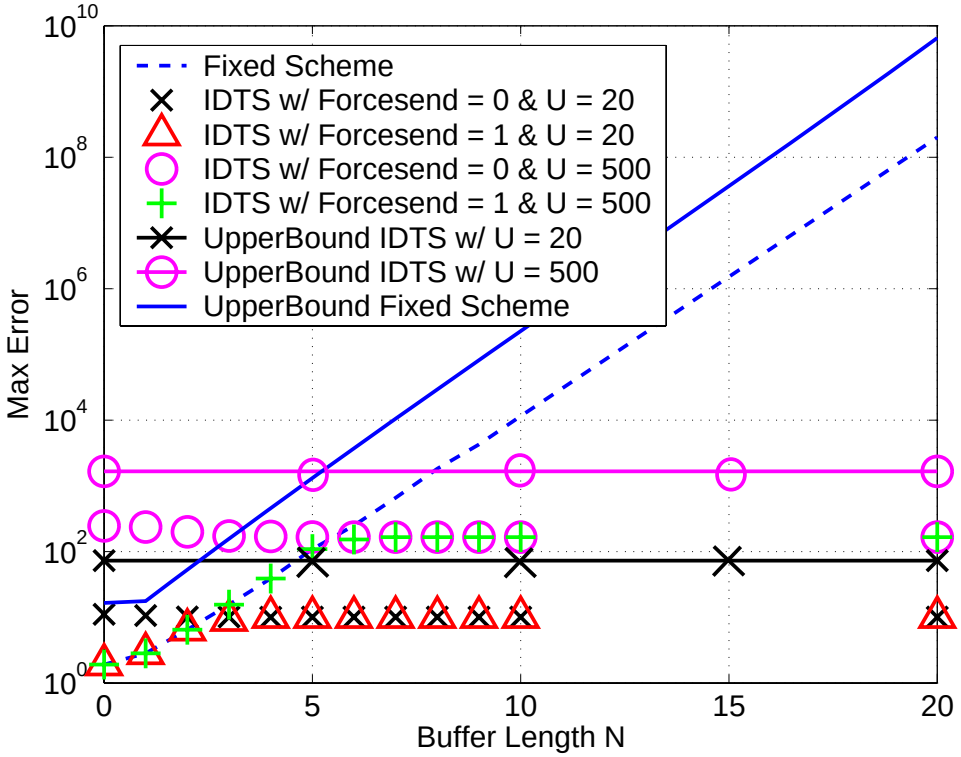

Figure 3.3: Worst case simulation error and theoretical bounds.

3.5 Conclusions and future work

We considered an NCS setting where the control signal is sent across a network to the plant. The goal was to design a system that sends less frequent but more informative information packets. The data in each control packet contains the control signal for the current time-step as well as a buffered sequence of predicted future control signals.

While this initial study introduced the IDTS communication scheme to reduce the transmission frequency and provided some initial insights into the performance characteristics, there is certainly more work that can be done. From the modeling standpoint, one can consider the effect of measurements taken from noisy sensors and using an observer to produce an estimate of the state. Network effects such as lost packets, delays, and quantization were all ignored; it would be interesting to see how the communication schemes presented here work when these scenarios are present and what modifications could be made to compensate for these effects.

In this work it was assumed the state feedback controller, F, was designed without regard to the network considerations. The closed loop properties depend on the gain, for example, a smaller $\|F\|$ can increase the minimum number of steps between transmissions. This relationship could be investigated in further detail. It would also be interesting to consider more general model predictive controllers in place of the anticipative controller.

Chapter 4

Estimation Schemes for NCS Using UDP-Like Transmission of Control Values

4.1 Introduction

This chapter continues the theme of transmitting control values from a controller to a remote plant. Instead of using a buffer of predicted control values to send less frequent but more informative packets, the controller transmits only the current control value and does so at every time-step. The network effect considered here is that the link is unreliable so that the packets could be lost. If a control packet is not received, the plant applies no control and evolve open loop. The majority of NCS research that considers designing a state estimator for this situation assumes the estimator knows whether or not the control packet is received at the plant, via an acknowledgement signal and TCP-like communication protocols, and hence what control signal is applied. The acknowledgement signal allows for the standard separation principle to apply, and the controller and estimator can be designed separately.

This chapter assumes the control packets are sent without an acknowledgement signal, via UDP-like communication protocols. While this makes the analysis more difficult and could require modifications to standard control algorithms, there is evidence that using the UDP-like protocol is advantageous over the TCP-like protocol due to lower latency, less overhead in terms of packet length, and decreased software complexity [61, 9]. Only a small subset of the NCS research to date considers UDP-like communication protocols, where there is no receive acknowledgement, for the network between the controller/estimator and the acutators/plant. In [76, 75] the authors show that in this setting the LQG controller is in general nonlinear and cannot, except for trivial cases, be found in closed form. Other researchers study the effects of using UDP-like communication protocols in NCS [65, 2].

We investigate a particular case of NCS using UDP-like communication to transmit control signals to a remote plant. We assume there is perfect communication between the sensors and the estimator/controller so that the measurement data is always available at the estimator. The lossy network connecting the estimator/controller to the actuators/plant uses UDP-like protocols. This setup is summarized in Fig. 4.1. We present two estimator algorithms for this situation that can both be shown to guarantee closed loop stability with a state feedback controller under certain conditions. The algorithms consist of an estimator, made up of a mode detector and state observer, and state feedback implemented with one of two options:

- using an enlarged control value that guarantees proper detection of the control packet fate or
- using standard state feedback and tolerate possible mis-detections of the control packet but achieve better closed loop performance.

The first option uses a larger control value to guarantee detection of whether or not the control packet is received. This constraint of using a large enough control input to ensure detection is removed in the second option. While this can result in mis-detections of the control packet fate, the system can still be stabilized and in fact can achieve better performance than when using the enlarged signal. These algorithms and their stability conditions are then compared to the well-known unknown input observer.

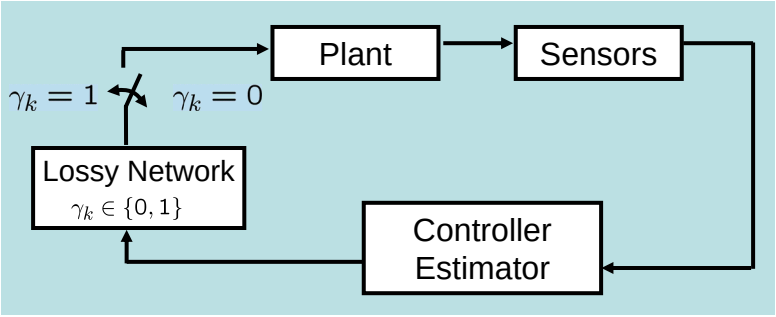

Figure 4.1: NCS feedback loop with control commands sent across a lossy network.

The chapter is organized as follows. In Section 4.2 we set up the problem in a mathematical framework. Naive schemes for building estimators are examined in Section 4.3. The proposed estimator algorithm and its convergence properties are presented in Section 4.4, as is the well-known unknown input observer, which is compared to our algorithm. Simulation examples are given in Section 4.5 to illustrate the theory. The work in this chapter is joint with Ling Shi and Richard M. Murray presented in [14, 15].

4.2 Problem Set Up

We consider a networked control system where the controller sends commands to the actuator across a packet dropping network as in Fig. 4.1. The network is assumed to be following a UDP-like protocol. The controller sends signals to the actuators but does not receive any form of acknowledgement, i.e., the controller does not know if the packet is dropped or not.

The plant we consider is a discrete-time linear system. If the control packet is not received, it is assumed the plant applies no control and evolves open loop. The plant dynamics are

$$x_{k+1} = Ax_k + \gamma_k Bu_k + w_k \tag{4.1}$$

$$y_k = Cx_k + v_k \ , \tag{4.2}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^r$ is the control input, and $y_k \in \mathbb{R}^m$ is the sensor output. The process noise is given by $w_k \in \mathbb{R}^n$ and the measurement noise by $v_k \in \mathbb{R}^m$, which are both assumed to be bounded. The variable $\gamma_k \in \{0, 1\}$ indicates if the packet containing u_k is received at the plant ($\gamma_k = 1$) or if it is dropped ($\gamma_k = 0$). It is assumed if the packet is dropped, the plant applies no control for that time-step, i.e., it evolves open loop. Note also there is no network between the plant/sensors and the estimator/controller, so measurements are always available. We further assume A is unstable, (A, B) is controllable, and (A, C) is observable, so that in the absence of the network F and L are designed to make $A + BF$ and $A - LCA$ stable.

We also make the assumption that $\text{rank}(CB) = \text{rank}(B) = r \leq n$. This is required as we desire to recover the fate of γ_k at time $k + 1$, meaning we need the effect of γ_k to be present in

$$y_{k+1} = Cx_{k+1} + v_{k+1} = CAx_k + \gamma_k CBu_k + d_k \ , \tag{4.3}$$

where

$$d_k = Cw_k + v_{k+1} \ . \tag{4.4}$$

If the rank condition on CB does not hold, then the γ_k would disappear from the y_{k+1} expression. Physically this condition can be interpreted as requiring any states that are directly affected by the input be measured. If this rank condition did not hold but say rank $(CAB) = \text{rank}(B) = r \leq n$, the algorithms presented below could still work with the modification that they would reveal the fate of γ_k at time $k + 2$ instead of time $k + 1$.

Under the UDP-like communication scheme we are considering, the estimator has no knowledge about the value γ_k. Therefore, an observer for this system could take the form of

$$\hat{x}_{k+1} = A\hat{x}_k + \hat{\gamma}_k Bu_k + L(y_{k+1} - CA\hat{x}_k - \hat{\gamma}_k CBu_k) \ , \tag{4.5}$$

where the decision must be made how to select $\hat{\gamma}_k$.

Writing the estimation error as $e_k = x_k - \hat{x}_k$, we see it evolves according to

$$e_{k+1} = (A - LCA)e_k + (\gamma_k - \hat{\gamma}_k)(B - LCB)u_k + z_k \ , \tag{4.6}$$

with

$$z_k = w_k - Ld_k \ . \tag{4.7}$$

Clearly, if $\hat{\gamma}_k = \gamma_k$, then (ignoring the noise terms) the estimation error evolves as $e_{k+1} = (A - LCA)e_k$, which is stable. Recalling the UDP-like communication protocol, however, the estimator receives no acknowledgement and hence does not know the value of γ_k when deciding on $\hat{\gamma}_k$. As a result, the estimator could either try to reason about γ_k (if possible) or simply set $\hat{\gamma}_k$ to a predetermined value. For any system we design, the state evolves open loop when $\gamma_k = 0$, but if we are able to design a system that could recover $\hat{\gamma}_k = \gamma_k$ at time $k+1$, then the estimation error is indifferent to the packet drops and we return to the TCP case and can tolerate a higher percentage of drops. The goal is to design an estimator and a control algorithm that can either recover the fate of γ_k at time $k+1$ or mitigate the effect of choosing an incorrect value for $\hat{\gamma}_k$.

We make two simplifying assumptions for the remainder of this chapter:

- the inputs are of single dimension $u_k \in \mathbb{R}$, i.e., single input systems, and
- the packet drops are independent and identically distributed (i.i.d.) with probability of receiving a packet given by

$$\mathbf{E}[\gamma_k] = \overline{\gamma} \ . \tag{4.8}$$

These assumptions simplify the derivations and analysis below. Single input systems allow for a nicer representation of an input constraint that is derived later in the chapter. When computing the expected value it is easier with i.i.d. packet drops, but it could also be accomplished with, for example, a markov packet dropping model. The estimation algorithm to be presented and all subsequent analysis can also be derived with slight modifications if these assumptions were not present.

4.3 Naive Schemes

The simplest method for choosing $\hat{\gamma}_k$ is to preselect a value. These methods are referred to as the naive schemes, and they are analyzed in this section. Let $\mathbf{X}_k = [x_k', e_k']'$, then the closed loop

dynamics are

$$\mathbf{X}_{k+1} = \left[\begin{array}{c|c} A & 0 \\ \hline 0 & A - LCA \end{array}\right] \mathbf{X}_k + \begin{bmatrix} \gamma_k \\ (\gamma_k - \hat{\gamma}_k)(I - LC) \end{bmatrix} Bu_k + \begin{bmatrix} w_k \\ z_k \end{bmatrix} . \tag{4.9}$$

To simplify the analysis of these naive schemes we ignore the noise terms, i.e., let $w_k = z_k = 0$. For the sake of showing the pitfalls of these schemes, it should be clear that if their flaws are exposed even in the noise free case, they certainly are not suitable when noise is present. When analyzing our algorithm we once again include the noise. Let us assume a part of these naive schemes is to include a state feedback controller $u_k = F\hat{x}_k$.

The closed loop evolution can then be written as a jump linear system (JLS),

$$\begin{aligned} \mathbf{X}_{k+1} &= \mathbf{A}_{\theta(k)} \mathbf{X}_k & (4.10) \\ \mathbf{A}_{\theta(k)} &= \left[\begin{array}{c|c} A + \gamma_k BF & -\gamma_k BF \\ \hline G_{\hat{\gamma}}(k) & A - LCA - G_{\hat{\gamma}}(k) \end{array}\right] & (4.11) \\ G_{\hat{\gamma}}(k) &= (\gamma_k - \hat{\gamma}_k)(B - LCB)F & (4.12) \\ \gamma_k, \hat{\gamma}_k &\in \{0, 1\} . & (4.13) \end{aligned}$$

This jump linear system has four modes corresponding to the different combinations of γ_k and $\hat{\gamma}_k$. Note that if $\hat{\gamma}_k = \gamma_k$, then $G_{\hat{\gamma}}(k) = 0$, and the estimation error evolves as $e_{k+1} = (A - LCA)e_k$, which is clearly stable. With the UDP-like communication protocol, however, the estimator receives no acknowledgement and hence does not know the value of γ_k, and the naive schemes simply preselect a value for $\hat{\gamma}_k$ for all time.

With the assumption of i.i.d. packet drops with $\mathbf{E}[\gamma_k] = \overline{\gamma}$, it might seem logical to set $\hat{\gamma}_k = \overline{\gamma}$ for all k, as was done in [75]. Though that problem setting is slightly different, the approach can be applied here, and the resulting jump linear system has two modes given by

$$\begin{bmatrix} A + \gamma_k BF & \gamma_k BF \\ G_{\overline{\gamma}}(k) & A - LCA - G_{\overline{\gamma}}(k) \end{bmatrix} ,$$

$$G_{\overline{\gamma}}(k) = (\gamma_k - \overline{\gamma})(B - LCB)F ,$$

with $\gamma_k \in \{0, 1\}$. In order for the closed loop to be stable, in some expected sense, clearly at least one of the switching modes must be stable. When $\gamma_k = 0$, the state evolves as $x_{k+1} = Ax_k$, so that mode is unstable. Thus, the only hope is if the mode corresponding to $\gamma_k = 1$ is stable, but with only F and L as design parameters, for $\overline{\gamma} \neq 1$ it may not even be possible to make this matrix stable!

Another option for preselecting $\hat{\gamma}_k$ is to set it to either 0 or 1. Clearly setting $\hat{\gamma}_k = 0$ does not work, as in this case none of the modes are stable. If we let $\hat{\gamma}_k = 1$, then the jump linear system is defined by

$$\begin{bmatrix} A + \gamma_k BF & \gamma_k BF \\ G_1(k) & A - LCA - G_1(k) \end{bmatrix},$$

$$G_1(k) = (\gamma_k - 1)(B - LCB)F\ ,$$

with $\gamma_k \in \{0, 1\}$. When $\gamma_k = 1$, we see the switching mode is stable. Thus far it would appear to be the best option to use as an estimator. The downfall is, of course, when $\gamma_k = 0$ it results in $G_1(k) \neq 0$, and the estimation error can be unstable. As a consequence, any sequence of packet drops causes the estimation error to grow. As one would expect, the estimation error can still converge to zero as $k \to \infty$, but only for values of $\overline{\gamma}$ close to 1.

Thus it would appear none of these naive schemes give suitable performance. Instead an estimation algorithm is presented that can be guaranteed to detect the value of γ_k by enforcing an input constraint and can be shown to provide stability. That is followed by a slight modification to the algorithm that removes conditions that guarantee detection of γ_k, but bound the growth in estimation error following a mis-detect. Conditions are given such that the estimation error is bounded, resulting in a stable closed loop.

4.4 Estimation Algorithm

As stated above, it is clear that we seek an estimator scheme that can recover $\hat{\gamma}_k = \gamma_k$, as this makes the NCS revert to the TCP-like communication protocol. We use an estimator algorithm consisting of the state observer from Eqn. (4.5), and choosing $\hat{\gamma}_k$ according to the mode detector,

$$\hat{\gamma}_k \quad = \quad \arg \min_{\beta \in \{0,1\}} \|y_{k+1} - CA\hat{x}_k - \beta CBu_k\|^2\ . \tag{4.14}$$

Essentially, this compares the residual between the new measurement and two predicted measurements made with the previous estimate of the state and (i) with the control applied ($\beta = 1$) and (ii) without the control applied ($\beta = 0$). Selecting the choice (with control or without) that yields the smaller residual, the mode detector above is shown to recover the true state of γ_k under the conditions below. Note once again the restriction to systems with $\text{rank}(CB) = \text{rank}(B) = r \leq n$ is necessary to ensure the β term appears in the minimization on the right hand side of Eqn. (4.14).

Proposition 4.1 *For the mode detector that chooses $\hat{\gamma}_k$ according to Eqn. (4.14), the following statements hold*

- *If* $u_k{}'B'C'CBu_k > 2\,|u_k{}'B'C'(CAe_k + d_k)|$ *then* $\hat{\gamma}_k = \gamma_k$ *and*
- *If* $\hat{\gamma}_k \neq \gamma_k$ *then* $u_k{}'B'C'CBu_k < 2\,|u_k{}'B'C'(CAe_k + d_k)|$.

Proof: Returning to Eqn. (4.14), we can write

$$\begin{aligned}
\|y_{k+1} - CA\hat{x}_k - \beta CBu_k\|^2 &= \|CAx_k + \gamma_k CBu_k + d_k - CA\hat{x}_k - \beta CBu_k\|^2 \\
&= \|CAe_k + d_k + (\gamma_k - \beta)CBu_k\|^2 \\
&= e_k{}'A'C'CAe_k + d_k{}'d_k + 2d_k'CAe_k \\
&\quad +(\gamma_k - \beta)^2\, u_k{}'B'C'CBu_k + \, 2(\gamma_k - \beta)u_k{}'B'C'(CAe_k + d_k)\,.
\end{aligned}$$

Since $e_k{}'A'C'CAe_k + d_k{}'d_k + 2d_k'CAe_k \geq 0$ and is independent of (γ_k, β), we can remove it from the minimization, leaving

$$J(\gamma_k, \beta) = (\gamma_k - \beta)^2\, u_k{}'B'C'CBu_k + 2(\gamma_k - \beta)u_k{}'B'C'(CAe_k + d_k)\,. \tag{4.15}$$

Recall that $\gamma_k \in \{0, 1\}$ and $\beta \in \{0, 1\}$, so we are left with four possibilities for the value of $J(\gamma_k, \beta)$ as shown in Table. 4.1.

Table 4.1: Possible values of cost function in Eqn. (4.15).

γ_k	β	$J(\gamma_k, \beta)$
0	0	0
0	1	$u_k{}'B'C'CBu_k - 2u_k{}'B'C'(CAe_k + d_k)$
1	0	$u_k{}'B'C'CBu_k + 2u_k{}'B'C'(CAe_k + d_k)$
1	1	0

If $\beta = \gamma_k$, then $J = 0$. Otherwise if $\beta \neq \gamma_k$, then $J = u_k{}'B'C'CBu_k + 2(\gamma_k - \beta)u_k{}'B'C'(CAe_k + d_k)$. Clearly since J is being minimized, a sufficient condition to choose γ_k correctly is

$$u_k{}'B'C'CBu_k \pm 2u_k{}'B'C'(CAe_k + d_k) > 0\,,$$

and since both terms are scalars, this is equivalent to the first condition. The second condition results since if $\hat{\gamma}_k \neq \gamma_k$, then $J < 0$, which is only possible if the inequality holds. ■

Proposition 4.2 *If the estimation error has converged and there is no noise, i.e., $e_k = 0$ and $d_k = 0$, then the mode detector returns the correct value of γ_k. If $u_k = 0$, the output of the mode detector does not affect the state observer.*

Proof: With $e_k = d_k = 0$ we see that $|u_k{}'B'C'(CAe_k + d_k)| = 0$. Thus, for any $u_k \neq 0$ with the rank condition on CB, the first condition in Proposition 4.1 is satisfied, and $\hat{\gamma}_k = \gamma_k$. If $u_k = 0$, the estimation error is unaffected by the values of γ_k and $\hat{\gamma}_k$ since $(\gamma_k - \hat{\gamma}_k)(B - LCB)u_k = 0$. ■

Recalling the assumption of single input systems, where $u_k \in \mathbb{R}$ is a scalar, this also means that $B'C'CB \in \mathbb{R}$ is a scalar quantity, and we have $u_k{}'B'C'CBu_k = u_k^2 B'C'CB$. Let

$$\Lambda = \frac{1}{B'C'CB} B'C' \,, \tag{4.16}$$

which allows us to restate Proposition 4.1 as

$$\bullet \quad |u_k| > 2 \ |\Lambda(CAe_k + d_k)| \quad \Rightarrow \quad \hat{\gamma}_k = \gamma_k \tag{4.17}$$

$$\bullet \quad \hat{\gamma}_k \neq \gamma_k \quad \Rightarrow \quad |u_k| < 2 \ |\Lambda(CAe_k + d_k)| \tag{4.18}$$

From Eqn. (4.17) we have a sufficient condition on (u_k, e_k, d_k) to assure $\hat{\gamma}_k = \gamma_k$. If we knew e_k and d_k, we could simply pick u_k to satisfy the condition, but of course e_k and d_k are unknown. Instead assume a bound is known on the initial conditions and that we consider only norm-bounded noises:

$$\begin{aligned} \|x_0\| &\leq \delta_x \\ \|e_0\| &\leq \delta_e \\ \|d_k\| &\leq \delta_d \\ \|z_k\| &\leq \delta_z \,. \end{aligned}$$

In the next section it is shown that by augmenting a state feedback signal to ensure the control value is large enough, the fate of γ_k can be detected while still assuring stability of the closed loop system.

4.4.1 Augmenting the Control Signal to Guarantee Detection

In this section we show that by using a modified state feedback with an augmented control signal we can not only guarantee detection of γ_k, i.e., $\hat{\gamma}_k = \gamma_k$, but also that the closed loop is stable in some sense. Before giving the expression for the control signal that guarantees detection, we derive some properties of the estimation error assuming the fate of the control packet is known.

The estimation error from the state observer, given in Eqn. (4.6), can be evaluated at time k according to

$$\begin{aligned} e_k &= (A - LCA)^k e_0 + \sum_{j=0}^{k-1} (A - LCA)^{k-j-1} h_j \\ h_j &= (\gamma_j - \hat{\gamma}_j)(B - LCB)u_j + z_j \ . \end{aligned}$$

If we assume that we are always able to pick a u_k satisfying $|u_k| > 2\ |\Lambda(CAe_k + d_k)|$, then $\hat{\gamma}_k = \gamma_k$ and $h_j = z_j$. Hence, the norm of the estimation error can be bounded in this case according to

$$\|e_k\| \leq \|(A - LCA)^k\|\delta_e + \sum_{j=0}^{k-1} \|(A - LCA)^{k-j-1}\|\delta_z \triangleq \eta_k \ . \tag{4.19}$$

As shown in the lemma below, this has a finite upper bound.

Lemma 4.3 *If we are able to always recover $\hat{\gamma}_k = \gamma_k$, the norm of the estimation error, which is given in Eqn. (4.19), is bounded as $k \to \infty$.*

Proof: Since $A - LCA$ is stable, we have $\lim_{k\to\infty} \|(A - LCA)^k\| = 0$, so the first term in Eqn. (4.19) goes to zero as $k \to \infty$. Let the second term be represented by $M_k\delta_z$, where

$$M_k = \sum_{j=0}^{k-1} \|(A - LCA)^{k-j-1}\| \ , \tag{4.20}$$

with $M_0 = 0$ and $M_1 = I$. Since the term δ_z is independent of k and finite, it remains to show that the sum M_k converges and be upperbounded by a finite value, $\lim_{k\to\infty} M_k = M < \infty$. Since $A - LCA$ is stable, there exists a finite integer $s > 0$, such that $K = \|(A - LCA)^s\| < 1$. Set q to be the smallest s, then for all $i \in [tq, (t+1)q - 1)$, $t \geq 1$, we get

$$\begin{aligned} \|(A - LCA)^i\| &= \|(A - LCA)^{q\times(i/q)}\| \\ &\leq \|(A - LCA)^q\|^{i/q} \\ &= K^{i/q} \\ &\leq K^t \end{aligned}$$

Therefore, we have

$$\begin{aligned} M &= \sum_{j=0}^{q-1} \|(A-LCA)^j\| + \sum_{j=q}^{\infty} \|(A-LCA)^j\| \\ &= \sum_{j=0}^{q-1} \|(A-LCA)^j\| + \sum_{j=q}^{2q-1} \|(A-LCA)^j\| + \sum_{j=2q}^{3q-1} \|(A-LCA)^j\| + \cdots \\ &\leq \sum_{j=0}^{q-1} \|(A-LCA)^j\| + QT + qK^2 + qK^3 + \cdots \\ &= \sum_{j=0}^{q-1} \|(A-LCA)^j\| + \frac{QT}{1-K} \end{aligned}$$

Hence, the estimation error is bounded as $k \to \infty$

$$\lim_{k\to\infty} \|e_k\| \leq \delta_z \left(\sum_{j=0}^{q-1} \|(A-LCA)^j\| + \frac{qK}{1-K} \right) . \tag{4.21}$$

■

As mentioned above, Eqn. (4.17) gives a condition to guarantee detection of γ_k that depends on two unknown terms: the estimation error e_k and noise term d_k. The proposition below gives another condition to guarantee detection that only depends on known quantities.

Proposition 4.4 *If we pick at each time step a control value that satisfies*

$$|u_k| > 2\|\Lambda\| \left(\|CA\|\eta_k + \delta_d \right) \triangleq \Delta_k , \tag{4.22}$$

then we are guaranteed to have $\hat{\gamma}_k = \gamma_k$. In addition, the right hand side of Eqn. (4.22) remains upperbounded by a finite value as $k \to \infty$.

Proof: We prove the first statement by induction. Pick any u_0 satisfying Eqn. (4.22). Then with η_k from Eqn. (4.19) we can write

$$\begin{aligned} |u_0| &> 2\|\Lambda\| \left(\|CA\| \left(\|(A-LCA)^0\|\delta_e + M_0\delta_z \right) + \delta_d \right) \\ &\geq 2\|\Lambda\| \left(\|CA\| \, \|e_0\| + \delta_d \right) \\ &\geq 2\|\Lambda\| \left(\|CAe_0\| + \delta_d \right) \\ &\geq 2\|\Lambda\| \, |CAe_0 + d_0| , \end{aligned}$$

which from Eqn. (4.17) implies $\hat{\gamma}_0 = \gamma_0$. Then, from Eqn. (4.19) $\|e_1\| \leq \|A - LCA\|\delta_e + M_1\delta_z$. Now repeat the steps above. Pick any u_1 satisfying Eqn. (4.22) and we get

$$\begin{aligned} |u_1| &> 2\|\Lambda\|\left(\|CA\|\,\|e_1\| + \delta_d\right) \\ &\geq 2\|\Lambda\|\left(\|CAe_1\| + \delta_d\right) \\ &\geq 2\|\Lambda\|\,|CAe_1 + d_1|\ , \end{aligned}$$

implying $\hat{\gamma}_1 = \gamma_1$ and $\|e_2\| \leq \|(A - LCA)^2\|\delta_e + M_2\delta_d$. Then by induction it can be shown that choosing u_k to satisfy Eqn. (4.22) gives $\hat{\gamma}_k = \gamma_k$ for all k, and the norm of the error stays bounded above by Eqn. (4.19).

The second statement is a direct result of Lemma 4.3. The right hand side of Eqn (4.22) is upper bounded as $k \to \infty$ by

$$2\|\Lambda\|\left[\|CA\|\left(\sum_{j=0}^{q-1}\|(A - LCA)^j\| + \frac{qK}{1-K}\right)\delta_z + \delta_d\right]$$

■

We now have a state observer and mode detector, together with a constraint on the control action to ensure the fate of the k^{th} control packet, γ_k, can be recovered at time $k+1$. This assures the estimation error is bounded, as shown in Lemma 4.3. The algorithm is summarized in Table 4.2.

Table 4.2: Estimator Algorithm.

1. Constrain the control to satisfy
$$|u_k| > 2\|\Lambda\|\left(\|CA\|\,\eta_k + \delta_d\right)$$
2. Run the mode detector to select $\hat{\gamma}_k$
$$\hat{\gamma}_k = \arg\min_{\beta\in\{0,1\}} \|y_{k+1} - CA\hat{x}_k - \beta CBu_k\|^2$$
3. Run the state observer
$$\hat{x}_{k+1} = A\hat{x}_k + \hat{\gamma}_k Bu_k + L\left(y_{k+1} - CA\hat{x}_k - \hat{\gamma}_k CBu_k\right)$$

So far the estimation error has been shown to be upperbounded using the scheme presented. Of course another major point of concern is the closed loop performance of the system, i.e., the state error. Enforcing the input constraint could lead to stable estimation but poor closed loop performance. Below it is shown that the closed loop is stable even with the input constraint by using a modified state feedback control law. First the noise-free case is shown to achieve almost sure stability, and then with the noise present the state is shown to be stable in the expected sense.

If we assume there is no noise acting on the system, we have $w_k = v_k = d_k = z_k = \delta_d = \delta_z = 0$. In this case the condition of Proposition 4.4 reduces to

$$|u_k| > 2 \|\Lambda\| \|CA\| \left\|(A - LCA)^k\right\| \delta_e \triangleq \widetilde{\Delta}_k \ . \tag{4.23}$$

As we now show, using a modified version of a state feedback controller we can make the closed loop system almost surely stable. The analysis models the system as a jump linear system of the form of Eqn. (4.10). As shown in [18, 17], the sufficient conditions for almost sure stability of a jump linear system with dynamics $\{\mathbf{A}_{\theta(k)}\}$ is that the modes $\{\theta(k)\}$ be a finite-state ergodic Markov process with unique invariant distribution $\mathbf{Pr}[\theta(k) = j] = \pi_j$ and that there exists some norm such that $\prod_{i=1}^{N} \|\mathbf{A}_i\|^{\pi_i} < 1$.

Theorem 4.5 *With no noise present ($w_k = v_k = 0$), using the state observer and mode detector described above along with the state feedback controller*

$$u_k = F\hat{x}_k + sgn(F\hat{x}_k) 2 \|\Lambda\| \|CA\| \left\|(A - LCA)^k\right\| \delta_e \ , \tag{4.24}$$

where F is designed such that $A + BF$ is stable, then the closed loop system is almost surely stable if there exists some H-norm such that

$$\|\mathbf{A}_1\|_H^{\overline{\gamma}} \ \|\mathbf{A}_2\|_H^{1-\overline{\gamma}} < 1 \ , \tag{4.25}$$

where

$$\mathbf{A}_1 = \left[\begin{array}{c|c} A + BF & -BF \\ \hline & \\ 0 & A - LCA \end{array}\right] \tag{4.26}$$

$$\mathbf{A}_2 = \left[\begin{array}{c|c} A & 0 \\ \hline & \\ 0 & A - LCA \end{array}\right] \ . \tag{4.27}$$

Proof: With the definition of $\widetilde{\Delta}_k$ in Eqn (4.23), the modified state feedback controller in Eqn. (4.24) is equivalent to $u_k = F\hat{x}_k + \mathrm{sgn}(F\hat{x}_k)\widetilde{\Delta}_k$, and it is easy to see that

$$|u_k| \quad = \quad |F\hat{x}_k| + \widetilde{\Delta}_k \geq \widetilde{\Delta}_k \ ,$$

with equality only holding when $|F\hat{x}_k| = 0$. If $|F\hat{x}_k| > 0$, then Eqn. (4.23) is satisfied and $\hat{\gamma}_k = \gamma_k$. If $|F\hat{x}_k| = 0$, then $u_k = 0$, and the control does not affect the estimation as described in Proposition 4.2.

We can write the closed loop system as

$$\mathbf{X}_{k+1} = \mathbf{A}_{\theta(k)}\mathbf{X}_k + \Omega_k \tag{4.28}$$

$$\theta(k) = \begin{cases} 1\,, & \text{if } \gamma_k = 1 \\ 2\,, & \text{if } \gamma_k = 0 \end{cases} \tag{4.29}$$

$$\Omega_k = \gamma_k \text{sgn}(F\hat{x}_k) \begin{bmatrix} B \\ 0 \end{bmatrix} \widetilde{\Delta}_k\,. \tag{4.30}$$

From Eqn. (4.26) - (4.27) we see the $\mathbf{A}_1$ matrix is stable and the $\mathbf{A}_2$ matrix is unstable. Following the definitions of the H-norm, $\|\mathbf{A}_1\|_H < 1$ and $\|\mathbf{A}_1\|_H < \|\mathbf{A}_2\|_H$.

Define $D_i(k) = \prod_{j=i}^{k-1} \mathbf{A}_{\theta(j)}$ for $i \leq k-1$ and $D_k(k) = I$, where $\theta(j) \in \{1, 2\}$ and the multiplication is on the left. For example, $D_0(3) = \mathbf{A}_{\theta(2)}\mathbf{A}_{\theta(1)}\mathbf{A}_{\theta(0)}$. Similarly, define $E_i(k) = \prod_{j=i}^{k-1} \|\mathbf{A}_{\theta(j)}\|_H$. We have the following easily verifiable relationships:

1. $D_i(k) = D_j(k)D_i(j)$ and $E_i(k) = E_j(k)E_i(j)$ for any $i \leq j \leq k$
2. $\|D_i(k)\|_H \leq E_i(k)$
3. $E_0(k) \geq E_i(k)\|\mathbf{A}_1\|_H^i$ for any $k - 1 \geq i \geq 0$
4. $\mathbf{Pr}\left[\lim_{k\to\infty} E_0(k) = \left(\|\mathbf{A}_1\|_H^{\overline{\gamma}} \;\; \|\mathbf{A}_2\|_H^{1-\overline{\gamma}}\right)^k\right] = 1$ from the law of large numbers, and from Eqn. (4.25) we see that $\mathbf{Pr}\left[\lim_{k\to\infty} E_0(k) = 0\right] = 1$; in fact, $E_0(k) \to 0$ as $k \to \infty$ exponentially fast in k.

Also let $\mathbf{B} = \left\| \begin{bmatrix} B \\ 0 \end{bmatrix} \right\|_H$. Using this notation allows us to write

$$\begin{aligned}
\|\mathbf{X}_k\|_H &= \left\| D_0(k)\mathbf{X}_0 + \sum_{i=0}^{k-1} D_{i+1}(k)\Omega_i \right\|_H \\
&\leq \|D_0(k)\|_H \|\mathbf{X}_0\|_H + \sum_{i=0}^{k-1} \|D_{i+1}(k)\|_H \|\Omega_i\|_H \\
&\leq E_0(k)\|\mathbf{X}_0\|_H + \sum_{i=0}^{k-1} E_{i+1}(k)\mathbf{B}\widetilde{\Delta}_i \\
&= E_0(k)\|\mathbf{X}_0\|_H + \sum_{i=0}^{k-1} E_{i+1}(k)\|\mathbf{A}_1\|_H^{i+1} \frac{\mathbf{B}\widetilde{\Delta}_i}{\|\mathbf{A}_1\|_H^{i+1}} \\
&\leq E_0(k)\|\mathbf{X}_0\|_H + \mathbf{B}E_0(k) \sum_{i=0}^{k-1} \frac{\widetilde{\Delta}_i}{\|\mathbf{A}_1\|_H^{i+1}} \\
&= E_0(k) \left(\|\mathbf{X}_0\|_H + 2\mathbf{B} \|\Lambda\| \, \|CA\| \delta_e \sum_{i=0}^{k-1} K_i \right) ,
\end{aligned}$$

where

$$K_i = \frac{\|(A - LCA)^i\|}{\|\mathbf{A}_1\|_H^{i+1}} .$$

The next step is to prove that K_i has a finite upper bound for all i. To simplify notation let $A_{BF} = A + BF$ and $A_{LC} = A - LCA$. Denote $\sigma_{BF} = \rho(A_{BF})$ and $\sigma_{LC} = \rho(A_{LC})$, i.e., the spectral radius of A_{BF} and A_{LC}, respectively, and $\overline{\sigma} = \max(\sigma_{BF}, \sigma_{LC})$. Note that since $\mathbf{A}_1$ is block diagonal with A_{BF} and A_{LC} on the diagonals, we have $\rho(\mathbf{A}_1) = \overline{\sigma}$. From page 299 of [26] we know that for any matrix norm $\|\cdot\|$ on a matrix $T \in \mathbb{R}^{n \times n}$ we have $\|T^i\| \geq \rho(T)^i$. Then we can write

$$\|\mathbf{A}_1\|_H^{i+1} \geq \left\|\mathbf{A}_1^{i+1}\right\|_H \geq \rho(\mathbf{A}_1)^{i+1} = \overline{\sigma}^{i+1} .$$

Letting V_{LC} be a matrix which can diagonalize A_{LC}, i.e., $V_{LC}^{-1} A_{LC} V_{LC} = \Lambda_{LC}$ with Λ_{LC} being the diagonal matrix of the eigenvalues of A_{LC}, we can write

$$\begin{aligned}
\|(A - LCA)^i\| &= \|A_{LC}^i\| \\
&= \|V_{LC}\Lambda_2^i V_{LC}^{-1}\| \\
&\leq \|V_{LC}\| \, \|V_{LC}^{-1}\| \, \|\Lambda_{LC}^i\| \\
&= \|V_{LC}\| \, \|V_{LC}^{-1}\| \, \sigma_{LC}^i
\end{aligned}$$

Now let $C_1 = \frac{\|V_{LC}\| \, \|V_{LC}^{-1}\|}{\overline{\sigma}}$, and combining the above statements we get

$$K_i \leq C_1 \left(\frac{\sigma_{LC}}{\overline{\sigma}} \right)^i \leq C_1 \ .$$

In fact, if $\sigma_{BF} > \sigma_{LC}$, then $\overline{\sigma} = \sigma_{BF}$ and $K_i \to 0$ as $i \to \infty$. Since (A, B) is controllable and (A, C) is observable, we can make $\sigma_{BF} > \sigma_{LC}$, and this is equivalent to having the observer eigenvalues faster than the controller.

As we only need K_i to be upperbounded, we can simply use the bound that $K_i \leq C_1$ for all i. Now we return to the H-norm on $\mathbf{X}_k$ to get

$$\begin{aligned} \|\mathbf{X}_k\|_H &\leq E_0(k) \left(\|\mathbf{X}_0\|_H + 2\mathbf{B} \, \|\Lambda\| \, \|CA\| \delta_e \sum_{i=0}^{k-1} K_i \right) \\ &\leq E_0(k) \left(\|\mathbf{X}_0\|_H + 2\mathbf{B} \, \|\Lambda\| \, \|CA\| \delta_e k C_1 \right) \ . \end{aligned}$$

Taking the limit as $k \to \infty$ we see that from the right hand side we get

$$\mathbf{Pr} \left[\lim_{k \to \infty} E_0(k) \left(\|\mathbf{X}_0\|_H + k 2\mathbf{B} \, \|\Lambda\| \, \|CA\| \delta_e C_1 \right) = 0 \right] = 1 \ ,$$

thus $\mathbf{Pr} \left[\lim_{k \to \infty} \|\mathbf{X}_k\|_H = 0 \right] = 1$. ■

With almost sure stability proven when the noise is absent, the results are now extended to show stability in an expected sense for the case where the bounded system and measurement noise are present, i.e., $\delta_z, \delta_d > 0$.

Theorem 4.6 *Using the estimator described in Table 4.2 along with a modified state feedback controller*

$$u_k = F\hat{x}_k + sgn(F\hat{x}_k)\Delta_k \ , \tag{4.31}$$

then if there exists a positive definite matrix $H > 0$ such that

$$\psi \triangleq (1 - \bar{\gamma}) \cdot \|A\|_H + \bar{\gamma} \cdot \|A + BF\|_H < 1 \ , \tag{4.32}$$

then the following holds:

$$\mathbf{E} \left[\, \|x_k\| \, \right] \leq \left(\psi^k \delta_x + \frac{1 - \psi^k}{1 - \psi} \cdot \Sigma \right) \frac{\overline{\lambda}_H}{\underline{\lambda}_H} \ , \tag{4.33}$$

with

$$\Sigma \triangleq \delta_w + \max_{k \geq 0} \|B\| \Delta_k + \|BF\| \eta_k \ , \tag{4.34}$$

and $\overline{\lambda}_H$ and $\underline{\lambda}_H$ signifying the maximum and minimum eigenvalues of H, respectively.

Proof: Using the control signal in Eqn. (4.31), the state dynamics update equation can then be rewritten as

$$\begin{aligned} x_{k+1} &= (A+\gamma_k BF)x_k + q_k \\ q_k &= w_k + \gamma_k B(sgn(F\hat{x}_k)\Delta_k - Fe_k) \; . \end{aligned} \tag{4.35}$$

Using the control signal in Eqn. (4.31) with the estimator algorithm guarantees that $\hat{\gamma}_k = \gamma_k$, and thus the error is bounded according to Eqn. (4.19), and we see

$$\|q_k\| \leq \delta_w + \|B\|\Delta_k + \|BF\|\eta_k \leq \Sigma \; .$$

Now define $\mathcal{A}_j^k = \prod_{i=j}^{k-1}(A+\gamma_i BF)$, with $\mathcal{A}_j^k = I$ if $j \leq k$. This allows us to express the state as

$$x_k = \mathcal{A}_0^k x_0 + \sum_{j=0}^{k} \mathcal{A}_{j+1}^k q_j \; .$$

Taking the expectation of the H-norm of this expression above we get

$$\begin{aligned} \mathbf{E}\left[\|x_k\|_H\right] &= \mathbf{E}\left[\left\|\mathcal{A}_0^k x_0 + \sum_{j=0}^{k} \mathcal{A}_{j+1}^k q_j\right\|_H\right] \\ &\leq \mathbf{E}\left[\|\mathcal{A}_0^k x_0\|_H\right] + \sum_{j=0}^{k} \mathbf{E}\left[\|\mathcal{A}_{j+1}^k q_j\|_H\right] \quad . \end{aligned} \tag{4.36}$$

Examining the first term we can write

$$\begin{aligned} \mathbf{E}\left[\|\mathcal{A}_0^k x_0\|_H\right] &\leq \mathbf{E}\left[\|\mathcal{A}_0^k\|_H \cdot \|x_0\|_H\right] \\ &\leq \mathbf{E}\left[\|\mathcal{A}_0^k\|_H\right] \cdot \delta_x \cdot \overline{\lambda}_H \\ &= \mathbf{E}\left[\left\|\prod_{j=0}^{k}(A+\gamma_j BF)\right\|_H\right] \delta_x \cdot \overline{\lambda}_H \\ &\leq \mathbf{E}\left[\prod_{j=0}^{k}\|(A+\gamma_j BF)\|_H\right] \delta_x \cdot \overline{\lambda}_H \\ &\leq \prod_{j=0}^{k}\mathbf{E}\left[\|(A+\gamma_j BF)\|_H\right]\delta_x \cdot \overline{\lambda}_H \; , \end{aligned} \tag{4.37}$$

where the last line comes from the assumption that the packet drops are independent from one time-step to the next. Note that the following holds:

$$\mathbf{E}\left[\|A+\gamma_j BF\|_H\right] = (1-\bar{\gamma})\cdot\|A\|_H + \bar{\gamma}\cdot\|A+BF\|_H = \psi \ ,$$

thus we have

$$\mathbf{E}\left[\|\mathcal{A}_0^k x_0\|_H\right] \leq \psi^k \delta_x \cdot \overline{\lambda}_H \ . \tag{4.38}$$

Now returning to the second term in Eqn. (4.36)

$$\begin{aligned}
\sum_{j=0}^{k}\mathbf{E}\left[\left\|\mathcal{A}_{j+1}^k q_j\right\|_H\right] &\leq \sum_{j=0}^{k}\mathbf{E}\left[\left\|\mathcal{A}_{j+1}^k\right\|_H \cdot \|q_j\|_H\right] \\
&\leq \sum_{j=0}^{k}\prod_{i=j+1}^{k}\mathbf{E}\left[\|(A+\gamma_i BF)\|_H\right]\cdot\Sigma\cdot\overline{\lambda}_H \\
&\leq \sum_{j=0}^{k}\prod_{i=j+1}^{k}\psi\cdot\Sigma\cdot\overline{\lambda}_H \\
&\leq \sum_{j=0}^{k}\psi^{k-j}\cdot\Sigma\cdot\overline{\lambda}_H \\
&= \frac{1-\psi^k}{1-\psi}\cdot\Sigma\cdot\overline{\lambda}_H \ .
\end{aligned} \tag{4.39}$$

Then noting $\|x_k\| \leq \|x_k\|_H/\underline{\lambda}_H$ and combining this with Eqns. (4.38) and (4.39) in Eqn. (4.36) we arrive at the expression in Eqn. (4.33). ■

To evaluate the performance as $k \to \infty$, it is easy to see that since $\psi < 1$ we have

$$\lim_{k\to\infty}\mathbf{E}\left[\|x_k\|\right] = \frac{\Sigma}{1-\psi}\cdot\frac{\overline{\lambda}_H}{\underline{\lambda}_H} \ .$$

Remark 4.7 The stability shown in the theorem above differs from other current stability results [69] for jump linear systems with noise, as those require the noise signal to be an l_2 sequence, while clearly the noise in Eqn. (4.35) is not.

4.4.2 Removing the Added Input Signal

As shown in the previous section, by combining the estimator algorithm with the modified state feedback control signal in Eqn. (4.31), detection of γ_k is guaranteed, and the closed loop state is stable. The question arises of whether it is necessary to include the extra control effort. Without this enlarged control value it is no longer possible to guarantee that $\hat{\gamma}_k = \gamma_k$, but as we see below it might still be possible to stabilize the plant. The estimator algorithm without the added input is simply the mode detector and state observer, as shown in Table 4.3.

Table 4.3: Estimator algorithm without the input constraint.

1. Run the mode detector to select $\hat{\gamma}_k$
$$\hat{\gamma}_k = \arg \min_{\beta \in \{0,1\}} \|y_{k+1} - CA\hat{x}_k - \beta CBu_k\|^2$$
2. Run the state observer
$$\hat{x}_{k+1} = A\hat{x}_k + \hat{\gamma}_k Bu_k + L\left(y_{k+1} - CA\hat{x}_k - \hat{\gamma}_k CBu_k\right)$$

In the remainder of this section we show that this estimator algorithm without the added input can stabilize the closed loop using state feedback control. We first derive an expression for the estimation error dynamics in the presence of possible mis-detects. This allows a bound on the estimation error, even if the fate of γ_k is not always recovered, and, subsequently, the characterization of the closed loop performance.

Lemma 4.8 *Using the estimator algorithm consisting of the mode detector and state observer as in Table 4.3, the estimation error dynamics can be written as*

$$e_{k+1} = (\tilde{A}_k - L\tilde{C}_k)e_k + r_k \ , \tag{4.40}$$

with

$$\tilde{A}_k = A - \alpha_k 2B\Lambda CA \tag{4.41}$$

$$\tilde{C}_k = CA - \alpha_k 2CB\Lambda CA \tag{4.42}$$

$$r_k = \alpha_k (B - LCB)\Lambda d_k + z_k \tag{4.43}$$

for some $\alpha_k \in [0, 1]$.

Proof: Define

$$\Omega_k = (\gamma_k - \hat{\gamma}_k)(B - LCB)u_k \tag{4.44}$$

and consider the possible combinations of $(\gamma_k, \hat{\gamma}_k)$. As seen in Table 4.1, if $\gamma_k = \hat{\gamma}_k \Rightarrow \Omega_k = 0$, but if $\gamma_k \neq \hat{\gamma}_k$ the following must hold:

$$u_k{}'B'C'CBu_k < -2(\gamma_k - \hat{\gamma}_k)u_k{}'B'C'(CAe_k + d_k) \ . \tag{4.45}$$

Let us first consider the case where $(\gamma_k, \hat{\gamma}_k) = (0, 1)$. Since u_k is a scalar, Eqn. (4.45) reduces to $u_k^2 < u_k \cdot 2\Lambda \cdot (CAe_k + d_k)$, requiring u_k and $\Lambda \cdot (CAe_k + d_k)$ to have the same sign. Thus,

- if $u_k > 0$, then $0 < u_k < 2\Lambda \cdot (CAe_k + d_k)$ or
- if $u_k < 0$, then $2\Lambda \cdot (CAe_k + d_k) < u_k < 0$.

Combining these two statements we can write $u_k = \alpha_k 2\Lambda \cdot (CAe_k + d_k)$ for some $\alpha_k \in [0,1]$. Combine this with the fact that it corresponds to the case with $(\gamma_k, \hat{\gamma}_k) = (0,1)$ and insert into Eqn. (4.44) to get $\Omega_k = -\alpha_k(B - LCB)2\Lambda \cdot (CAe_k + d_k)$. Similarly it can be shown that for $(\gamma_k, \hat{\gamma}_k) = (1,0)$ we have $u_k = -\alpha_k 2\Lambda \cdot (CAe_k + d_k)$. Incorporating all possible combinations for $(\gamma_k, \hat{\gamma}_k)$ allows us to write Eqn. (4.44) as

$$\Omega_k = -\alpha_k(B - LCB)2\Lambda \cdot (CAe_k + d_k) \; . \tag{4.46}$$

From Eqn. (4.6) we can write the error update equation as

$$e_{k+1} = (A - LCA)e_k + \Omega_k + z_k \; ,$$

and inserting Eqn. (4.46) into this we arrive at Eqn. (4.40). ■

Using the estimation error dynamics in Eqn. (4.40), the lemma below gives conditions on the observation gain L such that estimation error is bounded. Note that the dynamics in Eqn. (4.40) can be thought of as an uncertain system through $(\tilde{A}_k, \tilde{C}_k)$, with a bounded noise term r_k. Also observe that $\alpha_k = 0$ when $\hat{\gamma}_k = \gamma_k$, so the uncertainty in $(\tilde{A}_k, \tilde{C}_k)$ only enters during mis-detects.

Lemma 4.9 *If there exists an observation gain L and a positive definite matrix $P > 0$ such that for all $\alpha_k \in [0,1]$*

$$\overline{\Delta V}_k \triangleq (\tilde{A}_k - L\tilde{C}_k)'P(\tilde{A}_k - L\tilde{C}_k) - P < 0 \; , \tag{4.47}$$

*then using **any** control signal the estimation error is upper bounded according to*

$$\|e_k\| \leq \sqrt{\frac{\lambda_2}{\lambda_1}} \left(1 + \lambda_6 \frac{\lambda_5 + \sqrt{\lambda_5^2 + \lambda_2\lambda_3}}{\lambda_3} \right) R \triangleq E \tag{4.48}$$

with

$$\begin{aligned}
\lambda_1 &= \underline{\lambda}(P) && (4.49)\\
\lambda_2 &= \overline{\lambda}(P) && (4.50)\\
\lambda_3 &= \min_{0 \leq \alpha_k \leq 1} \underline{\lambda}(-\overline{\Delta V}_k) && (4.51)\\
\lambda_4 &= \max_{0 \leq \alpha_k \leq 1} \overline{\lambda}(-\overline{\Delta V}_k) && (4.52)\\
\lambda_5 &= \max_{0 \leq \alpha_k \leq 1} \left\| (\tilde{A}_k - L\tilde{C}_k)'P \right\| && (4.53)\\
\lambda_6 &= \max_{0 \leq \alpha_k \leq 1} \left\| \tilde{A}_k - L\tilde{C}_k \right\| && (4.54)\\
R &= 2\,\|(B - LCB)\Lambda\|\,\delta_d + \delta_z && (4.55)
\end{aligned}$$

where $\overline{\lambda}(X)$ and $\underline{\lambda}(X)$ signify the largest and smallest eigenvalue of some matrix X, respectively.

Proof: Construct a nonnegative function $V_k = {e_k}' P e_k$. Then, using Eqn. (4.40) the change in this function from one time-step to the next is given by

$$\begin{aligned}
\Delta V_k &= V_{k+1} - V_k \\
&= {e_{k+1}}' P e_{k+1} - e_k P e_k \\
&= \left((\tilde{A}_k - L\tilde{C}_k)e_k + r_k\right)' P \left((\tilde{A}_k - L\tilde{C}_k)e_k + r_k\right) - {e_k}' P e_k \\
&= {e_k}' \left[\left(\tilde{A}_k - L\tilde{C}_k\right)' P \left(\tilde{A}_k - L\tilde{C}_k\right) - P\right] e_k + {r_k}' P r_k + 2{e_k}' \left(\tilde{A}_k - L\tilde{C}_k\right)' P r_k \quad (4.56)
\end{aligned}$$

The first term is simply ${e_k}' \overline{\Delta V}_k e_k$, which is negative per Eqn. (4.47); the term ${r_k}' P r_k$ is always positive; and the sign of the last term depends on the vectors e_k and r_k. To use V_k as a Lyapunov function we need it to decrease along system trajectories, i.e., $\Delta V_k < 0$; however, from Eqn. (4.56) we see that it is not always guaranteed to be negative. Analyzing the terms in Eqn. (4.56) further, we note

$$\begin{aligned}
{e_k}' \overline{\Delta V}_k e_k &\leq -\lambda_3 \|e_k\|^2 \quad (4.57) \\
{r_k}' P r_k &\leq \lambda_2 \|r_k\|^2 \leq \lambda_2 R^2 \; . \quad (4.58)
\end{aligned}$$

Using the Cauchy-Schwarz inequality we see

$$\begin{aligned}
\left| {e_k}' \left(\tilde{A}_k - L\tilde{C}_k\right)' P r_k \right| &\leq \left\| {e_k}' \left(\tilde{A}_k - L\tilde{C}_k\right)' P \right\| \cdot \|r_k\| \\
&\leq \|e_k\| \cdot \lambda_5 R \; . \quad (4.59)
\end{aligned}$$

Combining Eqn. (4.56) with Eqn.s (4.57), (4.58), and (4.59) we get

$$\Delta V_k \leq -\lambda_3 \|e_k\|^2 + 2\lambda_5 R \|e_k\| + \lambda_2 R^2 \; . \quad (4.60)$$

Note ΔV_k is negative quadratic in $\|e_k\|$, so solving for the positive root we get

$$e^* = \frac{R}{\lambda_3} \left(\lambda_5 + \sqrt{\lambda_5^2 + \lambda_2 \lambda_3}\right) \; . \quad (4.61)$$

As $\|e_k\| > e^*$ implies $\Delta V_k < 0$, it is then true that $V_{k+1} < V_k$ and $\|e_{k+1}\|^2 \leq \frac{V_{k+1}}{\lambda_1} \leq \frac{V_k}{\lambda_1} \leq \|e_k\|^2 \frac{\lambda_2}{\lambda_1}$. If $\|e_k\| \leq e^*$, it is possible $\Delta V_k \geq 0$. We do know, however, that

$$\|e_{k+1}\| \leq \lambda_6 \|e_k\| + R \; ,$$

therefore $\|e_k\| \leq e^*$ implies $\|e_{k+1}\| \leq \lambda_6 e^* + R$. If the actual value of e_{k+1} is such that $\|e_{k+1}\| > e^*$, then $\Delta V_{k+1} < 0$ and $\|e_{k+2}\| \leq \|e_{k+1}\|\sqrt{\frac{\lambda_2}{\lambda_1}}$. If the value is $\|e_{k+1}\| \leq e^*$, then $\|e_{k+2}\| \leq \lambda_6 e^* + R$. Then, assuming the initial error also satisfies $\|e_0\| \leq E$, we get the expression in Eqn. (4.48). ■

With a bound on the estimation error now derived for the case without the added input signal, we see that this estimation scheme can tolerate possible mis-detects of the fate of the control packets and now provide a bound for the closed loop error using state feedback, as shown in the theorem below.

Theorem 4.10 *If there exists an observation gain L and a positive definite matrix $P > 0$ satisfying the conditions of Lemma 4.9, then using the estimator algorithm in Table 4.3 with a simple state feedback controller without the added effort, i.e., $u_k = F\hat{x}_k$, the state is bounded in the expected sense:*

$$\mathbf{E}\left[\|x_k\|\right] \leq \left(\psi^k \delta_x + \frac{1-\psi^k}{1-\psi} \cdot \Theta\right) \frac{\overline{\lambda}(H)}{\underline{\lambda}(H)} \,, \tag{4.62}$$

where

$$\Theta = \|BF\|\sqrt{\frac{\lambda_2}{\lambda_1}}\left[1 + \frac{\lambda_6}{\lambda_3}\left(\lambda_5 + \sqrt{\lambda_5^2 + \lambda_2\lambda_3}\right)\right] R + \delta_w \,. \tag{4.63}$$

Proof: The proof is the same as Theorem 4.6 without the addition of the $B\Delta_k$ term to the bounded noise, and the estimation error bounded according to Lemma 4.9. ■

Remark 4.11 The stability conditions presented above are conservative. Performing the worst case analysis for the noise sequences and using the properties of the norm make the upper bounds conservatives. Furthermore, the conditions are only sufficient, not necessary, and it turns out the algorithm performs favorably even when the conditions are not satisfied. The value of the derivations above is to show that the algorithms can stabilize the system. As we show in the simulations below, the actual performance can be substantially better than these upper bounds.

4.4.3 Unknown Input Observers

The purpose of the estimator algorithm described above is to develop an estimation and control scheme that can tolerate not having information on the fate of the control packets, i.e., γ_k. We now present an alternative estimator that does the same, namely the well known unknown input observer [41]. It consists of using estimator of the form

$$\hat{x}_{k+1} = (MA - KC)\hat{x}_k + Ky_k + Gy_{k+1} \,, \tag{4.64}$$

with $M = I - GC$ and the gains (K, G) chosen so that

- $MB = 0$ and

- $MA - KC$ is stable

Let $G = B(CB)^+ + Y(I - (CB)(CB)^+)$ satisfy $MB = 0$ for any arbitrary matrix Y, where $(CB)^+$ is the left (pseudo) inverse of CB. Applying the unknown input observer to the UDP system, the estimation error dynamics are given by

$$e_{k+1} = (MA - KC)e_k + w_k + Kv_k + G(Cw_k + v_{k+1}) \; . \tag{4.65}$$

According to [41], the necessary and sufficient conditions for the unknown input observer to exist are that: $\text{rank}(CB) = \text{rank}(B) = r \leq n$, so that $(CB)^+$ exists, and (MA, C) be detectable. The lemma below now relates the unknown input observer to the estimator algorithm presented in the previous section.

Lemma 4.12 *The following are true:*

1. *The existence of a unknown input observer with MA stable implies the conditions for Lemma 4.9 are satisfied with $L = G$.*

2. *The existence of an L satisfying the conditions for Lemma 4.9 implies the existence of a unknown input observer.*

Proof:

1. The unknown input observer exists with MA stable, and let $L = G$ be the observer gain used in the estimator algorithm from Section 4.4.2. Then since $MB = B - LCB = 0$, this means $(\tilde{A}_k, \tilde{C}_k) = (A, C)$, and since $MA = A - LCA = \tilde{A}_k - L\tilde{C}_k\tilde{A}_k$ is stable, then clearly Eqn. (4.47) can be satisfied.

2. If Eqn. (4.47) is satisfied for all $\alpha_k \in [0, 1]$, then it is satisfied for $\alpha_k = \frac{1}{2}$, which implies the pair

$$(A - B\Lambda CA, CA - CB\Lambda CA)$$

must be detectable. If we let $K = 0$, then for the unknown input observer to exist MA must be stable. Since $\text{rank}(CB) = \text{rank}(B)$, we have $(CB)^+ = (B'C'CB)^{-1}B'C'$, but as we are considering single input systems, $(B'C'CB)^{-1}$ is a scalar. Thus we can write $MA = A - B\Lambda CA - Y(CA - CB\Lambda CA)$, but since the pair above is detectable, MA can be made stable through proper selection of Y. Hence, the unknown input observer exists.

■

Thus, we see that if the sufficient conditions for the estimator algorithm are satisfied, then the unknown input observer exists as well, so the question arises of which is better to use? An

upper bound for the unknown input observer estimation error can be computed and compared with the upper bound using the estimator algorithm, both with and without the added control input. When the necessary and sufficient conditions for the unknown input observer are not satisfied, then by definition constructing an observer that cancels out the unknown input is unstable, whereas the conditions presented above for upper bound on the estimator algorithm are only sufficient conditions. Thus if the unknown input observer does not exist, although there is no proof that the estimator scheme without the added input is upperbounded, there is also no proof saying it is unstable and it is a better alternative. The example below helps to illustrate the usage of the different schemes.

4.5 Simulation Example

To illustrate the effectiveness of the estimator algorithm consider the following example:

$$A = \begin{bmatrix} 1.5 & 0.1 \\ 0.3 & 1.3 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; C = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

This system has unstable eigenvalues of $(1.2, 1.6)$. The pairs (A, B) and (A, CA) are controllable and observable, respectively, and $CB = 1$, so the estimation algorithm can be used to choose $\hat{\gamma}_k$. For this system the unknown input observer does not exist since the pair (MA, C) is not detectable, so this method is left out of all the simulations. The initial state and error are assumed to be bounded by $\|x_0\| \leq \sqrt{2}$ and $\|e_0\| \leq \sqrt{2}$.

First we assume that there is no noise acting on the system, and the feedback and observer gains are chosen to place the eigenvalues of $A + BF$ and $A - LCA$ at $(0.07, 0.08)$ and $(0.05, 0.06)$, respectively. This translates into setting the gains to $F = [-20.6, -2.65]$ and $L = [4.6401, 0.9984]'$. We pick $\overline{\gamma} = 0.75$ and simulate the system from random initial conditions and with random realizations of the packet drop sequence. Using the new scheme with the added input signal, a typical response profile is shown in Fig. 4.2. The top plots show the evolution of the state vector and the estimation error. The middle plots show the sequence of γ_k for this simulation and of $(\gamma_k - \hat{\gamma}_k)^2$. As can be seen, our estimation algorithm recovers $\hat{\gamma}_k = \gamma_k$ at every time-step, which allows the estimation error to converge quickly. The control history is plotted in the bottom plots, showing that the corrective term decays to zero, and u_k approaches $F\hat{x}_k$.

In Fig. 4.3 the norm of the state and estimation error are plotted for the algorithm with the added input signal and the other naive schemes described in Section 4.2. As expected, the naive schemes that use $\hat{\gamma}_k = 0$ and $\hat{\gamma}_k = \overline{\gamma}$ can be seen to have diverging state and estimation error (they are the two lines that go off the plot after time-step 2). The naive scheme using $\hat{\gamma}_k = 1$ performs slightly better. As expected, the norms of the state and estimation error decrease during the burst of successful packet reception, but during a burst of drops the norms grow very large. In fact, they

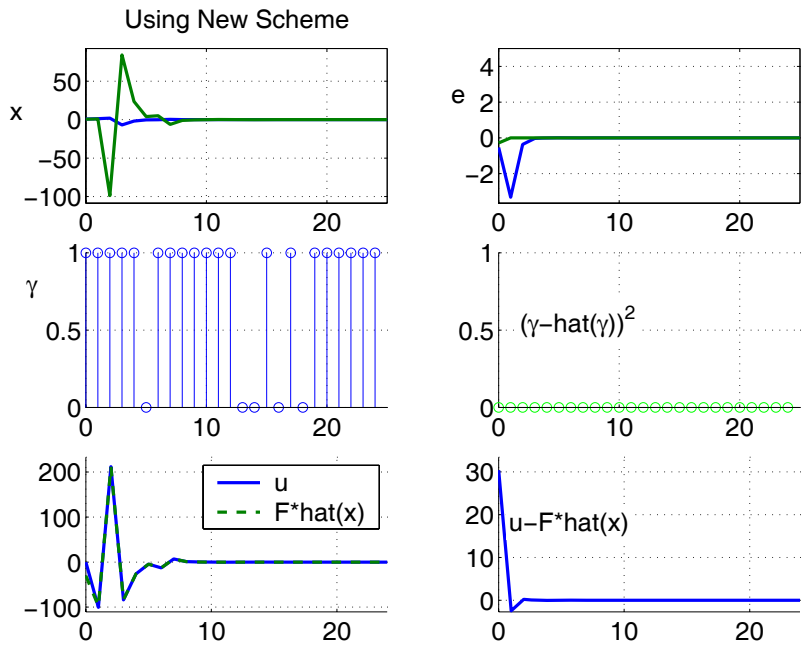

Figure 4.2: Simulation results utilizing new estimation scheme with added control.

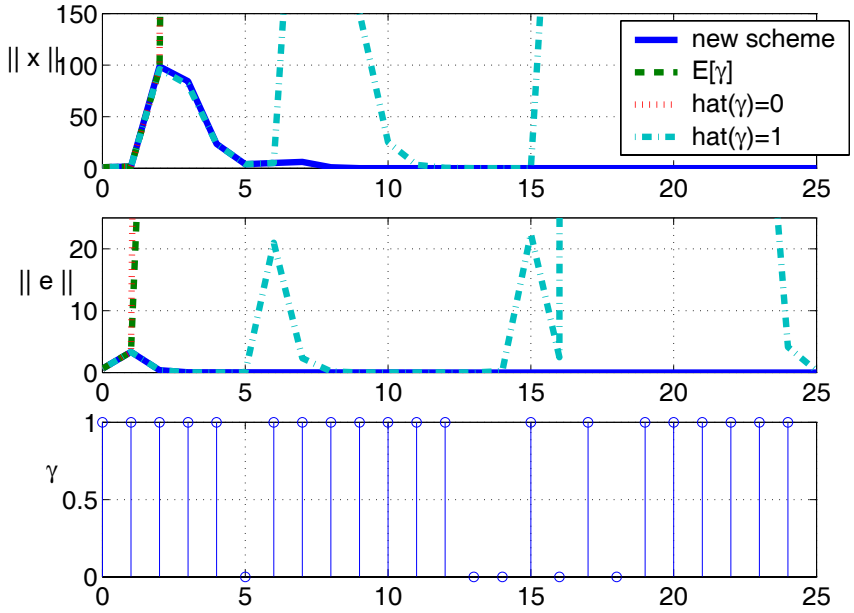

Figure 4.3: Comparison of state history for various control-estimation schemes.

grow to as large as $\|e_{19}\| = 7.62 \times 10^5$ and $\|x_{21}\| = 1.95 \times 10^7$. So although the naive scheme of $\hat{\gamma}_k = 1$ may converge almost surely as $k \to \infty$ for $\overline{\gamma}$ close to unity, it can display this terrible performance during a burst of packet drops.

Now consider the same dynamics and bounds on the initial state and estimation error, but include nonzero noise to be present with $\delta_w = 1$ and $\delta_v = 0.1$. The estimator algorithm is run with observer

gain $L = [3.9, 0.98]'$ and a state feedback controller with gain $F = [-12.95, -2.05]$. These correspond to the eigenvalues of $A + BF$ and $A - LCA$ placed at $(0.35, 0.4)$ and $(0.178 + 0.0820\sqrt{-1}, 0.178 - 0.0820\sqrt{-1})$, respectively. We simulated the UDP estimation algorithm described in this chapter, both with and without the extra control input, as well as both the naive case of selecting $\hat{\gamma}_k = 1$, and the TCP case where the estimator has direct knowledge of γ_k. A total of 10,000 simulations are run each for 50 time steps. Random initial conditions and noise sequences are chosen but were the same for all the different estimation schemes. We use an average packet acceptance rate of $\bar{\gamma} = 0.85$.

The averages of the state and estimation error norms across all 10,0000 simulations are shown in Figure. 4.4. The scheme using $\hat{\gamma}_k = 1$ quickly diverges, while as expected the TCP case has the best results. Both UDP estimation schemes, with and without the added input value, show average estimation errors that overlap with the TCP case. In fact, when including the additional input value, the estimates are identical to the TCP case since we recover $\hat{\gamma}_k = \gamma_k$. The price to pay is that the state norm is larger as a result of including the extra input term. Using the UDP estimation scheme without the additional input we see that the performance is virtually identical to that of the TCP case for both the estimation error and the state norm. We do not always recover the fate of γ_k using this scheme, but in fact the selection of $\hat{\gamma}_k$ is correct nearly 99% of the time.

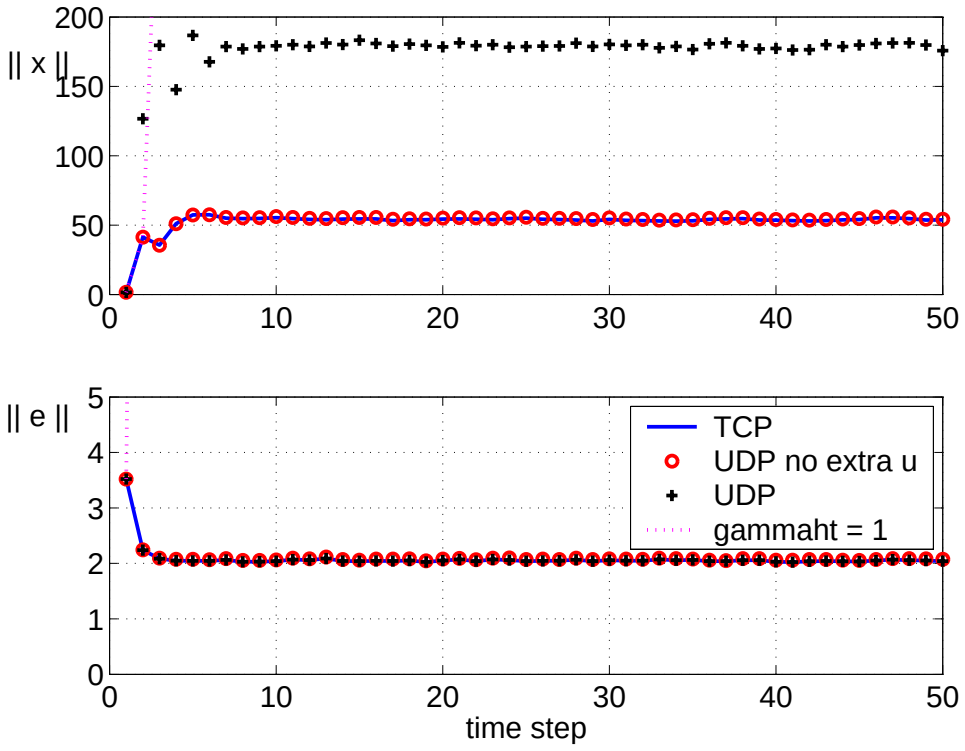

Figure 4.4: Average of $\|x_k\|$ and $\|e_k\|$ across all 10,0000 simulations..

Figure 4.5 shows the results from a particular simulation. Note how the UDP estimation scheme without the added control tracks the TCP case except for a few time-steps starting at 19, when

it results in $\hat{\gamma}_k \neq \gamma_k$. This mistake causes the estimation error to increase slightly, which likewise induces a larger state norm. After a few successive time steps with no error in $\hat{\gamma}_k$, the estimation error and state norm quickly collapse onto the TCP case. For all the estimation schemes, the periods of time where the control packets are dropped, i.e., $\gamma_k = 0$, correspond to the state norm increasing. The bottom plot shows the control efforts. Note how the UDP estimation scheme applies a larger control effort because of the larger state norm and the added input used to detect γ_k.

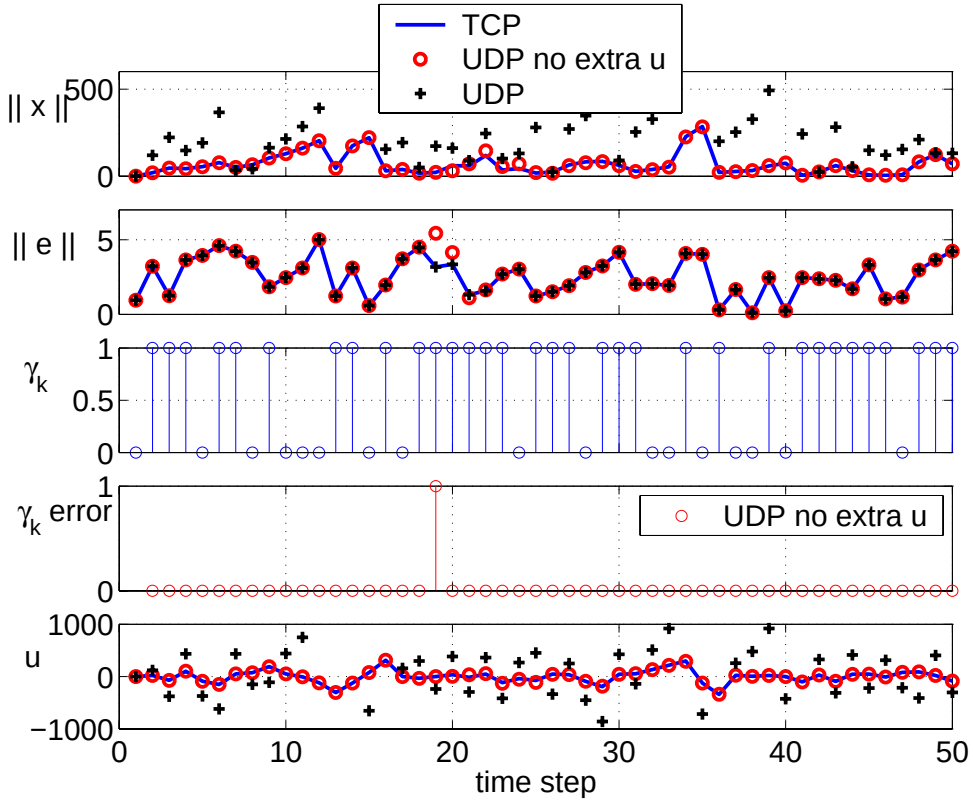

Figure 4.5: Plots from a single simulation for various estimation schemes.

4.6 Conclusions and Future Work

We presented an estimation algorithm for UDP-like networked control systems. First, to ensure detection of the fate of the control packet, an added control input is included. An upper bound for the expected value of the state norm in the presence of bounded state and measurement noise was presented. If the added control input is removed, the estimator algorithm is no longer guaranteed to detect the fate of the control packet. Nonetheless, under certain conditions on the system parameters, it can still be shown to produce an upper bound to the estimation error, which in turn allows an upper bound to the expected value of the state norm to be derived. The estimator algorithm is then compared to the unknown input observer, which can directly remove the dependence of the

control signal on the estimation error. A simulation example shows how the estimator algorithm works well, especially compared to the other methods (even if the sufficient conditions for stability are not satisfied), and provides guidelines for designing the estimator in this case.

Since the conditions for stability of the estimator algorithm overlap with those of the unknown input observer, it would be good if necessary stability conditions for the estimator algorithm could be derived. The algorithm appears to work well even if the sufficient conditions are not satisfied. Thus, relaxing this conservatism would make the result stronger. Modifying the model to instead include different noise types or uncertainties, have non i.i.d. packet drops, and add intelligence at the plant to apply some predicted control rather than evolve open loop when the control packet is dropped are all areas that can be investigated, though the results presented here should only need to be modified slightly to include these scenarios. The most interesting extension might be to insert a network between the sensors and estimator so that the estimator does not always have access to the sensor data, which would most likely require some additional logic in the algorithm.

Chapter 5

Using Hierarchical Decomposition to Speed Up Consensus

5.1 Introduction

Recent years have seen a large amount of research focused on issues relating to multi-agent and cooperative control [57]. The task is generally to have a group of systems/agents collectively achieve a desired task in a decentralized fashion while making use of shared information. Examples of specific applications include those in the areas of: consensus [58], behavior of swarms [87], multi-vehicle formation control [19], sensor fusion [78], and many others. These settings, like NCS, require information be exchanged between dynamical elements. As a result, managing the information flow has a direct impact on system performance. In this chapter we focus on the problem of consensus, i.e., having a group of agents reach agreement/consensus on a quantity of interest, and design a scheme to manage the flow of information in the consensus loop to speed up the time to convergence.

The average consensus problem, which is considered in this chapter, is to find a distributed algorithm such that a collection of agents reaches consensus on the average of their initial conditions. To do so, the agents must communicate their values to other agents, but they can only communicate with some subset of the other nodes. Given certain conditions on the topology of this communication network, and using an update rule that changes their value in the direction of the aggregate value of the nodes they communicate with, the average consensus can be achieved.

In addition to proving consensus can be reached, performance of the consensus algorithm is also a key area of research. One performance measure is in terms of the robustness to possible communication sharing impediments. Issues such as communication delays and changes in the communication topology over time have been examined, see [58] and [62]. Another performance measure that has been studied lately, and that is the focus of the present work, is the time to reach consensus.

Tools from graph theory [22] have been used to represent the information sharing topology and aid in the analysis in consensus problems. The key factor in determining the time to reach convergence has been shown to be the second smallest eigenvalue, represented as λ_2, of the graph Laplacian (the matrix representing the evolution of the agent's values based on the communication topology). In [90], the authors attempt to speed up the time to convergence, while trading-off robustness, by optimally choosing how much relative weight each node should give to the other nodal values in the update rule in order to maximize the ratio $\lambda_2/\lambda_{\max}$. A version for the discrete time case is given in [88]. Introducing additional communication links into the network and creating small world networks, as was done in [56], is another approach. The main idea of these approaches and others is the attempt to increase λ_2.

This chapter introduces a new approach to speed up convergence in consensus algorithms, applicable to graphs that can be decomposed into a hierarchical graph. It consists of splitting the overall graph into layers of smaller connected subgraphs. Consensus is performed within the individual subgraphs, starting with those of the lowest layer of the hierarchy and moving upwards. Certain "leader" nodes bridge the layers of the hierarchy. By exploiting the larger λ_2 values of the smaller subgraphs, this scheme can achieve faster overall convergence than the standard single-stage consensus algorithm running on the full graph topology. Furthermore, using consensus for the individual subgraphs endows them with the benefits associated with standard consensus algorithm, such as robustness to information perturbations, no need for a global planner within the subgraphs, etc. The contribution of this chapter is to extend the basic understanding of consensus algorithms to situations when the system may have a hierarchical structure. This hierarchical structure is typical in layered communication networks, where some nodes are gateways between clusters of local nodes and the rest of the network.

The chapter is organized as follows. Section 5.2 provides a quick review of graph theory and continuous-time consensus algorithms and introduces the concept of hierarchical decomposition of graphs. The hierarchical consensus algorithm is described and analyzed in Section 5.3. In Section 5.4 we describe methods to help select which hierarchical decomposition to chose. Section 5.5 contains examples and simulations to illustrate the algorithm and how to choose a decomposition. Conclusions and a description of future work are given in Section 5.6. The proofs for the analysis in Section 5.3 are presented in the appendix in Section 5.7. This work is joint with Kevin Lynch, Karl Johannsson, and Richard M. Murray, a version of which was submitted in [12].

5.2 Graphs and Continuous-Time Consensus

5.2.1 Graph Theory

Consider a group of N agents with an underlying graph topology $\mathcal{G} = \{V, E\}$, where $V = \{1, \ldots, N\}$ is the set of nodes and the edge set $E = \{(i, j)\} \subseteq V \times V$ is the node pairs (i, j), where node j sends information to node i. If the communication link between nodes i and j is bidirectional, then both (i, j) and $(j, i) \in E$. Define the neighbor set of node i as $\mathcal{N}_i = \{j \mid (i, j) \in E\}$. The adjacency matrix is the matrix $A = [a_{ij}]$, where $a_{ij} = 1$ if $(i, j) \in E$ otherwise $a_{ij} = 0$, and it is usually assumed that $a_{ii} = 0$. Letting $D = \text{diag}(d_i)$ with $d_i = \sum_j a_{ij}$, we then have the Laplacian matrix for graph $\mathcal{G}$ given by

$$L = D - A\,. \tag{5.1}$$

The Laplacian matrix has been widely studied in the context of graph theory [22], and it plays a central role in the analysis of consensus algorithms [57].

5.2.2 Hierarchical Graph Decomposition

The algorithm presented in this chapter is applicable to graphs that can be hierarchically decomposed. In this section we provide a formal definition for a hierarchical decomposition and present some associated properties.

Definition 5.1 *An M-layer hierarchical decomposition of a connected graph $\mathcal{G} = \{V, E\}$ consists of a collection of subgraphs $\mathcal{G}_j^i = \{V_j^i, E_j^i\}$, with $i = 1, \ldots M$, of $\mathcal{G}$. The vertex set of $\mathcal{G}_j^i$ is denoted by $V_j^i \subseteq V$, and $E_j^i = \{(m, n) \in E \mid m, n \in V_j^i\} \subseteq E$ denotes the edge set. Let S^i denote the number of subgraphs, $\mathcal{G}_1^i, \mathcal{G}_2^i, \ldots, \mathcal{G}_{S^i}^i$, at layer i. Let $\mathcal{V}^i = \bigcup_{j=1}^{S^i} V_j^i$ be the set of all nodes in layer i. The collection of subgraphs $\mathcal{G}_j^i$ must satisfy the following properties:*

1. *Each $\mathcal{G}_j^i$ is connected, and $\left|V_j^i\right| \geq 1$.*
2. *There is only one top-layer graph, i.e., $S^M = 1$.*
3. *The lowest-layer graphs contain all the nodes of the graph, i.e., $\mathcal{V}^1 = V$.*
4. *The subgraphs at the same layer i are disjoint, i.e., $V_j^i \bigcap V_k^i = \emptyset$ for all $j \neq k$ and $j, k \in \{1, \ldots, S^i\}$.*
5. *For each $\mathcal{G}_j^i, i < M$, there exists exactly one parent subgraph $\mathcal{G}_m^{i+1}$ that shares a single node, i.e., exactly one $m \in \{1, \ldots, S^{i+1}\}$ satisfies $|V_j^i \bigcap V_m^{i+1}| = 1$.*

An example of a hierarchical decomposition is given in Fig. 5.1. Notice that the hierarchical decomposition does not require that all available links be utilized since the links between nodes 2 and 3 as well as between nodes 5 and 6 are never used. Of course each graph does not necessarily have a unique hierarchical decomposition, and others could be created that utilize these links.

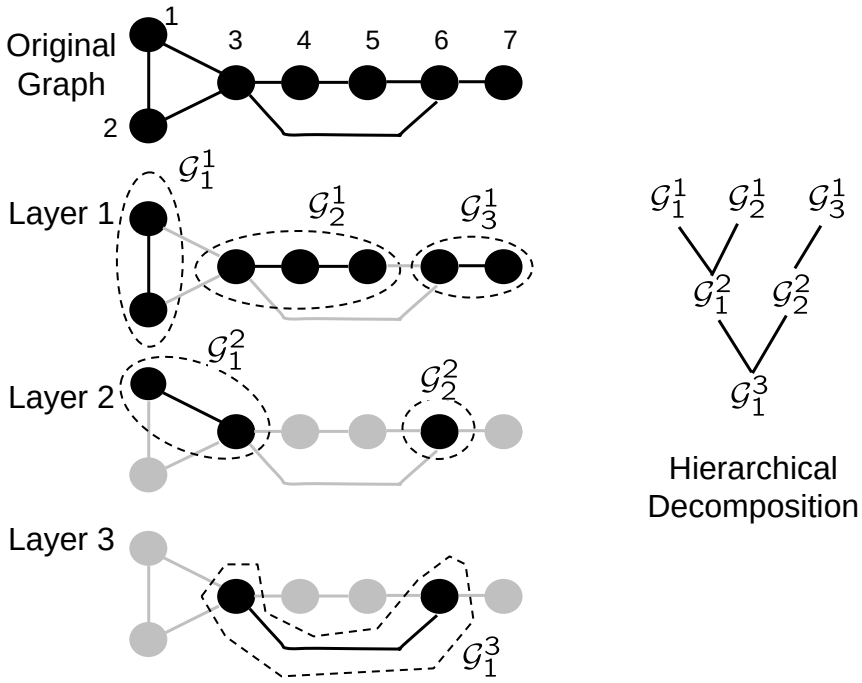

Figure 5.1: Example hierarchical decomposition.

We introduce a few more definitions and properties associated with the subgraphs of a hierarchical decomposition. All nodes present in a given layer $i < M$ have an associated *parent node* for that layer. If node k is in subgraph $\mathcal{G}_j^i$, then node k's parent node is the one node in $\mathcal{G}_j^i$ that is also in some subgraph of the next layer $\mathcal{G}_m^{i+1}$. Let $p(k,i)$ denote the parent node of node k for layer i, i.e., if $k \in V_j^i$ and its parent node is in V_m^{i+1}, then $p(k,i) = V_j^i \bigcap V_m^{i+1}$. A node can be its own parent node. As an example, referring to the hierarchical decomposition in Fig. 5.1, we see that in layer 1, node 1 is the parent node for both itself and node 2, i.e., $p(1,1) = p(2,1) = 1$.

For every layer i, each node k also has an associated *leader node*, denoted by $\mathcal{L}_k^i$, whose definition is slightly different than its parent node. If a node k is in layer i, $k \in \mathcal{V}^i$, then it is its own leader node $\mathcal{L}_k^i = k$. If the node is in layer $i-m$, but it is not in any of layers from $i-m+1, i-m+2, \ldots, i$, i.e., $k \in \mathcal{V}^{i-m}$ but $k \notin \bigcup_{l=0}^{m-1} \mathcal{V}^{i-l}$, then its leader node is the successive parent node from layer $i-m$ to $i-1$, i.e., $\mathcal{L}_k^i = p(p(\cdots p(p(k,i-m),i-m-1),\cdots),i-1)$. Later, the leader node is shown to be important, as each node "follows" the value of its leader node. In Fig. 5.1 we see that node 2's leader nodes in layers $1, 2$, and 3 are nodes $2, 1$, and 3, respectively, i.e., $\mathcal{L}_2^1 = 2$, $\mathcal{L}_2^2 = 1$, and $\mathcal{L}_2^3 = 3$.

The *total node set* for the subgraph $\mathcal{G}_j^i$, denoted $\mathbb{V}_j^i$, is the set of all nodes whose leader node is in subgraph $\mathcal{G}_j^i$, i.e., $\mathbb{V}_j^i = \left\{k \mid \mathcal{L}_k^i \in V_j^i\right\}$. For the first layer we see $\mathbb{V}_j^1 = V_j^1$ for all $j = 1, \ldots, S^1$. For each layer i, associate to each node k a *total neighbor set* that is given by $\mathcal{N}_k^i = \mathbb{V}_j^i$ for $\mathcal{L}_k^i \in V_j^i$, where this set represents node k and all nodes that are connected to it across layers 1 through i. We let $\mathcal{N}_k^0 = k$, and note that if node k is in layer i, the nodes that "follow" it are those in the set $\mathcal{N}_k^{i-1} \backslash k$. Since all the subgraphs of any layer i are disjoint, then for any nodes k and j if there is any overlap in their total neighbor sets, $\mathcal{N}_k^i \bigcap \mathcal{N}_j^i \neq \emptyset$, the sets must be identical, $\mathcal{N}_k^i = \mathcal{N}_j^i$, and their leader nodes must be in the same subgraph of layer i. From this we can see that every total neighbor set $\mathcal{N}_k^i$ is repeated $\left|\mathcal{N}_k^i\right|$ times, and there a total of S^i unique neighbor sets for layer i. The union of the total neighbor sets at any layer equals V. We also have the following relationship:

$$\mathcal{N}_k^{i+1} = \bigcup_{m \in V_j^{i+1}} \mathcal{N}_m^i \ \text{, for } \ \mathcal{L}_k^{i+1} \in V_j^{i+1} \ . \tag{5.2}$$

Note in the final layer $\mathbb{V}_1^M = \mathcal{N}_k^M = V$ for all k, meaning both the total node set and every total neighbor set in the final layer is the set containing all nodes of the graph. Now we refer to Fig. 5.1 to see the total node sets for layer 2 are $\mathbb{V}_1^2 = \{1, 2, 3, 4, 5\}$ and $\mathbb{V}_2^2 = \{6, 7\}$. Focusing on node 2, the total neighbor sets are $\mathcal{N}_2^1 = \{1, 2\}$, $\mathcal{N}_2^2 = \{1, 2, 3, 4, 5\}$, and $\mathcal{N}_2^3 = \{1, \cdots, 7\}$.

5.2.3 Continuous-Time Consensus

Denote the state of node i as x_i. The standard consensus algorithm consists of each agent's dynamics evolving according to

$$\dot{x}_i = \sum_{j=1}^{N} a_{ij} \left(x_j - x_i\right) \ , \quad i = 1, \ldots, N \ , \tag{5.3}$$

with a_{ij} defined by the adjacency matrix as given above. If we let the vector $x = [x_1, \ldots, x_N]^T$, then we can write the consensus dynamics of the group as

$$\dot{x} = -Lx \ , \tag{5.4}$$

where L is the graph Laplacian as defined above. Consensus is reached when all the nodal values are the same, $x_1 = x_2 = \cdots = x_N$. The goal is average consensus, where the consensus value is the average of the initial conditions, $\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i(0)$. Let $\overline{\mathbf{x}} = \mathbf{1}_N^T \overline{x}$ be the vector version of the scalar quantity $\overline{x}$, with $\mathbf{1}_N$ the row ones vector with dimension N. The consensus error is then defined as

$$e(t) = x(t) - \overline{\mathbf{x}} \ . \tag{5.5}$$

The standard consensus algorithm, and certain variants of it, have been widely studied over the years. Much focus has been given to requirements on the topology of the network, and hence the Laplacian, for consensus to be achieved. Likewise, the performance of the consensus algorithm, i.e., the time to reach consensus, is governed by the Laplacian. More specifically, following [57], with a connected graph and L symmetric (all links are bidirectional), all the eigenvalues lie on the real axis in the right half plane. The error convergence is then bounded by the second smallest eigenvalue λ_2 of L according to

$$\|e(t)\|_2 \leq e^{-\lambda_2 t}\|e(0)\|_2 \ , \tag{5.6}$$

so the consensus error goes to 0 as $t \to \infty$. Clearly, larger values of λ_2 cause the upper bound of the error to converge faster.

For use with the hierarchical scheme, we introduce the term *starting value* to differentiate from initial conditions. The initial conditions $x(0)$ are the values of the nodes at the very beginning of the consensus algorithm, i.e., at the lowest layer of the hierarchy. The starting values are the node values at the start of a new layer. Denote the start time of layer i as t_{i-1}^+, and the end time as t_i^-, and for layer 1 we assume $t_0^+ = 0$. The starting values for layer i are thus $x(t_{i-1}^+)$. We then define the *subgraph average* to be the average of the starting values of the subgraph nodes

$$\overline{S}_j^i = \text{average}\left(\{x_k(t_{i-1}^+) \mid k \in V_j^i\}\right) \ , \tag{5.7}$$

and the nodal subgraph error is

$$\tilde{e}_k(t) = \begin{cases} x_k(t) - \overline{S}_j^i & \text{for } k \in V_j^i \\ 0 & \text{if } k \notin \mathcal{V}^i \end{cases} \tag{5.8}$$

for $t \in [t_{i-1}^+, t_i^-]$. The consensus error is still relative to the average of the initial conditions as in Eqn. (5.5).

In the next section we develop a variant of the standard consensus algorithm, applicable to graphs that can be hierarchically decomposed, to speed up the convergence time. It takes advantage of larger λ_2 values for the smaller subgraphs compared to the λ_2 of the full graph. For example, the full graph in Fig. 5.1 has $\lambda_2 = 0.586$, while the smallest λ_2 of all the subgraphs of the hierarchical decomposition is 1.

5.3 Hierarchical Consensus Algorithm Description and Analysis

In this section we introduce the hierarchical consensus algorithm and then provide an analysis deriving a bound on the consensus error under this scheme and end with a discussion of the key features of the algorithm.

5.3.1 Hierarchical Consensus Algorithm

As opposed to the standard single-stage consensus algorithm that utilizes the full communication topology, the new consensus scheme aims to speed up the convergence by exploiting the hierarchical decomposition of the graph. The scheme consists of "disconnecting" the hierarchical graph into layers consisting of smaller disconnected subgraphs. Consensus is run within the subgraphs while moving up the hierarchy. The scheme does not introduce additional links to the topology, but may not utilize all available links depending on the decomposition.

The scheme works by running consensus dynamics, like Eqn. (5.3), within each subgraph, starting with the subgraphs in the lowest layer. The subgraphs start converging towards the average of their starting values at that layer. The layer is complete when the nodes within that layer have all converged to within a specified tolerance ϵ_s of their respective subgraph average, that is $\|\tilde{e}\|_\infty \leq \epsilon_s$. As we show later, this stopping condition can be assured by enforcing a minimum layer time. After this stopping criterion is met, the algorithm moves to the next higher layer of the hierarchy, repeating until the final layer is reached. The flow diagram for this scheme is shown in Fig. 5.2.

Subgraphs of consecutive layers are connected by the nodes that are present in both layers. These nodes allow the information to flow between the levels in the hierarchy. As the algorithm moves to higher levels, these nodes must disseminate information to their follower nodes. We assume the leaders are able to relay their values down to their followers, still utilizing the communication topology but in a different mode than before. This allows all nodes to instantaneously assume the value of their leader node, i.e., for every layer i we get $x_k(t) = x_{\mathcal{L}_k^i}(t)$.

In designing the hierarchical scheme, we want the consensus value that every subgraph converges towards to be equal to the average of the initial conditions of that subgraph's total node set. This means we want $\overline{S}_j^i$ as defined in Eqn. (5.7) to be equal to average $\big(\{x_k(0) \mid k \in \mathbb{V}_j^i\}\big)$. To assure this we simply rescale the starting values of the nodes within the subgraph. Combining this rescaling with the relay option that sets all follower node values to their leader's value, we have upon starting

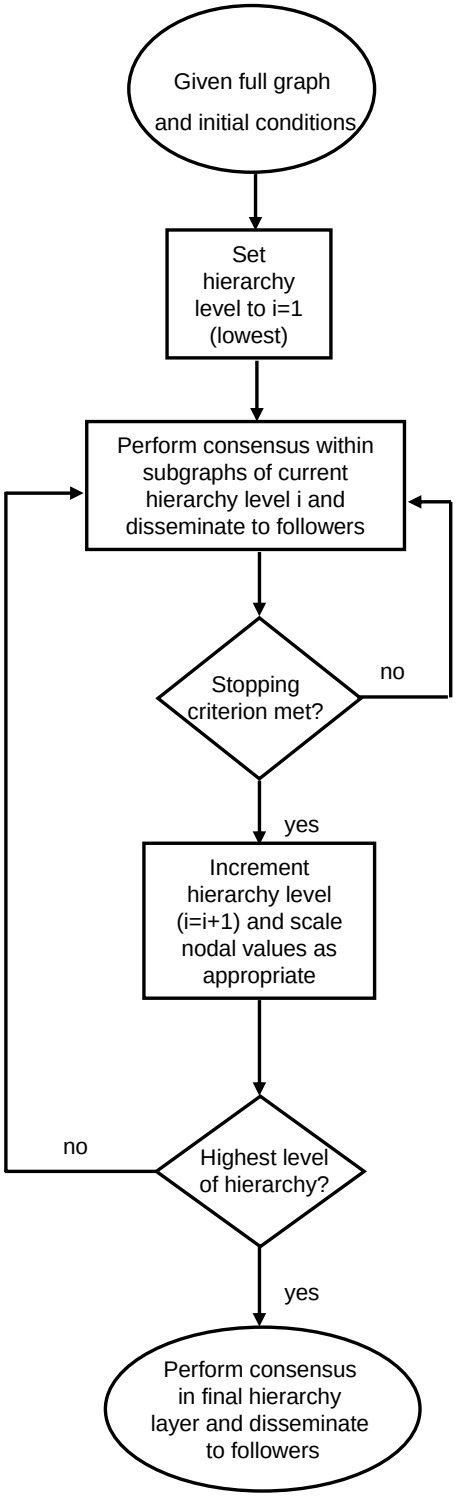

Figure 5.2: Flow diagram for the hierarchical consensus scheme.

a new layer i

$$\begin{aligned} x_k(t_{i-1}^+) &= \alpha^i_{\mathcal{L}_k^i} x_{\mathcal{L}_k^i}(t_{i-1}^-) \\ \alpha^i_{\mathcal{L}_k^i} &= \left|\mathbb{V}_j^i\right|^{-1} \left|V_j^i\right| \left|\mathcal{N}^{i-1}_{\mathcal{L}_k^i}\right| \text{ , for } \mathcal{L}_k^i \in V_j^i \text{ ,} \end{aligned} \tag{5.9}$$

and we let $\alpha_k^1 = 1$ for all k. It is important to note that in order for the nodes to compute α_k^i they only need to know the local topology of the hierarchical graph; that is the number of nodes in their subgraph and the total number of followers for each of those nodes. The scaling factors are independent of the actual values of the nodes. If all nodes in a subgraph have the same number of total followers, the corresponding scale value is 1, i.e., if $\left|\mathcal{N}^{i-1}_{\mathcal{L}_k^i}\right| = \left|\mathcal{N}^{i-1}_{\mathcal{L}_l^i}\right|$ for all $k, l \in V_j^i$, then it is easy to see $\left|\mathbb{V}_j^i\right| = |V_j^i||\mathcal{N}^{i-1}_{\mathcal{L}_k^i}|$ and $\alpha_k^i = 1$. Self-similar graphs, such as that in Fig. 5.3, have this property.

5.3.2 Analysis

We want to bound the consensus error for the hierarchical scheme and compare this with the bound of the standard single-stage consensus algorithm utilizing the full graph topology that is given in Eqn. (5.6). As mentioned above, a layer is considered completed as soon as $\|\tilde{e}\|_\infty \leq \epsilon_s$. We first derive a bound for the consensus error of the hierarchical scheme, which depends on the value of ϵ_s, and then we show how to guarantee this tolerance can be met.

We need to assume a bound on the initial conditions,

$$\|x(0)\|_\infty \leq \beta \text{ ,}$$

to aid in the analysis of the rescaling. Next, define for each layer i the vectors $\overline{\mathcal{N}}^i = [\overline{\mathcal{N}}^i_1, \overline{\mathcal{N}}^i_2, \ldots, \overline{\mathcal{N}}^i_N]^T$ and $\widetilde{\mathcal{N}}^i = [\widetilde{\mathcal{N}}^i_1, \widetilde{\mathcal{N}}^i_2, \ldots, \widetilde{\mathcal{N}}^i_N]^T$ with

$$\overline{\mathcal{N}}^i_k = \text{average}\left(\{x_m(0) \mid m \in \mathcal{N}_k^i\}\right) \tag{5.10}$$

$$\widetilde{\mathcal{N}}^i_k = \text{average}\left(\{x_m(t_{i-1}^+) \mid m \in V_j^i\}\right) \text{ , for } \mathcal{L}_k^i \in V_j^i \text{ .} \tag{5.11}$$

Thus, $\overline{\mathcal{N}}^i_k$ represents the average of the initial conditions of all nodes connected to node k from layer 1 through layer i, and $\widetilde{\mathcal{N}}^i_k$ represents the average of the starting values at layer i of all nodes in the subgraph containing node k's leader node.

To bound the consensus error we start by noting that for each layer i we can write

$$\begin{aligned} \|e(t)\|_2 &= \|x(t) - \overline{\mathbf{x}}\|_2 \\ &= \left\| x(t) + \overline{\mathcal{N}}^i - \overline{\mathcal{N}}^i + \widetilde{\mathcal{N}}^i - \widetilde{\mathcal{N}}^i - \overline{\mathbf{x}} \right\|_2 \\ &\leq \left\| x(t) - \widetilde{\mathcal{N}}^i \right\|_2 + \left\| \widetilde{\mathcal{N}}^i - \overline{\mathcal{N}}^i \right\|_2 + \left\| \overline{\mathcal{N}}^i - \overline{\mathbf{x}} \right\|_2 \end{aligned} \tag{5.12}$$

for any $t \in [t_{i-1}^+, t_i^-]$. The first term on the right hand side of Eqn. (5.12) represents the error of the nodes with respect to their subgraph averages. The second term is the difference between the total neighbor subgraph averages and the total neighbor initial condition averages. The final term represents the difference between the average of the initial conditions of the total neighbor sets for layer i and the initial conditions of all the nodes. We now provide bounds for each of the terms on the right hand side of Eqn. (5.12).

Lemma 5.2 *Using the hierarchy scheme with stopping tolerance ϵ_s, then for any layer i the difference between the total neighbor subgraph averages and the total neighbor initial condition averages can be bounded by*

$$\left\| \widetilde{\mathcal{N}}^i - \overline{\mathcal{N}}^i \right\|_2 \leq (i-1)\epsilon_s \sqrt{N} \; . \tag{5.13}$$

Proof: See Appendix in Section 5.7. ■

Lemma 5.3 *For any layer $i < M$, the difference between the average of the initial conditions of the total neighbor sets and the initial conditions of all the nodes is bounded by*

$$\left\| \overline{\mathcal{N}}^i - \overline{\mathbf{x}} \right\|_2 \leq \|e(0)\|_2 \; . \tag{5.14}$$

For the final layer we have by definition

$$\left\| \overline{\mathcal{N}}^M - \overline{\mathbf{x}} \right\|_2 = 0 \; . \tag{5.15}$$

Proof: See Appendix in Section 5.7. ■

With these lemmas we have bound the last two terms on the right hand side of Eqn. (5.12). Now we seek to bound the first term. To do so we use the following results which bound the consensus error at the end and the beginning of each layer.

Lemma 5.4 *Given the stopping criterion $\|\tilde{e}\|_\infty \leq \epsilon_s$, at the end of every hierarchy level $i < M$ the consensus error is bounded according to*

$$\|e(t_i^-)\|_2 \leq \|e(0)\|_2 + i\epsilon_s\sqrt{N}\,. \tag{5.16}$$

As the final layer M does not terminate, i.e., $t_M^- \to \infty$, we get that the steady state consensus error for the hierarchy scheme is bounded by

$$\lim_{t\to\infty} \|e(t)\|_2 \leq (M-1)\epsilon_s\sqrt{N}\,. \tag{5.17}$$

Proof: See Appendix in Section 5.7. ■

Lemma 5.5 *The consensus error at the beginning of every layer i can be bound by*

$$\|e(t_{i-1}^+)\|_2 \leq \|e(0)\|_2 + \epsilon_s\sqrt{N}\left(2\left\|\tilde{\alpha}^i\right\|_\infty + i - 1\right) + \left\|\tilde{\alpha}^i - 1\right\|_\infty \sqrt{N}\beta\,, \tag{5.18}$$

where $\tilde{\alpha}^i = \left[\alpha_k^i\right]_{k\in\mathcal{V}^i}$.

Proof: See Appendix in Section 5.7. ■

Using the bound on the consensus error at the start of each layer, we now provide a bound on the subgraph error at the start of each layer.

Lemma 5.6 *At the start of any layer i, the subgraph error $\tilde{e}$ as defined in Eqn. (5.8) is bounded by*

$$\left\|\tilde{e}(t_{i-1}^+)\right\|_2 \leq \widetilde{e}_0^i\,,$$

where

$$\tilde{e}_0^i = b^i\,\|e(0)\|_2 + \left\|\tilde{\alpha}^i - 1\right\|_\infty \beta\sqrt{N} + 2\epsilon_s\sqrt{N}\left(\left\|\tilde{\alpha}^i\right\|_\infty + i - 1\right)\,, \tag{5.19}$$

with $b^i = 2$ if $i < M$ and $b^M = 1$.

Proof: See Appendix in Section 5.7. ■

We are now ready to determine a bound for the first term on the right hand side of Eqn. (5.12), as given below.

Lemma 5.7 *Using the hierarchy scheme, the subgraph error is bounded by*

$$\left\|x(t) - \widetilde{\mathcal{N}}^i\right\|_2 \leq \widetilde{e}_0^i\left[\sum_{j=1}^{S^i}\left(\max_{k\in V_j^i}\left|\mathcal{N}_k^{i-1}\right|\right)e^{-2\lambda_2^{i,j}\left(t-t_{i-1}^+\right)}\right]^{\frac{1}{2}} \tag{5.20}$$

for $t \in [t_{i-1}^+, t_i^-]$, with $\widetilde{e}_0^i$ as given in Eqn. (5.19) and $\lambda_2^{i,j}$ representing the second smallest eigenvalue of the subgraph $\mathcal{G}_j^i$.

Proof: See Appendix in Section 5.7. ■

Now that we have provided bounds for all the terms on the right hand side of Eqn. (5.12), we are in a position to bound the consensus error.

Theorem 5.8 *Using the hierarchical consensus scheme with layer stopping tolerance $\left\|\tilde{e}(t_i^-)\right\|_\infty \leq \epsilon_s$, the consensus error during each layer i can be bounded according to*

$$\|e(t)\|_2 \leq \widetilde{e}_0^i \left[\sum_{j=1}^{S^i} \left(\max_{k \in V_j^i} \left|\mathcal{N}_k^{i-1}\right|\right) e^{-2\lambda_2^{i,j}\left(t - t_{i-1}^+\right)}\right]^{\frac{1}{2}} + (i-1)\epsilon_s\sqrt{N} + \left(b^i - 1\right)\|e(0)\|_2 \ , \qquad (5.21)$$

with $\tilde{e}_0^i$ as in Eqn. (5.19), $b^i = 2$ if $i < M$ and $b^M = 1$, and $\lambda_2^{i,j}$ representing the second smallest eigenvalue of the subgraph $\mathcal{G}_j^i$

Proof: This is a direct consequence of Eqn. (5.12) with Lemmas 5.2, 5.3, and 5.7. ■

As mentioned in the description of the hierarchical algorithm, each layer $i < M$ terminates when $\|\tilde{e}\|_\infty \leq \epsilon_s$. Up to this point we simply assumed this stopping criterion is met before moving to the next layer. With the analysis above and assuming we know a bound on the initial error, we can assure the criterion is met by keeping the time within each layer to be larger than a certain value, as show below.

Lemma 5.9 *To assure the stopping criterion $\left\|\tilde{e}(t_i^-)\right\|_\infty \leq \epsilon_s$ is met for each layer $i < M$ of the hierarchical scheme, we simply make sure the layer time (time spent in the layer), which is given by*

$$T_i = t_i^- - t_{i-1}^+ \ , \qquad (5.22)$$

is large enough to satisfy the following inequality:

$$\tilde{e}_0^i \sum_{j=1}^{S^i} e^{-\lambda_2^{i,j} T_i} \leq \epsilon_s \ , \qquad (5.23)$$

for every layer i.

Proof: From Lemma 5.6 we know the subgraph error is bounded by $\tilde{e}_0^i$ at the start of layer i. Using the fact that within any layer the subgraph error from each subgraph must be no greater than the total subgraph error from all those in the layer, we can bound the starting subgraph error for each subgraph by $\tilde{e}_0^i$ as well. Since each subgraph is connected, the error of subgraph $\mathcal{G}_j^i$ converges at a

rate no slower than $\lambda_2^{i,j}$. Hence the individual subgraph error for graph $\mathcal{G}_j^i$ is

$$\tilde{e}_0^i e^{-\lambda_2^{i,j}\left(t-t_{i-1}^+\right)} .$$

The norm of the total subgraph error for any layer is bound by the sum of the norm of the individual subgraph errors in that layer. Hence, we arrive at

$$\|\tilde{e}(t)\|_2 \leq \tilde{e}_0^i \sum_{j=1}^{S^i} e^{-\lambda_2^{i,j}\left(t-t_{i-1}^+\right)} \tag{5.24}$$

for $t \in [t_{i-1}^+, t_i^-]$. From this we see that if Eqn. (5.23) is satisfied, then $\|\tilde{e}\|_2 \leq \epsilon_s \Rightarrow \left\|\tilde{e}(t_i^-)\right\|_\infty \leq \epsilon_s$. ■

With the definition of layer times in Eqn. (5.22), and since $t_0^+ = 0$, we see that

$$t_i = \sum_{j=1}^{i} T_j \ . \tag{5.25}$$

With the bound for $\|e(t)\|_2$ determined, we can compute when this bound is lower than that of the standard single-stage consensus. In the hierarchical scheme, only after the final layer begins are all the nodes be hierarchically connected. Therefore we look at the consensus error of the hierarchical scheme during the final layer and compare with the standard algorithm.

Theorem 5.10 *The bound on the norm of the consensus error is smaller for the hierarchical scheme than the standard single-stage consensus algorithm for all time $t_{M-1}^+ + T$ with $0 \leq T_* < T < T^* < \infty$ and satisfying the following inequality*

$$e^{-\lambda_2 T} e^{-\lambda_2 t_{M-1}^+} \|e(0)\|_2 - e^{-\lambda_2^M T} \tilde{e}_0^M \left(\max_{k \in \mathcal{V}^M} \left| \mathcal{N}_k^{M-1} \right| \right) \geq (M-1)\epsilon_s \sqrt{N} \ , \tag{5.26}$$

with λ_2^M the second smallest eigenvalue of the Laplacian of $\mathcal{G}_1^M$. That is to say the bound on the hierarchical scheme is smaller than the bound of the single-stage consensus during the finite time interval $t \in (t_{M-1}^+ + T_*, t_{M-1}^+ + T^*)$.

Proof: This is a straightforward comparison of the bound on the consensus error in the hierarchical scheme using Eqn. (5.21), with the final layer $i = M$ and the bound on the standard consensus algorithm from Eqn. (5.6). ■

5.3.3 Discussion

We have derived a bound on the hierarchy consensus error and compared it to the standard single-stage consensus algorithm culminating with Theorem 5.10. The key factors that determine when

the condition is met are: the stopping tolerance, ϵ_s; the layer times, T_i; the speed of convergence of the full graph λ_2 and subgraphs $\lambda_2^{i,j}$; the initial error, $\|e(0)\|_2$ and how tight the known bound on the initial error is.

Notice that the term on the right hand side of the inequality in Eqn. (5.26) is the bound on the steady state consensus error of the hierarchy scheme, which depends on the number of nodes, number of layers, and stopping criterion tolerance ϵ_s. In fact ϵ_s plays more of a role, as described below. The first term on the left hand side of Eqn. (5.26) is the consensus error bound for the standard scheme. Notice that at the time of the start of the final layer of the hierarchy, $t = t_{M-1}^+$ and $T = 0$, the second term is greater than $\|e(0)\|_2$, and since $t_{M-1}^+ > 0$, the condition of Eqn. (5.26) can not be satisfied at $T = 0$. As T increases, since we assume $\lambda_2^M > \lambda_2$, the second term goes to zero faster than the first. Thus there is a time $T = T_*$ such that the inequality is satisfied, and the larger the difference between the eigenvalues the faster the inequality is satisfied.

The inequality can also be satisfied faster the sooner the final layer starts, i.e., smaller values of t_{M-1}^+. Of course for the final layer to start, all previous layers must have satisfied the stopping criterion. Thus we see that the layer times T_i should be chosen as small as possible to satisfy Eqn. (5.23). Key in this is how conservative of a bound we can assume on $\|e(0)\|_2$. Since most likely the starting norm error is not known, we instead use a bound for the value in Eqn. (5.23), as well as in evaluating the inequality in Eqn. (5.26). If the bound is conservative, then the layer times are longer than necessary, and the inequality is conservative.

The stopping tolerance ϵ_s also determines the layer times, so at first one might decide not to make this value too small. Since the steady state error of the hierarchy scheme is proportional to ϵ_s and we want to have a small steady state error, there is a trade-off. Note that as T continues to increase, the inequality in Eqn. (5.26) once again fails to be satisfied after a certain point $T = T^* > T_*$. This occurs when the standard consensus error bound crosses the (nearly) steady state value of the hierarchical scheme consensus error bound. Finally, it should be noted that using the analysis above one could chose a desired level of consensus error to be reached and could chose the appropriate parameters so that the hierarchical scheme reach this bound first.

5.4 Choosing A Hierarchical Decomposition

Section 5.3 describes the hierarchical consensus algorithm and provides analysis of the algorithm's performance given a hierarchical decomposition. In Section 5.5, an example is given with an assumed decomposition. As noted in Section 5.2, given a communication topology, there is not necessarily a unique hierarchical decomposition. In this section we explore ways to choose which hierarchical decomposition is appropriate to use. Then in Section 5.5 we illustrate these choices with examples.

We seek to choose a decomposition that gives good performance. Therefore we can use the analysis in Section 5.3.2 to compare the performance of different hierarchical decompositions. If one could write down all the possible hierarchical decompositions for a given graph, the performance for each decomposition could be compared. While this is feasible for graphs with a small number of total possible decompositions, it can be impractical for graphs with a large number of possible decompositions. There are certain features common to different graph decompositions that we explore in this section. The goal is to be able to use the case studies presented in this section as design guides for selecting a hierarchical decomposition. The decomposition features that we explore in this section are: (i) comparing using more layers with subgraphs that are more highly connected within each layer to using fewer layers with subgraphs not as highly connected in each layer, and (ii) determining to which subgraph and layer a node should belong.

Let us start by examining the tradeoff between the following options:

- using more layers with subgraphs that are more highly connected within each layer or
- using fewer layers with subgraphs not as highly connected in each layer.

As shown in Section 5.3.2, the performance of the hierarchical consensus is exponential with regards to the second smallest eigenvalues of the subgraphs and linear with regards to the number of layers. Therefore, choosing a decomposition with subgraphs with higher connectivity and larger $\lambda_2^{i,j}$ values for the subgraphs is advantageous even if it requires more layers. From Eqn. (5.17), we see the bound on the steady state error depends on the number of layers M. Thus for the same value of ϵ_s, using more layers increases the steady state bound. Of course by changing the value of ϵ_s, the steady state bound can be made equal for different number of layers.

Next we consider determining which subgraph and layer a node should belong to. Once again the key to the performance is increasing the $\lambda_2^{i,j}$ values. As described in Definition 5.1, to be a valid decomposition only one node from each subgraph can be present in the next layer of the hierarchy. Therefore, it may be necessary to force a node to "act dead" in a particular layer. That means the node does not participate in the consensus in that layer. In choosing which nodes should "act dead," care must be taken so that the resulting subgraph is not disconnected. Choosing amongst the subgraphs that are connected, a node should be part of a subgraph such that the corresponding $\lambda_2^{i,j}$ values are largest.

5.5 Examples

To help illustrate the analysis above and show the effectiveness of the hierarchical scheme, we present a simulation example below.

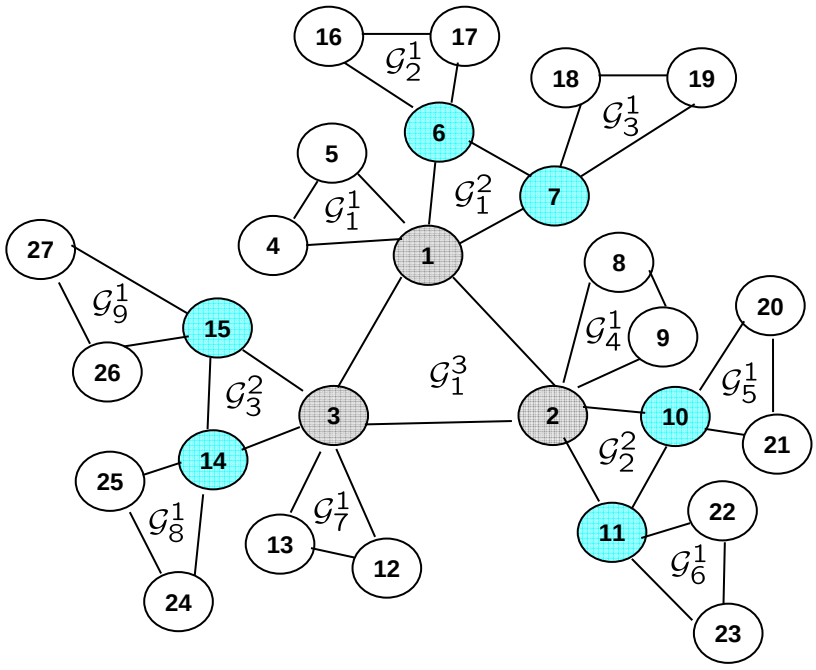

Figure 5.3: Graph with hierarchical decomposition.

Consider the 27 node hierarchical graph of Fig. 5.3. The overall graph has second smallest eigenvalue given by $\lambda_2 = 0.1625$, while all subgraphs of the hierarchical decomposition are fully connected graphs with three nodes and thus have $\lambda_2^{i,j} = 3$. This means the smaller subgraphs converge nearly 18.5 times faster than the full graph. We pick the initial conditions uniformly distributed in the interval $x_k(0) \in [0, \beta]$ with $\beta = 1000$. This means $0 \leq \overline{x} \leq 1000$, so the initial error can be bound by $\|e(0)\|_2 \leq 1000\sqrt{27}$.

Simulation results are shown in Fig. 5.4 with stopping tolerance $\epsilon_s = 10^{-5}$. The actual initial error for this simulation is $\|e(0)\|_2 \approx 281\sqrt{27}$, yet the bounds and the layer times were computed using $\|e(0)\|_2 \leq 1000\sqrt{27}$, as this is the assumed knowledge at design time. Notice the hierarchy bound is first lower than the full graph bound at roughly 16 seconds and stays below until 109 seconds where the error bound is around 10^{-4}. The actual error the hierarchy scheme goes below the full graph scheme at roughly 15.3 seconds and stays below until after 150 seconds, thus showing the actual performance is even better than the bounds indicate. In Fig. 5.5 we plot the upper-bound of $\|e(t)\|_2$ for the hierarchy scheme for different values of ϵ_s. Notice how the time at which the hierarchy bound is first below the standard scheme bound increases and the steady state value of the hierarchical scheme decreases as ϵ_s decreases.

Now we compare different hierarchical decompositions to illustrate how to best choose a decomposition and to aid the discussion presented in Section 5.4. We once again use the graph in Fig. 5.3 in these examples. First we analyze the option of choosing the number of layers to use. Then we

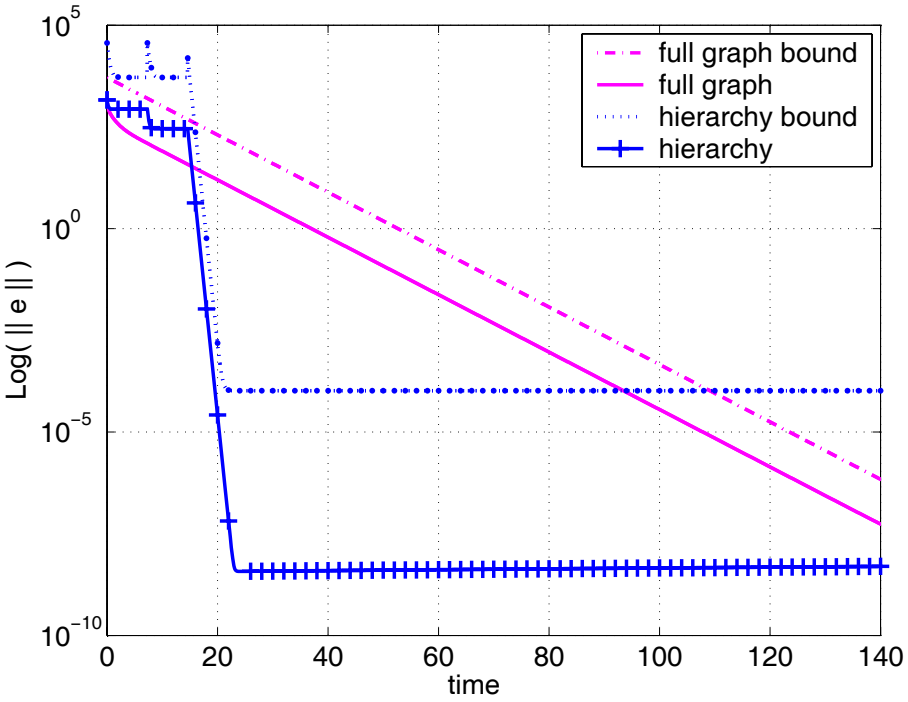

Figure 5.4: Error results for $\epsilon_s = 10^{-5}$ and $\|e(0)\|_2 \approx 281\sqrt{27}$.

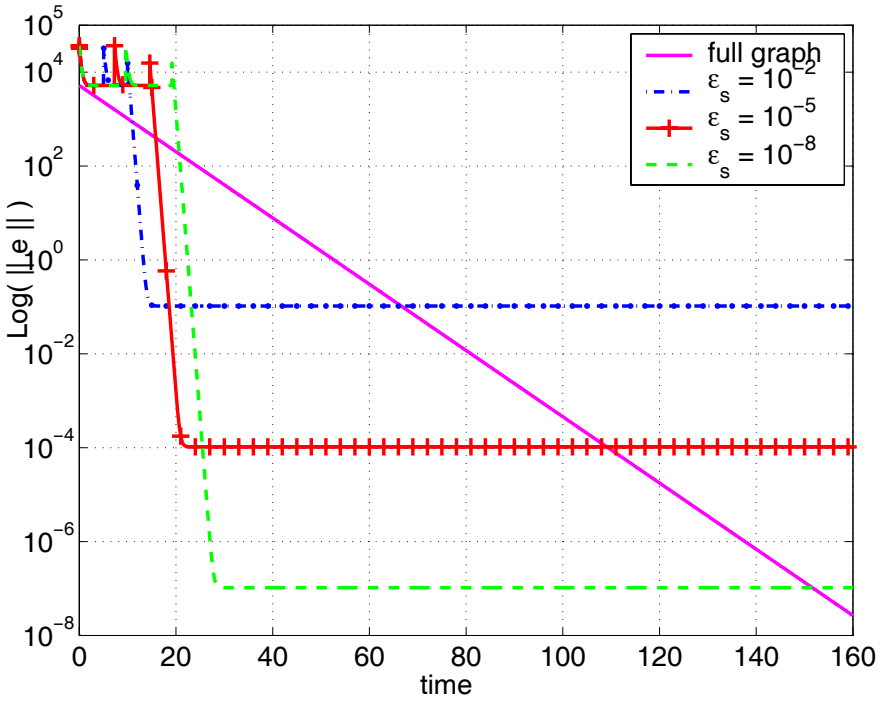

Figure 5.5: Error bounds for different values of ϵ_s.

include a few additional links into the graph topology that require decisions be made about which subgraphs certain nodes should belong to.

We begin by analyzing the choice of number of layers. The three-layer decomposition that is depicted in Fig. 5.3 and used in the simulations above is one possible decomposition. Another alternative is to use only two layers with the first layer subgraphs being given by the node sets $V_1^1 = \{1, 4, 5, 6, 7, 16, 17, 18, 19\}$, $V_2^1 = \{2, 8, 9, 20, 21, 22, 23\}$, and $V_3^1 = \{3, 12, 13, 14, 15, 24, 25, 26, 27\}$ and the second layer as $V_1^2 = \{1, 2, 3\}$. This simply combines the first and second layers of the three-layer decomposition. This two-layer decomposition is referred to as having a final layer that is "fast," since the second smallest eigenvalue of final layer for this configuration is identical to that of the original three-layer decomposition, i.e., $\lambda_2^2 = 3$. The subgraphs for the first-layer of this decomposition all have second smallest eigenvalue equal to 0.5505, much slower than the three-layer decomposition. One could also use a two-layer configuration whose first layer is the same as the three-layer configuration and second layer combines the second and third layers of the three-layer decomposition, thus with final layer node set given by $V_1^2 = \{1, 2, 3, 6, 7, 8, 9, 14, 15\}$. This two-layer decomposition is referred to as having a final layer that is "slow," since the second smallest eigenvalue of final layer for this configuration is slower than that of the original three-layer decomposition, i.e., $\lambda_2^2 = 0.5505$.

The bounds for these three possible decompositions are shown in Fig. 5.6. As expected, since the subgraphs of the three-layer decomposition have the largest λ_2 values, this decomposition gives the best performance, in terms of time to "convergence." Of course, as discussed above, it has a larger steady state error due to the additional layer. Comparing the two different choices for the two-layer decompositions, we see that the one with the "fast" final layer performs. This is because it begins the final layer consensus much later due to the longer time spent in the first layer as a result of the slower subgraphs in that layer. It is interesting to note that the two-layer decomposition with "slow" final layer actually has smaller error than the three-layer decomposition for a small period of time. This is due to the final layer beginning sooner for this decomposition than the three-layer decomposition. Since the λ_2 value of the final layer for the three-layer decomposition is much larger than the "slow" two-layer decomposition, 3 to 0.5505, the error for the three-layer decomposition eventually is better than the "slow" two-layer decomposition.

Next we analyze a case where we must decide to which subgraph nodes must belong. Once again consider the graph in Fig. 5.3 but with the outer node groups all connected in a ring by including additional links between the node pairs $(4, 27)$, $(5, 16)$, $(8, 19)$, $(9, 20)$, $(21, 22)$, $(12, 23)$, $(13, 24)$, and $(25, 26)$. Fig. 5.7 shows the graph topology with the extra links as dashed red lines.

Due to these additional links we have more options in choosing the subgraphs. We consider three different decompositions. We still have as an option the original three-layer decomposition used above. This effectively ignores the extra links and does not use them to pass information between the outer layer subgraphs. Recall each subgraph of this decomposition has second smallest eigenvalue equatl to 3. The first reorganized decomposition considered is a two-layer decomposition

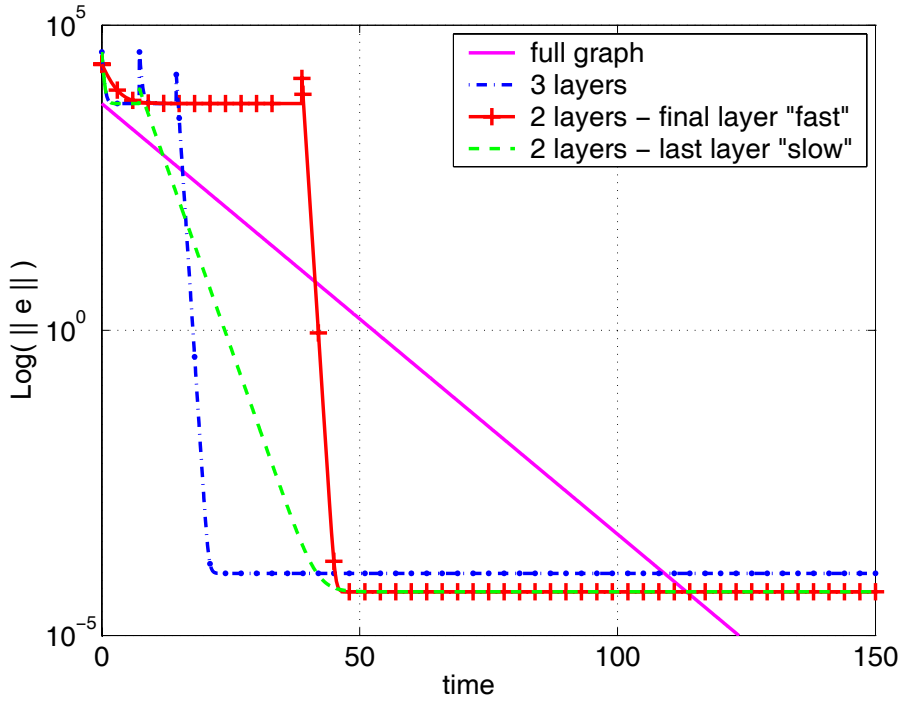

Figure 5.6: Error bounds for different decompositions using different number of layers.

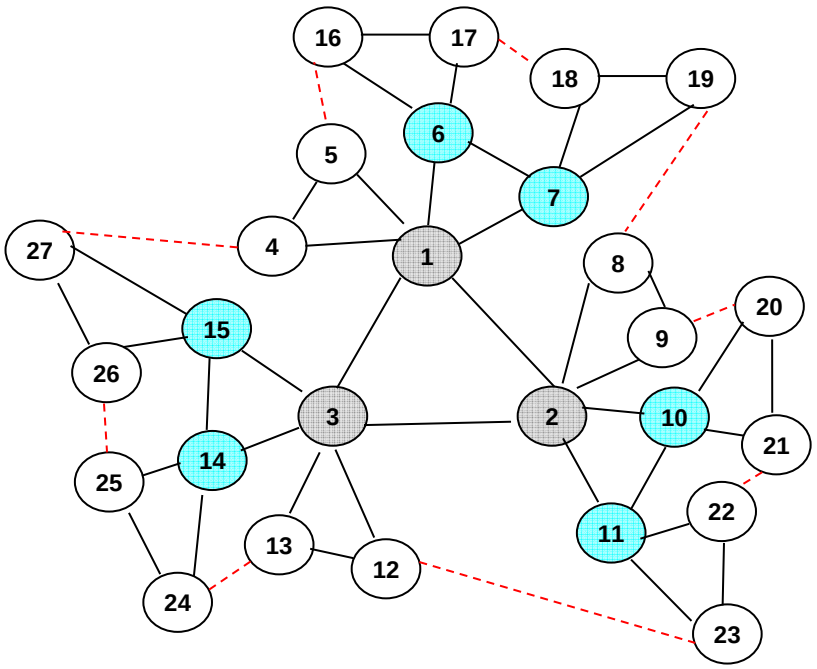

Figure 5.7: Graph topology with extra links.

whose final layer is simply the subgraph with $V_1^2 = \{1, 2, 3\}$. The first layer is given by the subgraphs with node sets $V_1^1 = \{1, 5, 6, 7, 8, 16, 17, 18, 19\}$, $V_2^1 = \{2, 9, 10, 11, 12, 20, 21, 22, 23\}$, and $V_3^1 =$

$\{3, 4, 13, 14, 24, 25, 15, 26, 27\}$. The first layer subgraphs all have second smallest eigenvalue of 0.5482. The second reorganized decomposition uses three-layers again with the final layer consisting of $V_1^3 = \{1, 2, 3\}$. The middle layer is the same as the original decomposition. The difference is in the initial layer where the nodes $\{4, 5, 8, 9, 12, 13\}$ are assigned to the subgraphs that do not contain the central nodes $\{1, 2, 3\}$. This also requires ignoring the links between nodes $(4, 5)$, $(8, 9)$, and $(12, 13)$. Thus we have the first layer node sets $V_1^1 = \{5, 6, 7, 8, 16, 17, 18, 19\}$, $V_2^1 = \{9, 10, 11, 12, 20, 21, 22, 23\}$, $V_3^1 = \{4, 13, 14, 15, 24, 25, 26, 27\}$, $V_4^1 = \{1\}$, $V_5^1 = \{2\}$ and $V_6^1 = \{3\}$. These first layer subgraphs (excluding the single-node subgraphs) all have second smallest eigenvalue of 1. The bounds for these different decompositions are shown in Fig. 5.8. As we can see the original decomposition performs best. This owes once again to the fact that the λ_2 values for this decomposition are larger than the rest.

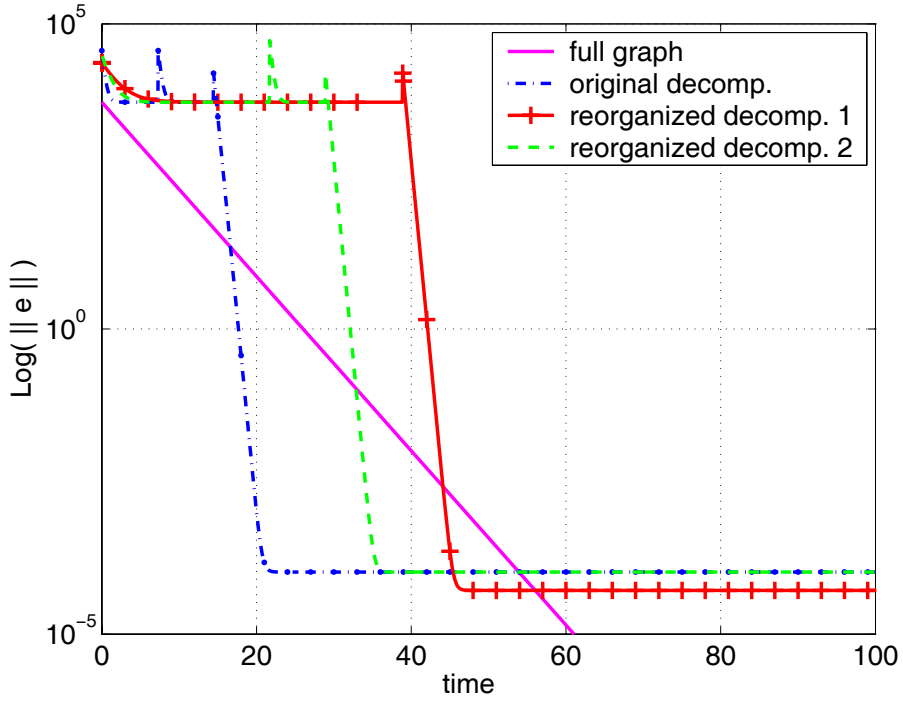

Figure 5.8: Error bounds for different decompositions depending on the subgraph assignment of certain nodes.

5.6 Conclusions and Future Work

In this work we introduced the hierarchical consensus scheme. We showed that by allowing subgraphs of a larger connected graph to converge separately and then joining the leaders of the subgraph to the larger graph, the overall time to consensus can be reduced. We showed the key parameters that determine the performance of the scheme and used examples to show the effectiveness.

There are many avenues to pursue in the future. We only analyzed the broadcast feature for disseminating the information from leader nodes to followers. Naturally one could analyze the case where the follower nodes still run consensus, treating the leaders' information as inputs to their layer. Adapting the algorithm to the case with non-static input values at the nodes could be very interesting. In this chapter we ignored the issue of non-unique hierarchical decompositions for a given graph. Determining a way to optimally choose how to hierarchically decompose a graph into layers and subgraphs would be very valuable to make the algorithm more applicable. We could also include various network effects such as delayed and dropped information in the analysis or consider discrete time consensus.

5.7 Appendix: Proofs

Proof: *[Lemma 5.2]* For any $i > 1$ and node k let

$$\delta_k^{i-1} = \widetilde{\mathcal{N}}_k^{i-1} - \overline{\mathcal{N}}_k^{i-1} \ , \tag{5.27}$$

and assume $\left|\delta_k^{i-1}\right| \leq \delta^{i-1}$. We want to first show that

$$\left|\widetilde{\mathcal{N}}_k^i - \overline{\mathcal{N}}_k^i\right| = \left|\delta_k^i\right| \leq \delta^{i-1} + \epsilon_s \ . \tag{5.28}$$

Assume $\mathcal{L}_k^i \in V_j^i$, then from Eqn. (5.11) we have

$$\widetilde{\mathcal{N}}_k^i \ = \ \frac{1}{\left|V_j^i\right|} \sum_{m \in V_j^i} x_m(t_{i-1}^+) \ .$$

From the hierarchy scheme layer stopping criterion we know $x_k(t_{i-1}^-) - \overline{S}_j^{i-1} = \Delta_k^{i-1} \in [-\epsilon_s, \epsilon_s]$. Combining this with Eqns. (5.9), (5.11), and (5.27), we can write

$$\begin{aligned} x_k(t_{i-1}^+) &= \alpha_k^i x_k(t_{i-1}^-) \\ &= \alpha_k^i \left(\overline{S}_j^{i-1} + \Delta_k^{i-1}\right) \\ &= \alpha_k^i \left(\widetilde{\mathcal{N}}_k^{i-1} + \Delta_k^{i-1}\right) \\ &= \alpha_k^i \left(\overline{\mathcal{N}}_k^{i-1} + \delta_k^{i-1} + \Delta_k^{i-1}\right) \ . \end{aligned}$$

Combining the two equations above we get

$$\widetilde{\mathcal{N}}_k^i = \frac{1}{\left|V_j^i\right|} \sum_{m \in V_j^i} \alpha_m^i \overline{\mathcal{N}}_m^{i-1} + \frac{1}{\left|V_j^i\right|} \sum_{m \in V_j^i} \alpha_m^i \left(\delta_m^{i-1} + \Delta_m^{i-1}\right) \ . \tag{5.29}$$

Using Eqns. (5.9) and (5.10) and expanding the first term on the right hand side, along with the implication of Eqn. (5.2) we see:

$$\begin{aligned}
\frac{1}{\left|V_j^i\right|}\sum_{m\in V_j^i}\left|\mathbb{V}_j^i\right|^{-1}\left|V_j^i\right|\left|\mathcal{N}_m^{i-1}\right|\left(\frac{1}{\left|\mathcal{N}_m^{i-1}\right|}\sum_{l\in\mathcal{N}_m^{i-1}}x_l(0)\right) &= \left|\mathbb{V}_j^i\right|^{-1}\sum_{m\in V_j^i}\sum_{l\in\mathcal{N}_m^{i-1}}x_l(0)\\
&= \left|\mathbb{V}_j^i\right|^{-1}\sum_{m\in\mathbb{V}_j^i}x_m(0)\\
&= \left|\mathcal{N}_k^i\right|^{-1}\sum_{m\in\mathcal{N}_k^i}x_m(0)\\
&= \overline{\mathcal{N}}_k^i\ .
\end{aligned}$$

Plugging this result into Eqn. (5.29), we see

$$\widetilde{\mathcal{N}}_k^i-\overline{\mathcal{N}}_k^i=\frac{1}{\left|V_j^i\right|}\sum_{m\in V_j^i}\alpha_m^i\left(\delta_m^{i-1}+\Delta_m^{i-1}\right)\ .$$

To bound this expression, follow similar derivations as above to see that

$$\begin{aligned}
\left|\frac{1}{\left|V_j^i\right|}\sum_{m\in V_j^i}\alpha_m^i\left(\delta_m^{i-1}+\Delta_m^{i-1}\right)\right| &= \left|\mathbb{V}_j^i\right|^{-1}\sum_{m\in V_j^i}\left|\mathcal{N}_m^{i-1}\right|\left|\delta_m^{i-1}+\Delta_m^{i-1}\right|\\
&\leq \left(\delta^{i-1}+\epsilon_s\right)\left|\mathbb{V}_j^i\right|^{-1}\sum_{m\in V_j^i}\left|\mathcal{N}_m^{i-1}\right|\\
&= \delta^{i-1}+\epsilon_s\ .
\end{aligned}$$

Combining the appropriate terms we arrive at Eqn. (5.28), from which we then get

$$\left\|\widetilde{\mathcal{N}}^i-\overline{\mathcal{N}}^i\right\|_2\leq\left(\delta^{i-1}+\epsilon_s\right)\sqrt{N}\ . \tag{5.30}$$

From the way they are defined in Eqns. (5.10) and (5.11), we have $\left\|\widetilde{\mathcal{N}}^1-\overline{\mathcal{N}}^1\right\|_2=0$, which implies $\delta^1=0$. It then becomes straightforward to see that $\delta^i=(i-1)\epsilon_s$, and combining with Eqn. (5.30) gives Eqn. (5.13). ■

Proof: *[Lemma 5.3]* By making use of the fact that the subgraphs at every layer are disjoint and that $\overline{\mathcal{N}}_k^i = \overline{\mathcal{N}}_j^i$ for any j and k in the same subgraph with $\mathcal{N}_i^k$ repeated $\left|\mathcal{N}_i^k\right|$ times, we can write

$$\begin{aligned}
\left\|\overline{\mathcal{N}}^i - \overline{\mathbf{x}}\right\|_2^2 &= \sum_{k=1}^{N} \left(\overline{\mathcal{N}}_k^i - \overline{x}\right)^2 \\
&= \sum_{m=1}^{S^i} \left|\mathbb{V}_m^i\right| \left[\left(\frac{1}{|\mathbb{V}_m^i|}\sum_{l\in\mathbb{V}_m^i} x_l(0)\right) - \overline{x}\right]^2 \\
&= \sum_{m=1}^{S^i} \frac{1}{|\mathbb{V}_m^i|}\left[\sum_{l\in\mathbb{V}_m^i} \left(x_l(0) - \overline{x}\right)\right]^2 \\
&= \sum_{m=1}^{S^i} \frac{1}{|\mathbb{V}_m^i|}\left[\sum_{l\in\mathbb{V}_m^i} \Delta_l\right]^2 ,
\end{aligned}$$

where we let $\Delta_k = x_k(0) - \overline{x}$. We can also use the disjoint property of the subgraphs to write

$$\begin{aligned}
\|e(0)\|_2^2 &= \|x(0) - \overline{\mathbf{x}}\|_2^2 \\
&= \sum_{k=1}^{N} \left(x_k(0) - \overline{x}\right)^2 \\
&= \sum_{k=1}^{N} \Delta_k^2 \\
&= \sum_{m=1}^{S^i} \sum_{l\in\mathbb{V}_m^i} \Delta_l^2 .
\end{aligned}$$

Combining the two equations above we get

$$\|e(0)\|_2^2 - \left\|\overline{\mathcal{N}}^i - \overline{\mathbf{x}}\right\|_2^2 = \sum_{m=1}^{S^i} \Delta e_m^i$$

with

$$\Delta e_m^i = \sum_{l\in\mathbb{V}_m^i} \Delta_l^2 - \frac{1}{|\mathbb{V}_m^i|}\left[\sum_{l\in\mathbb{V}_m^i} \Delta_l\right]^2 .$$

Noticing that $\frac{1}{|\mathbb{V}_m^i|}\Delta e_m^i$ equals the variance of the set of numbers $\{\Delta_l \mid l \in \mathbb{V}_m^i\}$, we see that indeed $\Delta e_m^i \geq 0$ for all m, implying $\|e(0)\|_2^2 - \left\|\overline{\mathcal{N}}^i - \overline{\mathbf{x}}\right\|_2^2 \geq 0$ and completing the proof. ■

Proof: *[Lemma 5.4]* The stopping criterion assures $\left\|x(t_i^-) - \widetilde{\mathcal{N}}^i\right\|_2 \leq \epsilon_s\sqrt{N}$ for $i < M$ and $\left\|x(t_M^-) - \widetilde{\mathcal{N}}^i\right\|_2 \to 0$ as $t_M^- \to \infty$. Combining this with Eqns. (5.12) and (5.13) and Lemma 5.3, we arrive at the desired results. ■

Proof: *[Lemma 5.5]* We prove this by taking the bound on the consensus error at the termination of layer $i-1$ given in Lemma 5.4 and bounding the change due to rescaling the leader node values

and assigning the follower nodes to this same value at the beginning of the new layer i. Start by writing out the nodal values at the beginning of layer i according to Eqn. (5.9):

$$\begin{aligned} x_k(t_{i-1}^+) &= \alpha^i_{\mathcal{L}^i_k} x_{\mathcal{L}^i_k}(t_{i-1}^-) \\ &= \alpha^i_{\mathcal{L}^i_k}\left(x_{\mathcal{L}^i_k}(t_{i-1}^-) - x_k(t_{i-1}^-) + x_k(t_{i-1}^-)\right) + x_k(t_{i-1}^-) - x_k(t_{i-1}^-) \\ &= x_k(t_{i-1}^-) + \left(\alpha^i_{\mathcal{L}^i_k} - 1\right) x_k(t_{i-1}^-) + \alpha^i_{\mathcal{L}^i_k}\left(x_{\mathcal{L}^i_k}(t_{i-1}^-) - x_k(t_{i-1}^-)\right) . \end{aligned}$$

The layer stopping criterion assures

$$\left|x_{\mathcal{L}^i_k}(t_{i-1}^-) - x_k(t_{i-1}^-)\right| \leq 2\epsilon_s .$$

Then, it is not too difficult to derive the bound

$$\|e(t_{i-1}^+)\|_2 \leq \|e(t_{i-1}^-)\|_2 + \left\|\tilde{\alpha}^i - 1\right\|_\infty \beta\sqrt{N} + 2\epsilon_s\sqrt{N}\left\|\tilde{\alpha}^i\right\|_\infty , \tag{5.31}$$

where we use the bound on the initial conditions to get $\|x(t_{i-1}^-)\|_2 \leq \beta\sqrt{N}$. This gives a bound on the jump in error from the end of one layer to the beginning of the next layer, and it is simply an additive jump given by the final two terms in Eqn. (5.31). Plugging in the expression for $\|e(t_{i-1}^-)\|_2$ from Eqn. (5.16), we arrive at the desired result. ■

Proof: *[Lemma 5.6]* If node k is not in layer i, then its value is identical to that of its leader hence $x_k(t_{i-1}^+) - \widetilde{\mathcal{N}}^i_k = x_{\mathcal{L}^i_k}(t_{i-1}^+) - \widetilde{\mathcal{N}}^i_{\mathcal{L}^i_k}$. Noting that $\left|\mathcal{N}^{i-1}_k\right| \geq 1$, and by definition $\tilde{e}_k = 0$ if node k is not in layer i, we can write

$$\begin{aligned} \left\|\tilde{e}(t_{i-1}^+)\right\|_2 &= \sqrt{\sum_{k\in\mathcal{V}^i} \left(\ \tilde{e}_k(t_{i-1}^+)\ \right)^2} \\ &\leq \sqrt{\sum_{k\in\mathcal{V}^i} \left(\ \left|\mathcal{N}^{i-1}_k\right|\ \tilde{e}_k(t_{i-1}^+)\right)^2} \\ &= \sqrt{\sum_{k=1}^{N}\left(x_k(t_{i-1}^+) - \widetilde{\mathcal{N}}^i_k\right)^2} \\ &= \left\|x(t_{i-1}^+) - \widetilde{\mathcal{N}}^i\right\|_2 \\ &\leq \left\|x(t_{i-1}^+) - \overline{\mathbf{x}}\right\|_2 + \left\|\overline{\mathcal{N}}^i - \widetilde{\mathcal{N}}^i\right\|_2 + \left\|\overline{\mathbf{x}} - \overline{\mathcal{N}}^i\right\|_2 \\ &= \left\|e(t_{i-1}^+)\right\|_2 + \left\|\overline{\mathcal{N}}^i - \widetilde{\mathcal{N}}^i\right\|_2 + \left\|\overline{\mathbf{x}} - \overline{\mathcal{N}}^i\right\|_2 . \end{aligned}$$

Using Eqn. (5.18) and Lemmas 5.2 and 5.3, we arrive at the value of $\widetilde{e}^i_0$ in Eqn. (5.19). ■

Proof: *[Lemma 5.7]* Noting that the follower nodes are equal to their leader's value, we can expand the subgraph error as

$$\begin{aligned}\left\|x(t)-\widetilde{\mathcal{N}}^i\right\|_2 &= \sqrt{\sum_{k=1}^{N}\left(x_k(t)-\widetilde{\mathcal{N}}_k^i\right)^2} \\ &= \sqrt{\sum_{j=1}^{S^i}\sum_{k\in V_j^i}\left|\mathcal{N}_k^{i-1}\right|\left(x_k(t)-\widetilde{\mathcal{N}}_k^i\right)^2} \\ &\leq \sqrt{\sum_{j=1}^{S^i}\max_{k\in V_j^i}\left|\mathcal{N}_k^{i-1}\right|\sum_{k\in V_j^i}\left(x_k(t)-\widetilde{\mathcal{N}}_k^i\right)^2}\end{aligned}$$

for $t\in[t_{i-1}^-,t_i^+]$. Note that the second summation term $\sum_{k\in V_j^i}\left(x_k(t)-\widetilde{\mathcal{N}}_k^i\right)^2$ is simply the square of the norm of the subgraph error for subgraph $\mathcal{G}_j^i$. Since the subgraph error from subgraph $\mathcal{G}_j^i$ is no greater than the total subgraph error $\tilde{e}$ and it decays at a rate no slower that $\lambda_2^{i,j}$, we can bound $\sum_{k\in V_j^i}\left(x_k(t)-\widetilde{\mathcal{N}}_k^i\right)^2 \leq \left(e^{-\lambda_2^{i,j}\left(t-t_{i-1}^+\right)}\tilde{e}_0^i\right)^2$. Plugging this into the expression above yields Eqn. (5.20). ■

Chapter 6

Conclusions and Future Work

6.1 Discussion and Summary

In this thesis I proposed and analyzed algorithms for managing information flow for several NCS scenarios: state estimation with lossy measurement signals, using input buffers to reduce the frequency of communication with a remote plant, and performing state estimation when control signals are transmitted to a remote plant via a lossy communication link with no acknowledgement signal at the estimator. I also explored the performance impact of managing information flow in the area of multi-agent systems. Focusing on the problem of multi-agent average consensus, I proposed an algorithm based on a hierarchical decomposition of the communication topology to speed up the time to convergence. For all these topics I focused on designing intuitive algorithms that intelligently manage the information flow and provided analysis and simulations to illustrate their effectiveness.

Chapter 2 analyzed the problem of state estimation where measurement packets are sent across a lossy network. An estimator algorithm was designed that is guaranteed to have an upper bound on the estimation error covariance whenever a measurement packet is received. The algorithm relies on transmitting buffered measurements consisting of current and several previous sensor measurements. This allows a probabilistic performance description of the estimator that is different than the expected value performance description commonly used.

Chapter 3 also considers using buffers in an NCS setting. Here the control signals are sent across a network to the plant. In this setting it was assumed the network did not lose any packets; rather, the goal was to design a system that sent less frequent but more informative information packets, thus making better use of a possibly shared communication resource. The data in each control packet contains the control signal for the current time step as well as a buffered sequence of predicted future control signals. In order to determine when to transmit packets, the Input Difference Transmission Scheme, which transmits only when the newly computed control signal and the previously transmitted value differ by a given threshold, was presented. The analysis of the algorithm showed how certain design parameters affect performance of the algorithm.

In Chapter 4, estimation algorithms for NCS that transmit control signals via a lossy UDP-like network with no acknowledgement signal were explored. A novel estimation scheme consisting of a mode detector and state observer was developed. To ensure detection of the fate of the control packet, an added control input was included. An upper bound for the expected value of the state norm in the presence of bounded state and measurement noise was presented. If the added control input is removed, the estimator algorithm is no longer guaranteed to detect the fate of the control packet. Nonetheless, under certain conditions on the system parameters it can still be shown to produce an upper bound to the estimation error, which allows an upper bound to the expected value of the state norm to be derived. The estimator algorithm is then compared to the unknown input observer, which can directly remove the dependence of the control signal on the estimation error.

In Chapter 5, the hierarchical consensus scheme was introduced to manage the information flow in multi-agent average consensus to improve performance. We showed that by allowing subgraphs of a larger connected graph to converge separately and then joining the leaders of the subgraph to the larger graph, the overall time to consensus can be reduced. We showed the key parameters that determine the performance of the scheme and used examples to show the effectiveness.

6.2 Future Directions

The fields of Networked Control Systems and multi-agent systems are still emerging areas. Most of the current research investigating the stability of NCS seems to be fragmented into particular system descriptions, i.e., the location (in the sensor-to-controller and/or controller-to-plant loops) and type of network (bit-limited, subject to delays and/or losses) being used. Creating a general framework for arbitrary NCS scenarios that can be reduced to the specific previously studied cases would be highly valuable. Despite the bevy of recent results on stability of NCS, there have been far fewer related to the performance of such systems. Perhaps a probabilistic performance description similar to the one presented in Chapter 2 can serve as a good starting point. Multi-agent systems have become more popular as control objectives increase in complexity and task. The field should continue to advance and find new applications in the near future.

The work presented in this thesis makes contributions to specific problem settings in NCS and multi-agent systems, and there are extensions possible for each of the cases considered. The discussion should begin with a few areas applicable to all the problem settings considered. The models of the system dynamics and random packet losses used throughout this thesis were kept simple. The dynamical systems were restricted to be linear time invariant. Certainly applying and adapting these algorithms to problem settings with nonlinear and perhaps time varying systems would be an area that would make them applicable to a wider range of systems. Allowing for different types

of uncertainties would do the same. For the situations that considered only i.i.d. random packet losses, allowing for more realistic models of these losses is an area to be pursued. As mentioned in the beginning of this thesis, the issues of delayed and quantized information were summarily ignored. These effects could be incorporated into the problem settings considered. In addition to these general directions, each of the individual algorithms has specific areas that can be pursued.

The state estimation setting and algorithm presented in Chapter 2 can be explored further. How the new probabilistic performance description of the estimator couples into the closed loop performance with a controller using these estimates is an interesting topic. Furthermore, extending the concept of probabilistic performance to give a description of the closed loop state would be quite useful.

The Input Difference Transmission Scheme presented in Chapter 3 was used to reduce the transmission frequency of control packets. It was assumed that the full and exact state value was available for calculating the control law. One could consider the effect of measurements taken from noisy sensors and using an observer to produce an estimate of the state. In this work it was assumed that the state feedback controller, F, was designed without regard to the network considerations. The effect that the choice of F has on the network performance could be investigated in further detail. It would also be interesting to consider more general Model Predictive Controllers in place of the anticipative controller.

The estimation algorithm presented in Chapter 4 has topics to pursue in the future. Since the sufficient conditions that were derived for stability of the estimator algorithm overlap with those of the unknown input observer, it would be useful if necessary stability conditions for the estimator algorithm could be derived. Since the algorithm appears to work well even if the sufficient conditions are not satisfied, relaxing this conservatism would make the result stronger. The results presented here should only need to be modified slightly to include additional intelligence at the plant to apply some predicted control value rather than evolve without applying a control signal when the control packet is dropped. The most interesting extension might be to insert a network between the sensors and estimator so that the estimator does not always have access to the sensor data, which would most likely require some additional logic in the algorithm.

In Chapter 5, the hierarchical consensus scheme was introduced. The most obvious extension here is to consider the case where the leaders cannot simply broadcast back down to their followers, but rather the follower nodes still run consensus, treating the leaders' information as inputs to their layer. Adapting the algorithm to the case with non-static input values at the nodes could be very interesting. In this chapter we briefly discussed ways to choose from the set of non-unique hierarchical decompositions for a given graph. Determining a way to optimally choose how to hierarchically decompose a graph into layers and subgraphs would be very valuable to make the

algorithm perform even better. Extending these results to the discrete time consensus problem would be highly valuable.

Bibliography

[1] P. J. Antsaklis and J. Baillieul(editors), "Special Issue: Technolog of Networked Control Systems," *Proceedings of the IEEE, Special Issue on Networked Control Systems*, vol. 95, no. 1, 2007.

[2] B. Azimi-Sadjadi, "Stability of Networked Control Systems in the Presence of Packet Losses," in *IEEE Conference on Decision and Control*, 2003.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distriubted Computation.* Prentice-Hall, 1989.

[4] V. Borkar and P. Varaiya, "Asymptotic Agreement in Distributed Estimation," *IEEE Trans. on Automatic Control*, vol. 27, no. 3, pp. 650–655, June 1982.

[5] M. S. Branicky, S. M. Phillips, and W. Zhang, "Stability of Networked Control Systems: Explicit Analysis of Delay," in *American Control Conference*, June 2000.

[6] R. W. Brockett and D. Liberzon, "Quantized Feedback Stabilization of Linear Systems," *IEEE Trans. on Automatic Control*, vol. 45, no. 7, pp. 1279–1289, July 2000.

[7] F. Bullo and D. Liberzon, "Quantized Control Via Locational Optimization," *IEEE Trans. on Automatic Control*, vol. 51, no. 1, pp. 2–13, January 2006.

[8] G. Cybenko, "Dynamic Load Balancing for Distributed Memory Multi-processors," *Journal of Parallel and Distributed Computing*, vol. 7, no. 2, pp. 279–301, October 1989.

[9] M. Das, R. Ghosh, B. Goswami, A. Gupta, A. P. Tiwari, R. Balasubrmanian, and A. K. Chandra, "Network Control System Applied to a Large Pressurized Heavy Water Reactor," *IEEE Trans. on Nuclear Science*, vol. 53, no. 5, pp. 2948–2956, October 2006.

[10] D. F. Delchamps, "Stabilizing a Linear System with Quantized State Feedback," *IEEE Trans. on Automatic Control*, vol. 35, no. 8, pp. 916–924, August 1990.

[11] N. Elia and S. K. Mitter, "Stabilization of Linear Systems with Limited Information," *IEEE Trans. on Automatic Control*, vol. 46, no. 9, pp. 1384–1400, September 2001.

[12] M. Epstein, K. M. Lynch, K. H. Johansson, and R. M. Murray, "Using Hierarchical Decomposition to Speed Up Average Consensus," in *IFAC World Congress*, 2008 (submitted).

[13] M. Epstein, L. Shi, S. D. Cairano, and R. M. Murray, "Control Over a Network: Using Actuation Buffers and Reducing Transmission Frequency," in *European Control Conference*, 2007.

[14] M. Epstein, L. Shi, and R. M. Murray, "An Estimation Algorithm for a Class of Networked Control Systems Using UDP-Like Communication Schemes," in *IEEE Conference on Decision and Control*, 2006.

[15] ——, " Estimation Schemes for Newtorked Control Systems Using UDP-Like Communication," in *IEEE Conference on Decision and Control*, 2007.

[16] M. Epstein, L. Shi, A. Tiwari, and R. M. Murray, "Probabilistic Performance of State Estimation Across a Lossy Network," *Automatica*, 2008 (to appear).

[17] Y. Fang, "A new general sufficient condition for almost sure stability of jump linear systems," *IEEE Trans. on Automatic Control*, vol. 42, no. 3, pp. 378–382, March 1997.

[18] Y. Fang and K. A. Loparo, "Stochastic stability of jump linear systems," *IEEE Trans. on Automatic Control*, vol. 47, no. 7, pp. 1204–1208, July 2002.

[19] J. A. Fax and R. M. Murray, "Information Flow and Cooperative Control of Vehicle Formations," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, September 2004.

[20] R. Freeman, P. Yang, and K. M. Lynch, "Stability and Convergence Properties of Dynamic Average Consensus Estimators," in *IEEE Conference on Decision and Control*, 2006.

[21] D. Georgieve and D. M. Tillbury, "Packet-based Control: The H_2-Optimal Solution," *Automatica*, vol. 42, no. 1, pp. 137–144, January 2006.

[22] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer-Verlag, 2001.

[23] V. Gupta, B. Hassibi, and R. M. Murray, "Optimal LQG Control Across Packet-Dropping Links," *Systems and Control Letters*, vol. 56, no. 6, pp. 439–446, June 2007.

[24] J. Hespanha, A. Ortega, and L. Vasudevan, "Towards the Control of Linear Systems with Minimum Bit-Rate," in *15th Int. Symp. on the Mathematical Theory of Networks and Systems*, 2002.

[25] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A Survey of Recent Results in Networked Control Systems," *Proceedings of the IEEE, Special Issue on Networked Control Systems*, vol. 95, no. 1, pp. 138–162, January 2007.

[26] R. A. Horn and C. R.Johnson, *Matrix Analysis*. Cambridge University Press, 1985.

[27] O. C. Imer, S. Yuksel, and T. Basar, "Optimal Control of LTI Systems Over Unreliable Communication Links," *Automatica*, vol. 42, no. 9, pp. 1429–1439, September 2006.

[28] H. Ishii and B. A. Francis, "Quadratic Stabilization of Sampled-Data Systems with Quantization," *Automatica*, vol. 39, no. 10, pp. 1793–1800, October 2003.

[29] Z. Jin, S. Waydo, E. Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held, and R. M. Murray, "MVWT-II: The Second Generation Caltech Multi-Vehicle Wireless Testbed," in *American Control Conference*, 2004.

[30] A. Kashay, T. Basar, and R. Srikant, "Quantized Consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, July 2007.

[31] P. A. Kawka and A. G. Alleyne, "Stability and Performance of Packet-Based Feedback Control Over a Markov Channel," in *American Control Conference*, 2006.

[32] F. L. Lian, J. R. Moyne, and D. M. Tillbury, "Performance Evaluation of Control Networks," *IEEE Control Systems Magazine*, vol. 21, pp. 66–83, February 2001.

[33] L. W. Liou and A. Ray, "A Stochastic Regulator for Integrated Communication and Control Systems: Part I - Formulation of Control Law," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 113, no. 4, pp. 604–611, December 1991.

[34] ——, "A Stochastic Regulator for Integrated Communication and Control Systems: Part II - Numerical Analysis and Simulation," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 113, no. 4, pp. 612–619, December 1991.

[35] X. Liu and A. Goldsmith, "Kalman Filtering with Partial Observation Losses," in *IEEE Conference on Decision and Control*, Dec 2004.

[36] R. Luck and R. Asok, "An Observer-based Compensator for Distributed Delays," *Automatica*, vol. 26, no. 5, pp. 903–908, September 1990.

[37] N. C. Martins and M. A. Daleh, "Fundamental Limitations of Performance in the Presence of Finite Capacity Feedback," in *American Control Conference*, June 2005.

[38] ——, "Feedback Control in the Presence of Noisy Channels: "Bode-Like Fundamental Limitations of Performance," *IEEE Trans. on Automatic Control*, May 2008(to appear).

[39] A. S. Matveev and A. V. Savkin, "The Prbolem of State Estimation via Asynchronous Communication Channels with Irregular Transmsiion Times," *IEEE Trans. on Automatic Control*, vol. 48, no. 4, pp. 670–676, April 2003.

[40] M. Mehyar, D. Spano, J. Pongsajapan, S. H. Low, and R. R. Murray, "Asynchronous Distributed Averaging on Communication Networks," *IEEE/ACM Trans. on Networking*, vol. 15, no. 3, pp. 512–520, June 2007.

[41] G. Millerioux and J. Daafouz, "Unknown Input Observers for Switched Linear Discrete Time Systems," in *American Control Conference*, 2004.

[42] L. A. Montestruque and P. J. Antsaklis, "Model-Based Networked Control Systems: Necessary and Sufficient Conditions for Stability," in *10th Mediterranean Contol Conference*, 2002.

[43] ——, "State and Output Feedback Control in Model-Based Networked Control Systems," in *IEEE Conference on Decision and Control*, 2002.

[44] ——, "On the Model-Based Control of Networked Systems," *Automatica*, vol. 39, no. 10, pp. 1837–1843, October 2003.

[45] ——, "Stability of Model Based Network Control Systems with Time Varying Transmission Times," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1562–1572, September 2004.

[46] L. Moreau, "Stability of Multiagent Systems With Time-Dependent Communciation Links," *IEEE Trans. on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[47] R. M. Murray(editor), *Control in an Information Rich World*. SIAM, 2003.

[48] P. Naghshtabrizi and J. P. Hespanha, "Anticipative and Non-Anticipative Controller Design for Network Control Systems," *Networked Embedded Sensing and Control*, pp. 203–218, 2006.

[49] G. N. Nair and R. J. Evans, "Communication-Limited Stabilization of Linear Systems," in *IEEE Conference on Decision and Control*, December 2000.

[50] ——, "Exponential Stabilisability of Finite-Dimensional Linear Systems with Limted Data Rates," *Automatica*, vol. 39, no. 4, pp. 585–593, April 2003.

[51] ——, "Stabilizability of Stochastic Linear Systems with Finite Feedback Data Rate," *SIAM Journal on Control and Optimization*, vol. 43, no. 2, pp. 413–436, July 2004.

[52] ——, "Topological Feedback Entropy and Nonlinear Stabilization," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1585–1597, September 2004.

[53] D. Nesic and A. R. Teel, "Input-Output Stability Properties of Networked Control Systems," *Automatica*, vol. 49, no. 10, pp. 1650–1667, October 2004.

[54] J. Nilsson, "Real Time Control Systems with Delay," Ph.D. dissertation, Lund Institute of Technology, 1998.

[55] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment," *IEEE Trans. on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, August 2004.

[56] R. Olfati-Saber, "Ultrafast Consensus In Small-World Networks," in *American Control Conference*, 2005.

[57] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE, Special Issue on Networked Control Systems*, vol. 95, no. 1, pp. 215–233, 2007.

[58] R. Olfati-Saber and R. M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1520–2533, September 2004.

[59] I. R. Petersen and A. V. Savkin, "Multi-Rate Stabilization of Multivariable Discrete-time Linear Systems Via a Limited Capacity Communication Channel," in *IEEE Conference on Decision and Control*, Dec 2001.

[60] V. N. Phat, J. Jiang, A. V. Savkin, and I. R. Petersen, "Robust Stabalization of Linear Uncertain Discrete-Time Systems Via a Limited Capacity Communication Channel," *Systems and Control Letters*, vol. 53, no. 5, pp. 347–360, December 2004.

[61] N. J. Ploplys and A. G. Alleyne, "UDP Network Communications for Distributed Wireless Control," in *American Control Conference*, 2003.

[62] W. Ren and R. W. Beard, "Consensus Seeking in Multi-Agent Systems Under Dynamically Changing Interaction Topologies," *IEEE Trans. on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[63] W. Ren, R. W. Beard, and E. Atkins, "Information Consensus in Multivehicle Cooperative Control: Collective Group Behaviour Through Local Interaction," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71–82, April 2007.

[64] C. L. Robinson, G. Baliga, S. Graham, and P. K. Kumar, "Design Patterns for Robust and Evolvable Networked Control," in *Conference on Systems Engineering Research*, 2005.

[65] C. L. Robinson and P. K. Kumar, "Control Over Networks of Unreliable Links - Controller Location and Performance Bounds," in *Proceedings of Control Over Communication Channels (ConComm)*, 2007.

[66] A. Sahai, "Anytime Information Theory," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[67] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry, "Foundations of Control and Esimation over Lossy Networks," *Proceedings of the IEEE, Special Issue on Networked Control Systems*, vol. 95, no. 1, pp. 163–187, January 2007.

[68] P. Seiler and R. Sengupta, "Analysis of Communication Losses in Vehicle Control Problems," in *American Control Conference*, June 2001.

[69] ——, "A Bounded Real Lemma for Jump Systems," *IEEE Trans. Automatic Control*, vol. 48, no. 9, pp. 1651–1654, September 2003.

[70] ——, "An H_∞ Approach to Networked Control," *IEEE Trans. on Automatic Control*, vol. 50, no. 3, pp. 356–364, March 2005.

[71] L. Shi, M. Epstein, and R. M. Murray, "Towards robust control over a packet dropping network," in *Mathematical Theory of Networks and Systems*, 2006.

[72] L. Shi, M. Epstein, A. Tiwari, and R. M. Murray, "Estimation With Information Loss: Asymptotic Analysis and Error Bounds," in *IEEE Conference on Decision and Control*, Dec 2005.

[73] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry, "Kalman Filtering with Intermittent Observations," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, September 2004.

[74] ——, "Optimal Control with Unreliable Communication: the TCP Case," in *American Control Conference*, 2005.

[75] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S. Sastry, "An LQG Optimal Linear Controller for Control Systems with Packet Losses," in *IEEE Conference on Decision and Control*, 2005.

[76] ——, "LQG Control With Missing Observation and Control Packets," in *IFAC World Congress*, 2005.

[77] S. C. Smith and P. Seiler, "Estimation with Lossy Measurements: Jump Estimators for Jump Systems," *IEEE Trans. on Automatic Control*, vol. 48, no. 12, pp. 2163–2171, December 2003.

[78] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Distributed Sensor Fusion Using Dynamic Consensus," in *IFAC World Congress*, 2005.

[79] ——, "Dynamic Consensus on Mobile Networks," in *IFAC World Congress*, 2005.

[80] R. F. Stengel, *Optimal Control and Estimation.* Dover Publications, 1994.

[81] S. C. Tatikonda, "Control Under Communication Constraints," Ph.D. dissertation, Massachusetts Institute of Technology, 2000.

[82] J. N. Tsitsiklis, "Problems in Decentrailzed Decision Making and Computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[83] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms," *IEEE Trans. on Automatic Control*, vol. 31, no. 9, pp. 803–812, September 1986.

[84] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability Analysis of Networked Control Systems," *IEEE Trans. on Control Systems Technology*, vol. 10, no. 3, pp. 438–446, May 2002.

[85] W. S. Wong and R. W. Brockett, "Systems with Finite Communication Bandwidth-Part I: State Estimation Problems," *IEEE Trans. Automatic Control*, vol. 42, no. 9, pp. 1294–1299, September 1997.

[86] ——, "Systems with Finite Communication Bandwidth-Part II: Stabilization with Limited Information Feedback," *IEEE Trans. Automatic Control*, vol. 44, no. 5, pp. 1049–1053, May 1999.

[87] W. Xi, X. Tan, and J. S. Baras, "A Stochastic Algorithm for Self-Organization of Autonomous Swarms," in *IEEE Conference on Decision and Control*, 2005.

[88] L. Xiao and S. Boyd, "Fast Linear Iterations for Distributed Averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, September 2004.

[89] Y. Xu and J. P. Hespanha, "Optimal Communication Logics in Networked Control Systems," in *IEEE Conference on Decision and Control*, 2004.

[90] P. Yang, R. Freeman, and K. M. Lynch, "Optimal Information Poropagation in Sensor Networks," in *International Conference on Robotics and Automation*, 2006.

[91] J. K. Yook, D. M. Tillbury, and N. R. Soparkar, "Trading Computation for Bandwidth: Reducing Communication in Distributed Control Systems Using State Estimators," *IEEE Trans. on Control Systems Technology*, vol. 10, no. 4, July 2002.

[92] L. Zhang and D. Hristu-Varsakelis, "Communication and Control Co-Design for Networked Control Systems," *Automatica*, vol. 42, no. 6, pp. 953–958, June 2006.