

# Decentralized Estimation for Stochastic Multi-Agent Systems

Fayette W. Shaw

A dissertation submitted in partial fulfillment  
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2012

Eric Klavins, Chair

Brian Fabien

Mehran Mesbahi

Program Authorized to Offer Degree: Mechanical Engineering

## ACKNOWLEDGMENTS

I thank Eric Klavins for his guidance, support, and patience. He encouraged me to pursue my own interests, form my own collaboration, and sew my own testbed. I thank my supportive labmates, the Self-Organizing Systems Lab, particularly Nils Napp, Josh Bishop, and Shelly Jang. Thanks to Alex Leone for collecting data on the PPT.

I thank my committee members, Mehran Mesbahi, Brian Fabien, Shwetak Patel, and Martin Berg, for their valuable feedback.

I thank James McLurkin of the Multi-Robot Systems Lab at Rice for being my collaborator and Albert Chiu and Alvin Chou for data collection on the SwarmBot testbed. Their efforts lead to the results presented in Sec. 5.7.

The work done in Ch. 7 was the result of tremendous effort. I thank Andrew Lawrence for development on Grouper and immeasurable help in the Ubicomp 2010 demo. I thank the Grouper Sweatshop: Noor Martin, Tam Minor, and Landon Meernik, who fabricated the wearable wrist modules. I thank the UbiComp lab, particularly Eric Larson, for their expertise and valuable feedback. I thank my user studies participants, many of whom are already listed here.

I thank the Ladies in Engineering Graduate Studies for excellent peer mentorship during the trials and tribulations of graduate school. I thank my family (and my surrogate family) for all their support: Brenda, Theodore, and Vivian Shaw; the Coopers; my “big sister” Liz Sacho, my “little sister” Ann Huang, and my knitting friends: Aubri, Desiree, Beth, and Noor, the best friends anyone could ask for.

And most of all, I thank Seth Cooper for without whom this would not be possible.

This work was supported NSF and AFOSR.



University of Washington

**Abstract**

Decentralized Estimation for Stochastic Multi-Agent Systems

Fayette W. Shaw

Chair of the Supervisory Committee:  
Associate Professor Eric Klavins  
Electrical Engineering

The work here addresses estimation and control of the state of a multi-agent system. Three algorithms are introduced in the framework of distributed scalar and graph estimation. The first is Input-Based Consensus (IBC), where agents update their estimates using discrete and continuous state information. For IBC, we prove the convergence and stability, examine robustness to dropped messages, and demonstrate the algorithm on two robotic testbeds. The second is Feed-Forward Consensus (FFC), where agents include relative state change when updating their estimates. For FFC, we prove the convergence and stability of the algorithm, and simulate it in the context of a fixed-topology robotic network. Lastly, we present graph consensus, where agents estimate a graph rather than a scalar. We present a model of the system, simulate it, and demonstrate the algorithm on a system of wearable devices built for this purpose.



## TABLE OF CONTENTS

	Page
List of Figures . . . . .	iii
Chapter 1: Introduction . . . . .	1
Chapter 2: Research Settings . . . . .	5
2.1 Programmable Parts Testbed (PPT) . . . . .	5
2.2 SwarmBot Testbed . . . . .	7
2.3 Grouper Testbed . . . . .	8
Chapter 3: Related Work . . . . .	12
3.1 Consensus and Coordination . . . . .	12
3.2 Distributed Estimation . . . . .	13
3.3 Decentralized Control . . . . .	14
3.4 Sensor Networks . . . . .	15
3.5 Ubiquitous Computing . . . . .	16
Chapter 4: Mathematical Preliminaries . . . . .	18
4.1 Notation . . . . .	18
4.2 Consensus and Graph Theory . . . . .	19
4.3 Stochastic Processes . . . . .	20
Chapter 5: Input-Based Consensus . . . . .	25
5.1 Problem Statement . . . . .	25
5.2 Algorithm . . . . .	26
5.3 Model . . . . .	27
5.4 Analysis . . . . .	27
5.5 Distributed Control of Stoichiometry . . . . .	35
5.6 Estimation and Control . . . . .	37
5.7 Robustness to Dropped Messages . . . . .	38
5.8 Extensions for Estimation . . . . .	46

5.9 Discussion . . . . .	48
Chapter 6: Feed-Forward Consensus . . . . .	52
6.1 Introduction . . . . .	52
6.2 Problem Statement . . . . .	53
6.3 Algorithm . . . . .	53
6.4 Model . . . . .	54
6.5 First Moment Dynamics . . . . .	54
6.6 Analysis . . . . .	55
6.7 Simulation Results . . . . .	62
6.8 Application: Stochastic Factory Floor . . . . .	63
6.9 Discussion . . . . .	67
Chapter 7: Graph Consensus with Wearable Devices . . . . .	69
7.1 Introduction . . . . .	69
7.2 Problem Statement . . . . .	70
7.3 Algorithms . . . . .	73
7.4 Model . . . . .	77
7.5 Simulation . . . . .	78
7.6 System Characterization . . . . .	78
7.7 Experiments . . . . .	81
7.8 Discussion . . . . .	86
Chapter 8: Conclusion . . . . .	88
Appendix A: Grouper Design Details . . . . .	92
A.1 Summary . . . . .	92
A.2 Components . . . . .	92
A.3 Circuit diagram . . . . .	93
A.4 XBee . . . . .	93
A.5 Sample Code . . . . .	95
A.6 Resources . . . . .	99
Bibliography . . . . .	100

## LIST OF FIGURES

Figure Number	Page
1.1 Two examples of decentralized behavior. . . . .	2
2.1 The Programmable Parts Testbed (PPT). . . . .	6
2.2 The SwarmBot testbed with individual robot and Swarm. . . . .	7
2.3 Various modules of the Grouper system. . . . .	9
2.4 Various projects using the LilyPad toolkit built by different groups. . . . .	11
4.1 Definitions illustrated with the PPT. . . . .	18
4.2 Example SHS. . . . .	21
5.1 SHS model of estimation problem from the point of view of robot $i$ . . . . .	27
5.2 Comparison of simulated data with varying consensus parameter . . . . .	33
5.3 Experimental setup for the Programmable Parts Testbed. . . . .	34
5.4 State switching controller from the point of view of robot $i$ . . . . .	36
5.5 Birth-death process. . . . .	36
5.6 Estimator and controller combined. . . . .	37
5.7 Controller with estimator. . . . .	38
5.8 Analytical solutions plotted with experimental data for LAC and IBC. . . . .	42
5.9 LAC compared to IBC numerically for dropped messages. . . . .	43
5.10 Simulated data for LAC and IBC. . . . .	43
5.11 Experimental data comparison for LAC and IBC. . . . .	44
5.12 A SwarmBot experiment. . . . .	45
5.13 Estimation for multiple species. . . . .	49
5.14 IBC for fixed topology graph. . . . .	50
6.1 FFC modeled for robot $i$ . . . . .	54
6.2 State space equivalence classes called levels. . . . .	59
6.3 Sample path for estimation and control events. . . . .	61
6.4 Demonstration of the FFC estimator and controller working together. . . . .	62
6.5 Normalized histogram data for FFC at various times. . . . .	63
6.6 Stochastic Factory Floor (SFF) testbed. . . . .	64



6.7	Array of robotic tiles. . . . .	65
6.8	Simulations of estimation on fixed topology tile space. . . . .	66
6.9	Local estimation for 12 tile elements. . . . .	67
7.1	Centralized cluster-about-the-leader behavior. . . . .	71
7.2	Centralized <i>global alert</i> behavior. . . . .	72
7.3	Decentralized <i>connected graph</i> behavior. . . . .	73
7.4	Model for edge $i$ . . . . .	77
7.5	Five nodes changing topology, graph trajectory for Grouper simulation. . .	78
7.6	Connectivity shown for simulation with five nodes changing topology. . . .	79
7.7	RSSI and message frequency for an ideal environment. . . . .	80
7.8	RSSI and message frequency for a non-ideal environment. . . . .	80
7.9	RSSI and message frequency for an indoor environment. . . . .	81
7.10	Summary of campus experiments. . . . .	83
7.11	Experimental setup. . . . .	84
7.12	Data for five experiments with alerts and events classified. . . . .	85
7.13	Connectivity estimated by each node, experiment 4. . . . .	85
7.14	Graph estimates for each node, experiment 4 at 45 s. . . . .	86
A.1	Early prototype with components sewn. . . . .	92
A.2	Circuit schematic for follower module. . . . .	93

## Chapter 1

### INTRODUCTION

The promise of robotics is to create agents to perform dirty, dull, and dangerous tasks. Along these lines, robots have already been used for search and rescue [28, 29, 101], surveillance [88, 89], and warehouse order fulfillment [100, 128]. The potential for robots is magnified when imagining robotic cooperation. The benefits of cooperation can be seen in large-scale, redundant, and robust biological systems, such as ants and bees. They appear to behave in a decentralized, asynchronous manner yet manage to reach a common goal.

In a colony of ants, each ant chooses a task in response to cues from the environment and other ants [13]; an example of this is shown in Fig. 1.1(a) [32]. At any point, an ant has the choice to pick up, carry, or drop a particle. When an ant encounters an accumulation of particles it chooses to drop its payload, thereby increasing the aggregate and causing the clustering. A large group of people can make decisions in a decentralized way to form a pattern. In Fig. 1.1(b), groups of people decide where to stand on the Washington Mall for Obama's 2008 inauguration. The people assign themselves to locations based on the positions of Jumbotron screens for the task of finding the best view. Agents in both systems lack a centralized coordinator and act independently based on their environment and neighbors.

Biological systems have long been an inspiration for multi-robot systems [94, 67, 123, 33, 68], particularly for large numbers of robots. Swarms of ants and bees perform robustly in the face of uncertainty and environmental changes. Although a swarm of insects is robust and adaptive, each individual is simple. The capabilities of the agents are local and reactive; agents do not appear to store much information or follow a high level plan. Instead, complex and robust global behavior seem to arise from the composition of simple agents.

Robustness requires that systems possess several desirable properties. Ideally, multi-

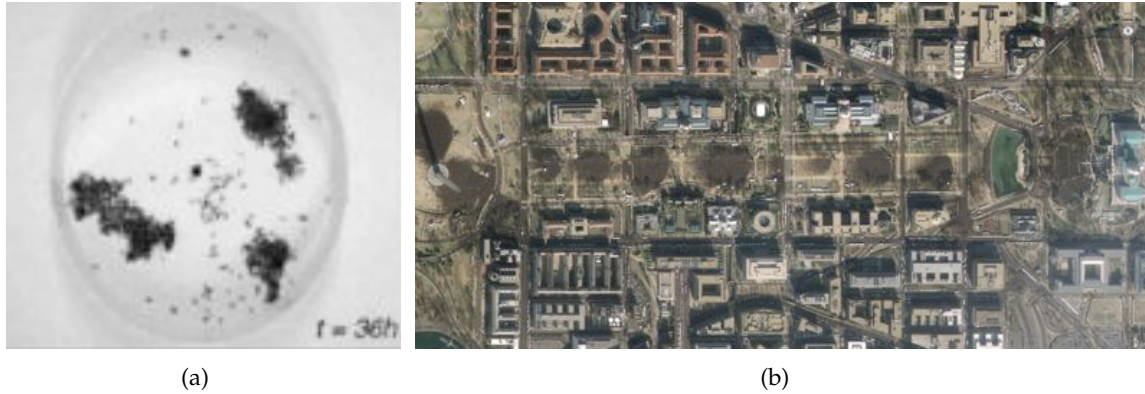


Figure 1.1: (a) Clustering ants in a Petri dish [32]. (b) Crowds gathered to witness Barack Obama's inauguration, January 20, 2009. Image from GeoEye satellite [39].

robot systems must be responsive to unknown environments with disturbances including robot failures, communication drops, changing populations, and sensor errors. However, as Veloso and Nardi point out, “the techniques for coordination and cooperation in [Multi-Agent Systems] are often not well suited to deal with the uncertainty and model incompleteness that are typical of robotics” [120]. Much work has been done since their article, but there is still a long way from multi-robot systems operating pervasively. We aim to address problems in cooperation to make large multi-agent systems more realizable.

One of the problems with any number of agents is agreement. To coordinate, agents must agree upon the state of the system. In our systems, agents come to agreement on their estimates of the global state using local interactions. They then use the estimate to drive the system to a desired state. The contribution of this thesis includes two classes of decentralized estimators for stochastic multi-agent systems. Algorithms were derived analytically, simulated, and demonstrated on three testbeds. The goals of this work are described as follows:

- Design algorithms for distributed control and estimation for multi-agent systems.
- Prove convergence and stability of these algorithms.
- Analyze robustness of algorithms.

- Demonstrate estimation and control algorithms on hardware testbeds.

We designed two algorithms, Input-Based Consensus (IBC) [107] and Feed-Forward Consensus (FFC) [108], both of which estimate a scalar quantity called the population fraction. The population fraction is the fraction of robots in a particular state. This estimate then informs a simple stochastic controller that switches the state of the robot, so that the robot can reassign its own state based on a desired population fraction.

*Input-Based Consensus* (IBC), described in Ch. 5, is a scalar estimation algorithm that uses the initial states of the robots for estimate updates. For this algorithm, we developed a mathematical model, proved convergence, and performed simulation and hardware experiments [107]. IBC is robust to arbitrary initial conditions and faulty communication; it converges with a finite variance under the requirement of well-mixed interactions and a constant number of agents. We experimentally derived a sensitivity function for the estimator variance as a function of the percentage of dropped messages [106]. We demonstrate that IBC with the controller converges in simulation. IBC is demonstrated two hardware platforms, the Programmable Parts Testbed and SwarmBot Testbed, both introduced in Ch. 2.

*Feed-Forward Consensus* (FFC), described in Ch. 6, is a scalar estimation algorithm that updates estimates via averaging and during a state change. We developed a mathematical model for agreement that converges with zero variance, proved convergence for the estimator and controller, and performed simulation. FFC is robust to an arbitrary number of agents and converges with zero variance under the requirement of perfect communication. We apply it in a fixed-topology condition in simulation. Ch. 6 summarizes the results of Shaw *et al.*[108].

The third contribution, detailed in Ch. 7, involves distributed estimation and agreement for the connectivity of a graph. We developed centralized and decentralized models but focused on the decentralized model for simulation and testing. We built Grouper, a hardware testbed used to estimate the proximity of a group of people to each other. Grouper alerts the group if it is too spread out. The hardware was characterized in various environments and tested outdoors to obtain real-world results.

Our goal is to build general framework for estimation of scalar and graph quantities in stochastic systems. We present our algorithms with proofs of convergence that are confirmed by simulation and a hardware realization with the hope of later extending our work to large-scale multi-agent systems.

## Chapter 2

### RESEARCH SETTINGS

We developed algorithms that are applicable to a wide class of systems and demonstrated them on three multi-agent testbeds. The first is the Programmable Parts Testbed and it consists of modular robotic components that self-assemble. The second testbed is a set of mobile robots called SwarmBots, which demonstrates a variety of multi-robot distributed algorithms including sorting, dispersing, and clustering. The third testbed, called Grouper, is a set of wearable devices used in group coordination. We present the testbeds here along with related hardware.

#### ***2.1 Programmable Parts Testbed (PPT)***

The Programmable Parts Testbed (PPT) [63] was built primarily by Nils Napp [83]. The PPT consists of triangular robots that are propelled by an external system of randomly actuated airjets and is shown in Fig. 2.1. The PPT serves as an approximation of chemical reactions on which to study a rich set of problems such as concurrency, non-determinism, and bottom-up self-assembly. The testbed simulates a test tube where components are so small that they are subjected to thermodynamic mixing. We limit ourselves from manipulating them individually. Instead, we design a common set of local rules for agents to produce a desired global behavior. The PPT was the inspiration for the scalar estimation algorithms presented in Ch. 5 and Ch. 6. We imagine components so small and so numerous that it is only possible to estimate the fraction of components in a state.

Each chassis (Fig. 2.1(b)) contains a microcontroller, three controllable magnetic latching mechanisms, three infrared transceivers, LEDs, and associated circuitry. When two robots collide, they temporarily join together using the embedded magnets in their chassis and exchange information via an infrared communication channel. Collision is the only mechanism for information exchange and occurs stochastically. Additionally, the

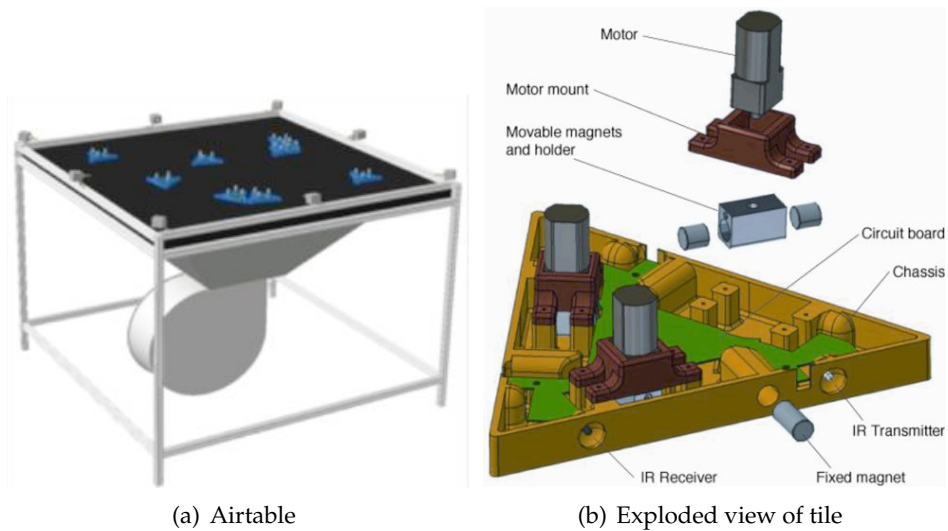


Figure 2.1: The Programmable Parts Testbed (PPT). (a) A schematic of the testbed. The parts float on an air table and are randomly stirred by air jets to induce collisions. (b) A diagram of a programmable part highlighting the latching mechanism.

robots can detach from one another by rotating motors that are attached to magnets. Thus, the robots can selectively reject undesirable configurations. We collect data using the LEDs on the robots from an overhead camera (LEDs not shown).

The PPT is part of a class of multi-robot systems of reconfigurable robots that do not have fixed morphology. The vision for modular reconfigurable robots is having a large-scale system with many inexpensive and robust components that can adapt to different scenarios; however, the state-of-the-art modules are few, expensive, and fragile. Yim *et al.* review current modular systems including the PPT [133].

The PPT is a stochastically reconfiguring system and we model it as a Markov process [64, 63]. Other testbeds are also described as Markov processes including passive parts [50], non-locomoting but actuated parts [10] and mobile robots [77]. In these testbeds, robots move randomly and generate random communication graphs.

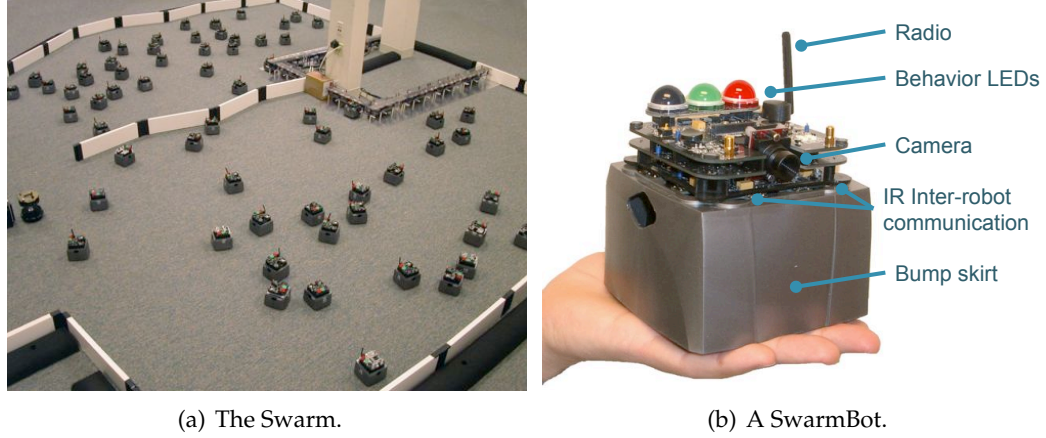


Figure 2.2: The SwarmBot testbed with individual robot and Swarm. The SwarmBot testbed is a mobile robot platform of up to 100 robots used to examine distributed multi-agent algorithms.

## 2.2 *SwarmBot Testbed*

The SwarmBot testbed (Fig. 2.2) is a multi-robot system built by James McLurkin [79]. Each robot has a processor, unique ID, four IR transceivers to communicate with other robots, and various sensors, such as bump skirt for obstacle avoidance, as shown in Figure 2.2(b). LEDs are mounted on top of the robots to indicate state for the user. There are on the order of 100 agents in existence, but we designed experiments for about 20 robots, due to spatial limitations for collecting data.

Algorithms already implemented on the SwarmBot testbed include communication routing, navigation, dynamic task assignment, and boundary detection [80, 79]. We use the SwarmBot testbed to demonstrate Input-Based Consensus, one of our scalar consensus algorithms. Specifically, we demonstrate IBC with dropped messages to examine the robustness of the algorithm (Ch. 5). We model the system as a set of well-mixed particles and we enforce this behavior by implementing a “bounce and reflect” behavior on the robots. Robots drive straight until contacting an obstacle and “reflect” back into the environment.

Few robotic testbeds exist where the agents are so numerous. The Centibots [66] aims to advance coordination with deployment of 100 robots for search and rescue and surveil-



lance. Kiva Systems [128] employs one thousand robots in two warehouses for order fulfillment. Increased numbers of agents present challenges for communication and coordination.

### **2.3 *Grouper Testbed***

We built the Grouper Testbed to explore graph consensus (Ch. 7). The purpose of this testbed is to estimate the proximity of a group and alert group members when the group is not together. Grouper is a wireless testbed for group coordination and consists of wearable modules.

Grouper consists of modules made primarily from the LilyPad Arduino microprocessor and related components [21]. The LilyPad Arduino is a sewable and washable microprocessor that can be connected to other components via conductive thread. This toolkit has allowed for simple integration of electronics into clothing [23, 22] and was chosen for size, available components, and ease of development, particularly for the wearable modules.

There are three types of modules: wearable, logger, and leader. Each module consists of an Arduino microprocessor [117], XBee Series 1 radio [31], lithium polymer battery, LilyPad power board, and electronics for sensory cues. The wearable Grouper module has been designed to be worn on the wrist and includes the Q board on its face to provide sensory cues (Fig. 2.3a). The Q board consists of two LEDs, green and red, to indicate if a user is within a desirable range, a speaker, and a vibrating motor. Together, these electronics alert a user when he or she has wandered too far from the group. The inside of the wrist module contains the LilyPad Arduino and battery components (Fig. 2.3b).

We designed the grouper leader module particularly for group coordination with a leader. The leader module features an LCD screen to list group members' names (Fig. 2.3c). The LCD indicates when a user is out of range or has been missing from the group for an extended period of time. The leader includes the Arduino Duemilanove [117], scroll buttons, and mode switch. The scroll button is used to navigate the LCD entries and the mode switch is used to change between group behaviors.

The logger was designed as a compact device for recording data (Fig. 2.3d). The logger

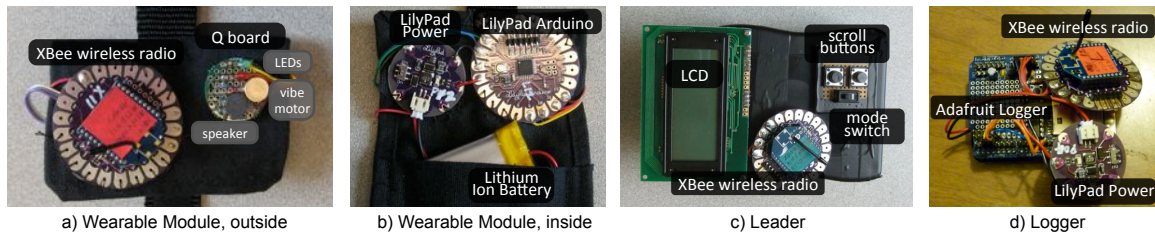


Figure 2.3: Various modules of the Grouper system. a) Wearable module. Designed to be worn on the wrist, the outside provides sensory cues. b) Wearable module, inside. The inside includes the microprocessor and battery. c) Leader module. An LCD to displays the names of group members. d) Logger Module. Used for data collection.

consists of an SD card reader, Arduino Deumilenove, XBee, buzzer, and various electronics. It acts as a node in the experiments. The drawback of using a single centralized logger is that it records data only whenever it receives a message, not when a user wanders away.

We designed Grouper so that users need not continually consult the modules. A user is alerted via a vibrating motor if he or she is too far. Note that Grouper works only proximally rather than directionally. An alert does not give the direction for a user to look; it alerts the user to notice surroundings and look for the group. This system is not a replacement for keeping track of members in a group; it simply augments a user's ability to track the group.

### 2.3.1 Person Tracking

Products exist for tracking people and most are geared toward tracking children. Low tech approaches include temporary tattoos [116], ID bracelets [121], reflective vests [102], and leashes. The problem with these low tech approaches is that they are reactive rather than preventive; they only work if a child is found by a friendly stranger who contacts the caregiver. An example of a reactive high tech solution is EmFinders EmSeeQ [59], which uses the cellular network to call 9-1-1 and is triggered by the caregiver. In contrast, active high tech solutions include Brickhouse's Lok8u GPS Child Locator [19] and Amber Alert GPS [1], which alert a caregiver if a person has wandered out of a safe zone. Disadvantages of current high tech solutions include consumer overhead and sensing limitations.

EmSeeQ, Lok8u, Amber Alert GPS, all require subscription plans for operation and GPS solutions do not work indoors.

Patients with dementia are also prone to wandering. Approaches to preventing wandering include those that address the actual events such as physical restraints, barriers, and electronic devices, and those that lessen wandering behavior such as drugs, exercise, and sensory therapy [7, 82, 99]. While effective, there are some ethical concerns for using tracking devices [54].

Most research platforms that track people are composed of intelligent badges that locate users [124] or track user interactions [62, 122]. Often, they have been implemented to infer social networks in office environments or to infer interactions with objects in educational environments. Hwang *et al.* built a system to observe exploratory behaviors in kindergarten field trips [56] and Park *et al.* designed smart badges for a “Smart Kindergarten” [87].

Current commercial and research approaches do not aim to keep people in a group; they locate wearers relative to a caregiver or an environment. In our work we not only observe a network but also apply sensory cues to guide the system to a desired state. Many solutions require attention from the caregiver rather than lessening cognitive load. Grouper alerts users in an attempt to reduce the attention required to keep a group together.

### 2.3.2 *Related Hardware*

The advent of the LilyPad Arduino has made the development of wearable devices much easier [22, 23]. Many projects have been developed using the LilyPad toolkit. Figure 2.4 shows a small sample of projects using LilyPad hardware [24]. Figure 2.4(a) shows the Reading Glove built by Karen and Josh Tenenbaum of Simon Frasier University. The glove provides interaction for story-telling when a reader picks up an object [115]. Figure 2.4(b) shows a dress, which is part of Spin on a Waltz. It, along with a suit jacket, were built by Ambreen Hussain for Viennese waltzers to control music interactively [55]. Lastly, Fig. 2.4(c) shows the LED Turn Signal Jacket built by Leah Buechley [20] of MIT Media Lab.

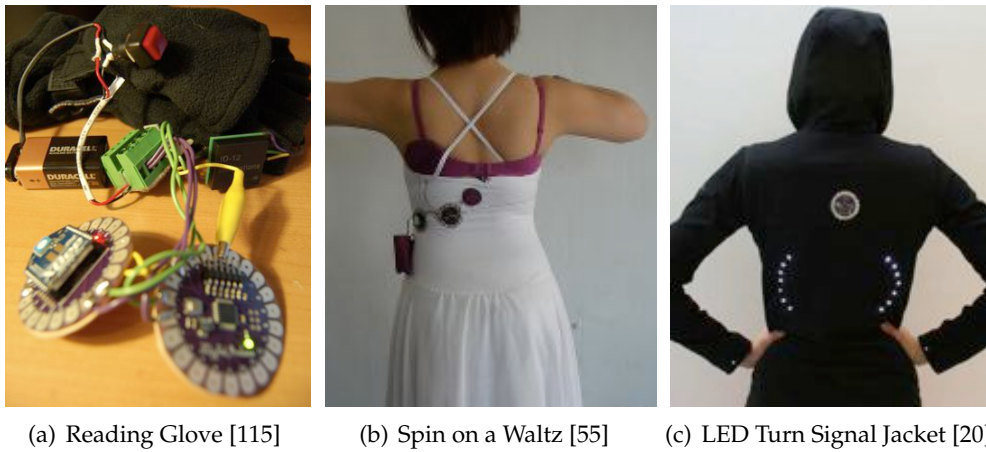


Figure 2.4: Various projects using the LilyPad toolkit built by different groups. a) The Reading Glove was built by Karen and Josh Tenanbaum. b) Spin on a Waltz is a dress and suit built by Ambreen Hussain. c) The LED Turn Signal Jacket was built by Leah Buechley.

The jacket lights up when a user presses a button and indicate the direction of turning.

## Chapter 3

### RELATED WORK

The work in this thesis draws from many areas with a focus on consensus and coordination. For both scalar and graph estimation, we aim to estimate an unobservable quantity using local communication and agree upon that quantity. We discuss the notions of distributed estimation and distributed control and how they are used in this thesis. Lastly, we discuss sensor networks and ubiquitous computing, which are most related to graph consensus.

#### 3.1 Consensus and Coordination

Consensus literature includes multi-agent phenomena from flocking birds to swarming robots [86, 25, 97, 81]. The end goal is for all agents is to reach agreement using local interaction. Olfati-Saber *et al.* [86] and Mesbahi and Egerstedt [81] present a comprehensive review of recent literature.

Our work is an extension of Linear Average Consensus (LAC), where agents come to agreement by averaging state. In fact LAC is a special case for our two algorithms IBC and FFC. LAC is governed by *Laplacian dynamics*, which can be used to describe the evolution of the agreement values and is detailed in Ch. 4.

We model our systems as randomly interacting particles whose interactions yield random graphs. Hatano and Mesbahi [47] consider consensus over random graphs and examine the convergence of the first and second moment. We use a similar approach and provide an alternate proof of convergence. Jadbabaie *et al.* [58] consider a switching graph and prove convergence by examining the the union of graphs. Our work involves tracking of a quantity that is the average of the agents' discrete states. Spanos *et al.* [111] examines tracking in a network that splits and merges; subnetworks of agents come to consensus on their initial conditions. Tsitsiklis [119] addresses consensus with an exogenous mea-

surement. Input consensus is also examined in Mesbahi and Egerstedt's text [81]. Systems with input and output agreement generate standard linear time invariant systems. Zhu and Martínez [136] present a discrete-time dynamic average consensus problem and track a time-varying function.

Much of the consensus literature is framed in the context of graph theory [43], introduced in Ch. 4. Graph representations of multi-agent systems are widely used for modeling information flow, communication, and sensor topology. Agents often use averaging to reach agreement on the state of the system. In our work, we form graph estimates by weighing information unequally based on perspective. Nodes reach agreement on graph connectivity and on the graph itself.

### **3.2 *Distributed Estimation***

The field of distributed systems has its foundations in computer science [74] and economics; the subject has broadened to include many applications and the terms. Thus, distributed and decentralized estimation have evolved many meanings in the literature. Often the terms refer to sensing and sensor fusion [111, 131], where agents may independently perform Kalman filtering to fuse measurements [85]. Alternatively, it refers to estimation of state variables in noisy processes using maximum likelihood state estimation [118].

In this thesis, distributed estimation refers to agents in a network, which each have a measurement of the global state that is updated by an algorithm that uses local information. The agents estimate the state for a process that evolves deterministically, according to stochastically chosen partners. Specifically, distributed estimation refers to agents in a network each having a measurement of the population fraction; the estimate gets updated through local interactions. The work in this thesis is most closely related to linear average consensus [129, 119] and it behaves like gossip networks [18].

We developed algorithms that incorporate information from distributed estimation and control our using stochastic self-assignment. Input-Based Consensus and Feed-Forward Consensus are methods for performing distributed estimation. An agent uses information

gathered by local interactions to perform decentralized control and drive the global system to a desired state.

### 3.3 *Decentralized Control*

In the context of multi-robot systems, we consider decentralized control as task assignment instead of navigation, mapping, or position control as in other applications. Task assignment refers to *which* robots should do a particular task and task allocation refers to deciding *how many* robots should do a particular task. On a small scale, groups of agents can be divided into teams and controlled centrally. On a larger scale, we allow the agents to assign themselves to tasks based on local measurements and switch these tasks stochastically. We believe that adding randomness to the system results in robustness. Our work provides a framework for task assignment in large stochastic systems and environments with unknown or unforeseen disturbances.

#### 3.3.1 *Task Assignment*

Again, we are inspired by biological systems where ants and bees use *stigmergy* and *quorum sensing* to address task assignment and allocation. Stigmergy is a mechanism for indirect communication where an agent alters the environment, such as leaving a pheromone, to affect the behavior of other agents in a system. *Quorum sensing*, is a mechanism of agreement in organisms from bacteria [14] to ants [92]. Using these two mechanisms, members of swarms are recruited to perform tasks. When necessary, agents switch tasks stochastically. This kind of behavior is observed in how bees choose between foraging for nectar or honey [105] or how ants choose a new home [92]. Task assignment and allocation observed in nature has provided insight to designing algorithms for robots.

In the survey paper by Baghaei *et al.* [11], the authors describe different architectures for robot cooperation and categorize the types of tasks that can be addressed by multi-agent systems. One of the framework for robotic cooperation is a team, such as in robot soccer, where robots play in positions and switch roles throughout a game [113]. In small groups, agents maintain a model of other agents and the system as a whole. This is unfeasible for

larger groups; specifically, since communication does not scale.

### 3.3.2 Task Allocation

Task allocation is divided into three categories: behavior-based, auction-based, and dynamic switching. Behavior-based algorithms are ones where agents have minimal state and sensor measurements are coupled to robot control. Behavior-based task allocation have been extensively studied by Mataric *et al.* [41, 76, 127] and Parker [73]. In Parker's work [73], robots are governed by motivation for every set of behaviors. Motivation is affected by sensory feedback, inter-robot communication, inhibitory feedback from other active behaviors, and internal motivations called robot impatience and robot acquiescence.

Auction-based algorithms define a utility function for a specific robot to complete a task. These allocations are optimized using a greedy algorithm and market strategies [40, 137, 30] and leverages the formalism of economics. Agents communicate to an auctioneer (decentralized and centralized algorithms exist) and the tasks are assigned to bidders. These algorithms require complicated arbitration and thus does not scale well with the number of robots.

The work in this thesis is closely related to dynamic task switching. Here, agents are initially assigned states, tasks, or roles, and switch according to sensor measurements. Several works examine dynamic switching as agents are deployed to various sites, simulating ant house hunting in nature [46, 17]. McLurkin *et al.* [80] describes several task distributed assignment algorithms including, random assignment, all-to-all communication of full-state information, sequential assignment, and tree assignment. However, these works do not examine the robustness of their algorithms.

## 3.4 Sensor Networks

Our testbed Grouper (introduced in Ch. 2) is an example of a wireless sensor network [5, 93]. Puccinelli *et al.* [93] present a survey paper of wireless sensor networks (WSNs) and address applications, existing systems, hardware design, and desirable characteristics. Wireless sensor networks lend themselves to distributed data collection and are often de-



ployed for long-term environment monitoring [75, 49]. In many wireless sensor networks, nodes are stationary and take data about the environment, sending data as infrequently as possible to conserve power.

In contrast, mobile sensor networks are generally mobile and may send data frequently; energy becomes a challenge. Grouper is an example of a mobile network that broadcasts frequently to maintain graph connectivity. Grouper is mobile ad hoc network (MANET) [103, 78]: a self-configuring wireless network that does not require any infrastructure. Zebra Net [60] is a tracking system for zebras and is comprised of a dynamic sensor network and a mobile base station. Nodes send information about their location peer-to-peer so that researchers can download data by encountering a few agents instead of the whole herd. Researchers modeling cow movements [104] developed a data-driven model to derive herd dynamics.

In addition to sending messages, Grouper uses a radio to sense proximity using Received Signal Strength Indicator (RSSI) of packets. This is not an uncommon usage for wireless radios [16, 2, 8] and RSSI measurements can be used for localization [6, 9, 110]. However, it is still debated whether RSSI is a good measure of proximity [112, 16, 135]. For instance, Benkić *et al.* characterize RSSI indoors and conclude it to be a poor distance estimator. In Sec. 7.6, we characterize RSSI as a function of distance for several environments.

### 3.5 Ubiquitous Computing

Ubiquitous computing, or ubicomp, is a model of interaction where there are many computers for a single user. Mark Weiser imagined a world where computers are integrated seamlessly to assist people: computers everywhere that provided a sense of order instead of sensory overload [125]. Abowd *et al.* [3] examine a brief history of ubicomp in the context of natural interfaces, context-aware computing, and automated capture and access. The concerns of ubicomp with the respect of scalability is similar to those in distributed robotics, especially modular robotics. Both require low-cost hardware; ubicomp aims to put computers everywhere and modular robotics aims to reconfigure robots into anything. In Ubicomp, computers are placed everywhere by augmenting the environment

or instrumenting users. One way to instrument users is to create wearable devices.

### *3.5.1 Wearable Computing*

Many researchers are finding ways of incorporating sensors and microprocessors into novel applications [84, 71]. Biosensors have been integrated into clothing to aid healthcare and military applications [132]. Much of the work in wearable computing have been used for identifying patterns in social interactions. For example, there are various intelligent badges that track user interactions [62, 124, 122, 87]. The contribution of this thesis is to observe a network and also aid in controlling the group.

## Chapter 4

### MATHEMATICAL PRELIMINARIES

#### 4.1 Notation

First, consider a set of  $n$  robots and each robot  $i$  has a discrete internal state  $q_i(t) \in \{0, 1\}$ . Define the *population fraction* as

$$x(t) \triangleq \frac{1}{n} \sum_{i=1}^n q_i(t). \quad (4.1)$$

Each robot also maintains an *estimate*  $\hat{x}_i(t) \in [0, 1]$  of the population fraction  $x(t)$ . It is assumed that each robot knows the value of  $n$ . The vector  $\mathbf{q} = (q_1 \dots q_n)^T$  is the vector of internal states and  $\hat{\mathbf{x}} = (\hat{x}_1 \dots \hat{x}_n)^T$  is the vector of estimates. For simplicity, we often omit the explicit dependency of  $x$ ,  $\hat{x}$ , and  $\mathbf{q}$ , on  $t$ . In the sequel, the symbol  $\langle \cdot \rangle$  denotes expected value and  $\mathbf{1} = (1, \dots, 1)^T$ . Table 4.3.1 is included at the end of the chapter to introduce notation used in this thesis.

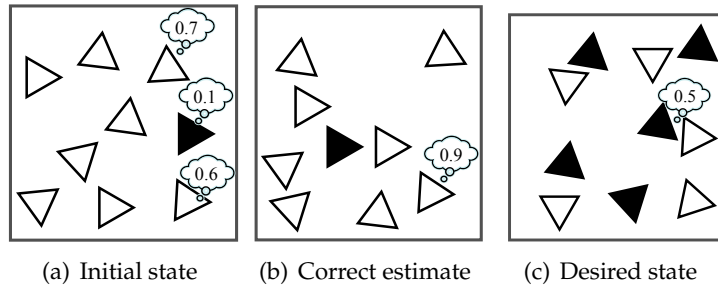


Figure 4.1: Definitions illustrated with the PPT. In this example, the reference population fraction is 0.5. (a) Initial condition with incorrect estimates and states. (b) The correct estimate of the population fraction has been reached and the robots agree on this estimate. (c) Robots have assigned themselves to the desired population fraction.

For scalar estimation, each robot is given a state of  $q = 0$  or  $q = 1$  and switch between them. In Fig. 4.1, the states are indicated by shading (light or dark). The population fraction is the proportion of light parts to the total number in the system. Figure 4.1 illustrates

the parts initially (a) with arbitrary estimates (thought bubbles) and states and (c) arriving at the desired reference. While conceived for the PPT, these terms can be defined similarly for the SwarmBot testbeds.

## 4.2 Consensus and Graph Theory

Consensus in multi-agent systems is used to describe phenomena from flocking birds to robots [86, 95, 25, 36]. A common task is to compute the average of the agents' initial states. The end goal for measurements to agree across agents. That is,

$$x_1 = x_2 = \dots = x_n$$

at equilibrium. Much of the consensus literature is framed in the context of graph theory [43]. Graph representations of multi-agent systems are widely used for modeling information, communication, and sensor topology. A graph  $\mathcal{G}$  consists of a pair  $(\mathcal{V}, \mathcal{E})$ , of vertices and edges. Here, the adjacency matrix  $A$  is defined for an undirected graph where entries  $(i, j)$  and  $(j, i)$  are denoted 1 if vertices  $i$  and  $j$  are connected. The degree matrix  $D$  counts the number of edges at each vertex. The Laplacian matrix is defined as  $L = D - A$ .

To reach agreement in a distributed manner, the approach taken frequently in the literature is for each agent  $i$  to maintain a state variable,  $x_i$ , which is updated when interacting with neighboring agents

$$\dot{x}_i = \sum_{j \in \mathcal{N}_i(t)} f_i(x_i, x_j). \quad (4.2)$$

A node  $j$  where  $j \in \mathcal{N}_i(t)$  is in the neighborhood of agent  $i$ ; agent  $i$  has access to state  $x_j$ . Note that the neighbor set  $\mathcal{N}_i$  may vary with time. For  $f_k(x_i, x_j) = x_j - x_i$ , the dynamics of the agents can be collectively written as

$$\dot{x} = -Lx, \quad (4.3)$$

where  $x$  is a vector of states. The system is said to obey *Laplacian dynamics*.

When the output of the update rule,  $f_i$ , is a linear combination of the input, the resulting protocol is referred to as *Linear Average Consensus* (LAC). The desired agreement value can be any convex combination of the initial states. Using LAC, agents compute the average

of the initial global state. LAC has is constrained that the sum of initial conditions,  $\sum_i y_i^0$ , must be constant for all time, where  $y_i^0$  is the initial value for robot  $i$ . If messages are ever lost, the global sum changes, so using LAC will not result in convergence to the correct value.

### 4.3 Stochastic Processes

The PPT [63] is designed to implement a chemical reaction, so it is natural to model it as a stochastic process. We also model the SwarmBot testbed as a stochastic process and design their behavior to induce random interactions. For the scalar estimation algorithms, we assume that the robots are *well-mixed*, any pair of robots interact uniformly on  $[0, dt]$ . Later, we add control to switch robots between states at a stochastic rate.

#### 4.3.1 Stochastic Hybrid Systems

We approximate each of our scalar estimation systems as a continuous-time Markov process [26, 106]. Each agent performs deterministic state updates at random times and can be described using Stochastic Hybrid System (SHS) formalism [48]. A SHS consists of a set of discrete and continuous states governed by a state-dependent stochastic differential equation and reset map.

Formally, the state space is expressed as  $X \times Q$ , where  $X$  denotes the continuous states and  $Q$  denotes the discrete states. Each discrete state is governed by continuous dynamics. A SHS is defined by a stochastic differential equation, or called *flow*,

$$\dot{x} = f(q, x), f : X \times Q \rightarrow X$$

and discrete reset maps

$$(q, x) = \phi_l(q^-, x^-), \phi_l : X \times Q \rightarrow X \times Q$$

with associated rates  $\lambda_l(x, q) : X \times Q \rightarrow [0, \infty)$ . The reset map  $\phi$  determines the state to which the process jumps from state  $(q^-, x^-)$ . The rate  $\lambda_l(x, q)$  is a probability that the

process transitions during  $[t, t + \delta]$ .

**Example:**

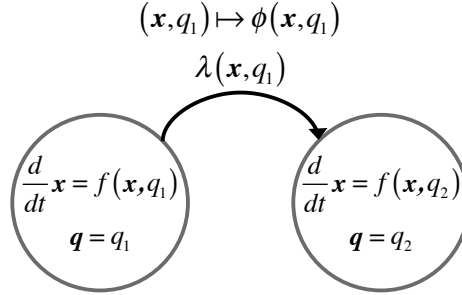


Figure 4.2: Example SHS. Each state is labeled by the discrete state and continuous dynamics. State transitions are labeled by the update  $\phi$  at rate  $\lambda$ .

Figure 4.2 is an example of a SHS. Each state is labeled by the discrete state and continuous dynamics:

$$\frac{d}{dt} \mathbf{x} = \begin{cases} f(\mathbf{x}, q_1) & \mathbf{q} = q_1 \\ f(\mathbf{x}, q_2) & \mathbf{q} = q_2, \end{cases} \quad (4.4)$$

which are state dependent. There is one state transition between  $q_1$  to  $q_2$ , which is labeled by the update

$$(\mathbf{x}, q_1) \mapsto \phi(\mathbf{x}, q_1), \lambda(\mathbf{x}, q_1)$$

at rate  $\lambda(\mathbf{x}, q_1)$ .

To reason about the dynamic behavior of a SHS, we use the following theorem about the extended generator  $\mathcal{L}$ , a partial differential equation describing the dynamics of the expected value of arbitrary test functions  $\psi(\mathbf{x}, q)$ . The extended generator  $\mathcal{L}$  is an operator on an arbitrary test function  $\psi(\mathbf{x}, q)$  given by

$$\mathcal{L}\psi(\mathbf{x}, q) = \frac{\partial \psi(\mathbf{x}, q)}{\partial \mathbf{x}} f(\mathbf{x}, q) + \frac{\partial \psi(\mathbf{x}, q)}{\partial t} + \sum_l \lambda_l(\mathbf{x}, q) (\psi(\phi_l(\mathbf{x}, q)) - \psi(\mathbf{x}, q)), \quad (4.5)$$

where  $\phi_l$  is the function for a particular change of state  $l$ , and  $\lambda_l$  is the associated rate. The complete formalism [48] includes noise, but we omit it here as it is not applicable to our

systems. Our systems have neither continuous flow ( $\dot{x} = 0$ ) nor time dependence, so we use the simplified extended generator  $\mathcal{L}$  with only discrete dynamics

$$\mathcal{L}\psi(x, q) = \sum_l \lambda_l(x, q) (\psi(\phi_l(x, q)) - \psi(x, q)), \quad (4.6)$$

We use this theorem below to reason about the dynamics of our probabilistic systems.

**Theorem 1** [48] *Let  $\psi : \mathbb{R}^n \rightarrow \mathbb{R}^m$  be an arbitrary function. Then*

$$\frac{d}{dt} \langle \psi(\mathbf{x}) \rangle = \langle \mathcal{L}\psi(\mathbf{x}) \rangle. \quad (4.7)$$

Theorem (4.7) is used to examine the dynamics of the expected value. The simplified extended generator (4.6) can be derived by taking the expected value of the master equation [61]. A master equation is a set of differential equations that describe the dynamics of a probability mass function. In our work, we derive moment dynamics using both the master equation [107] and the extended generator [108, 106].

For a process  $(Y, t)$ , the change in probability can be expressed as a master equation

$$\frac{d}{dt} P(y, t) = \int W(y|y') P(y', t) - W(y'|y) P(y, t) dy',$$

where  $W(y|y')$  is the probability that the system transitions to  $y$  in time  $t + dt$  given that state is  $y'$  at time  $t$ . The first term in the integral refers to the transitions going into the state and the second refers to the transitions leaving.

For a discrete-state Markov process that transitions from state  $l$  to  $m$ , the probability that the process is in state  $l$  is given by

$$\frac{dP_l}{dt} = \sum_l (W_{lm} P_m - W_{ml} P_l) \quad (4.8)$$

where  $W_{lm}$  is the transition intensity to state  $l$  from  $m$  and  $P_l$  is the probability of the process being in state  $l$ . Similar terms are named accordingly. Written as matrices, (4.8) can be

written as the dynamics of an arbitrary probability mass function are given by

$$\frac{d}{dt}\mathbf{p} = \mathbf{W}\mathbf{p}, \quad (4.9)$$

where  $\mathbf{p}$ , where  $\mathbf{p} = (p_1 p_2 \dots p_m)^T$ . Equation (4.8) is called the *master equation* and  $\mathbf{W}$  is the *infinitesimal generator* for a discrete Markov process. We can use the master equation (4.9) to calculate the change in expected value

$$\frac{d}{dt}\langle \mathbf{p} \rangle = \langle \mathbf{W}\mathbf{p} \rangle. \quad (4.10)$$

Note the similarity between (4.7) and (4.10).



$t, t^- / t^+$	time, time immediately before/after interaction
$n$	number of agents
$i, j$	agent indices
$\mathcal{N}_i$	neighborhood of agent $i$
$\langle \cdot \rangle$	expected value
$\mathbb{1}$	vector of 1s
$\binom{n}{2}$	$\frac{n(n-1)}{2}$
$\otimes$	Kronecker product
$I$	identity matrix
$A$	adjacency matrix
$D$	degree matrix
$L$	Laplacian matrix
$M(i, j)$	indices $(i, j)$ for matrix $M$
$vec(\cdot)$	vector representation of a matrix
$X$	continuous state space
$Q$	discrete state space
$f$	continuous function
$\mathcal{L}$	extended generator
$l, m$	reaction or state index
$\phi_l$	discrete update function for reaction $l$
$\lambda_l$	rate for associated for reaction $l$
$\psi(x, q)$	arbitrary test function
$x$	population fraction (global state)
$q_i$	agent discrete state
$\hat{x}_i$	agent estimate of population fraction
$\zeta$	consensus parameter
$k$	natural reaction rate, estimator rate
$K_i$	controller rate for agent $i$
$\gamma$	fraction of successful messages
$A_{ij}, B_{ij}$	weight matrices for interaction between agents $i, j$
$C_{iji}, D_{iji}$	weight matrices for interaction between agents $i, j$ ; agent $i$ drops a message
$e$	edge $e = (i, j, rssi_{ij}, age_{ij}) \in E$ ; data for edge between agents $i$ and $j$
$E_i(j, k)$	edge between agents $j$ and $k$ from the point of view of agent $i$

Table 4.1: Notation table.

## Chapter 5

### INPUT-BASED CONSENSUS

#### 5.1 Problem Statement

We consider a kind of consensus problem motivated by the self-assembly of simple robotic parts [63] into a larger structure. To maximize the yield of a target structure  $A$  it may be useful to balance the ratio of its subcomponents  $B$  and  $C$ . In this chapter, we capture the essence of this problem using a population of stochastically interacting robots. Each robot has a discrete state that is either zero or one and, by switching between these values, they can control the proportion of robots in each state. This proportion is called the *population fraction* and represents the global state of the system. Each robot estimates the population fraction so changing its discrete state to drive the population fraction to a *reference state*.

Consider a set of  $n$  robots where each robot  $i$  has a discrete internal state,  $q_i(t) \in \{0, 1\}$ . Recall that the *population fraction*  $x(t)$  in (4.1)  $x(t)$  the mean of discrete states  $q$ . Each robot also maintains an *estimate*  $\hat{x}_i(t) \in [0, 1]$  of  $x(t)$ . We assume that the robots are well-mixed; that is, the pair of agents that interacts at any time is uniformly distributed. From this well-mixed assumption, any pair of robots is equally likely to interact in the next  $dt$  seconds with probability  $k dt$ .

When two robots  $i$  and  $j$  interact, they update their estimates  $\hat{x}_i$  and  $\hat{x}_j$  according to function  $f$ . We present three problems:

- 1) The *estimation problem*: Define an estimate update  $(\hat{x}_i, \hat{x}_j) \mapsto f(\hat{x}_i, q_i, \hat{x}_j, q_j)$  so that  $\hat{x}_i(t)$  converges to the population fraction  $x(t)$ , with high probability as  $t \rightarrow \infty$ .
- 2) The *control problem*: Define a rate  $K_i(x, q)$  for each agent that governs switching between states  $q = 0$  and  $q = 1$ .
- 3) The *simultaneous estimator and controller problem*: Demonstrate that the solution to the control problem runs concurrently with the estimator. That is, use the estimate for switch-

ing rate  $K(\hat{x}, q)$  to drive the system to the reference.

In this chapter, we introduce our estimation algorithm referred to as Input-Based Consensus (IBC), prove the convergence for the estimator and controller separately, and demonstrate them together in simulation. Later, we also examine the robustness of IBC to dropped messages and extend it to multiple states.

## 5.2 Algorithm

We first consider the estimation problem: robots update their estimates for constant discrete states. Robots  $i$  and  $j$  interact at time  $t$  and update their estimates according to

$$\begin{aligned}\hat{x}_i(t^+) &= \zeta (a\hat{x}_i(t^-) + (1-a)\hat{x}_j(t^-)) + (1-\zeta) \left( \frac{1}{n}q_i + \frac{n-1}{n}q_j \right) \\ \hat{x}_j(t^+) &= \zeta ((1-a)\hat{x}_i(t^-) + a\hat{x}_j(t^-)) + (1-\zeta) \left( \frac{n-1}{n}q_i + \frac{1}{n}q_j \right) \\ \hat{x}_l(t^+) &= \hat{x}_l(t^-) \text{ for all } l \neq i, j.\end{aligned}\tag{5.1}$$

Here,  $a, \zeta \in (0, 1)$  are design parameters. The parameter  $a$  weighs the relative importance of estimates  $\hat{x}_i$  and  $\hat{x}_j$ . The parameter  $\zeta$  is the *consensus parameter* and weighs the relative importance of estimates and discrete states. We call  $\zeta$  the consensus parameter because our algorithm IBC reduced to Linear Average Consensus (LAC), introduced in Ch. 4 when  $\zeta = 1$ . The parameter  $\frac{1}{n}$  weighs a robot's contribution of discrete state to the rest of the population. It is assumed that each robot knows the value of  $n$ . The symbols  $t^-$  and  $t^+$  denote the times immediately before and after the interaction, respectively. The last line of the above update rule represents the fact that robots not participating in the interaction do not update their estimates.

We can write this update as matrices, for an example in which robots 1 and 2 interact within a system of three robots:

$$\begin{aligned}\hat{\mathbf{x}}(t^+) &= \zeta \begin{pmatrix} a & 1-a & 0 \\ 1-a & a & 0 \\ 0 & 0 & \frac{1}{\zeta} \end{pmatrix} \hat{\mathbf{x}}(t^-) + (1-\zeta) \begin{pmatrix} \frac{1}{n} & \frac{n-1}{n} & 0 \\ \frac{n-1}{n} & \frac{1}{n} & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{q}, \\ \text{or } \hat{\mathbf{x}}(t^+) &= \zeta \mathbf{A}_{12} \hat{\mathbf{x}}(t^-) + (1-\zeta) \mathbf{B}_{12} \mathbf{q},\end{aligned}\tag{5.2}$$

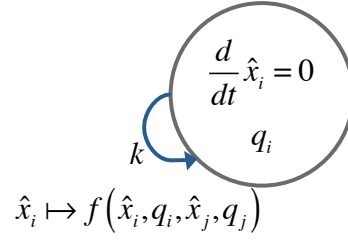


Figure 5.1: SHS model of estimation problem from the point of view of robot  $i$ . Robots  $i$  and  $j$  interact at rate  $k$ , as denoted by the blue arrow. Estimate  $\hat{x}_i$  is updated as a function of estimates  $\hat{x}_i, \hat{x}_j$  and constant discrete states  $q_i, q_j$ .

where  $A_{12}$  denotes the adjacency matrix for an interaction between robots 1 and 2. Generally, matrices  $A_{ij}$  and  $B_{ij}$  for robots  $i$  and  $j$  are defined as follows:

$$\begin{aligned}
 A_{ij}(i, i) &= A_{ij}(j, j) = a & B_{ij}(i, i) &= B_{ij}(j, j) = \frac{1}{n} \\
 A_{ij}(i, j) &= A_{ij}(j, i) = 1 - a & B_{ij}(i, j) &= B_{ij}(j, i) = \frac{n-1}{n}, \\
 A_{ij}(l, l) &= \frac{1}{\xi}
 \end{aligned} \tag{5.3}$$

where  $(i, j)$  indicate matrix indices and all remaining matrix entries are 0 and  $l \neq i, j$ . Each reaction corresponds with update matrices  $A_{ij}$  and  $B_{ij}$ .

### 5.3 Model

We model our system as a SHS, as introduced in Ch. 4, and show this formalism for a robot  $i$ . Recall that a SHS comprises of continuous and discrete dynamics. In our system, the estimate  $\hat{x}$  evolves only by discrete jumps. Thus, there is no continuous flow, so  $\frac{d}{dt} \hat{x} = 0$  for all agents  $i$ . At rate  $k$  robot  $i$  interacts with robot  $j$  and updates its estimate  $\hat{x}_i$  as a function of estimates  $\hat{x}_i, \hat{x}_j$  and constant discrete states  $q_i, q_j$ .

### 5.4 Analysis

#### 5.4.1 First Moment Dynamics

We can derive the dynamics for the moments of the estimate and discrete value. Because there are only discrete dynamics and no flow, substituting the definition of the ex-

tended generator function (4.6) into Thm. 1 (4.7) gives

$$\frac{d}{dt}\langle\psi(\hat{x})\rangle = \left\langle k \left( \sum_{i<j} \psi(\phi_{i,j}(\hat{x})) - \psi(\hat{x}) \right) \right\rangle,$$

where the indices  $i < j$  refer to the possible interactions between robots  $i$  and  $j$ ,  $\psi(\hat{x})$  is an arbitrary test function, and  $\phi_{i,j}$  is the mapping after a state transition. Since we are interested in the behavior of the estimate, we choose the test function  $\psi(\hat{x}) = \hat{x}$ . The dynamics of the expected value of the estimate vector  $\langle x \rangle$

$$\frac{d}{dt}\langle\hat{x}\rangle = k \left\langle \left( \zeta \sum_{i<j} A_{ij} - \binom{n}{2} I \right) \hat{x} + (1 - \zeta) \sum_{i<j} B_{ij} q \right\rangle, \quad (5.4)$$

where  $\binom{n}{2} = \frac{n(n-1)}{2}$ .

To simplify (5.4), define

$$A \triangleq \sum_{i<j} A_{ij} \quad \text{and} \quad B \triangleq \sum_{i<j} B_{ij}.$$

The matrix  $A$  can be derived as follows:

$$A = \left[ (n-1)a + \left( \binom{n}{2} - (n-1) \right) \frac{1}{\zeta} \right] I + (1-a) (\mathbb{1}\mathbb{1}^T - I).$$

The diagonal terms of  $A$  represent the  $n-1$  interactions for agent  $i$  weighted  $a$  added to the  $\binom{n}{2} - (n-1)$  of non-interactions weighted  $\frac{1}{\zeta}$ . The off-diagonal terms for  $A$  are  $1-a$ . The entries for  $B$  can be similarly derived. It can be shown that

$$A = (n-1) \left( a + \frac{n-2}{2\zeta} \right) + a-1 I + (1-a) \mathbb{1}\mathbb{1}^T \quad (5.5)$$

$$B = \frac{n-1}{n} \mathbb{1}\mathbb{1}^T. \quad (5.6)$$

Therefore, equation (5.4) becomes

$$\frac{d}{dt}\langle\hat{x}\rangle = k \left[ \left( \zeta A - \binom{n}{2} I \right) \langle\hat{x}\rangle + (1 - \zeta) B q \right]. \quad (5.7)$$

Since we are interested in the equilibrium solution, for estimator dynamics (5.4) we set (5.7) to zero so that  $\frac{d}{dt}\langle \hat{x} \rangle = 0$ . The equilibrium solution is

$$\langle \hat{x} \rangle^* = -(1 - \zeta)H^{-1}Bq, \quad (5.8)$$

where

$$H \triangleq \zeta A - \binom{n}{2} I \quad (5.9)$$

**Theorem 2** *The unique fixed point of the estimator is*

$$\langle \hat{x} \rangle^* = x\mathbb{1}. \quad (5.10)$$

The estimates converge to the population fraction for constant discrete states.

**Proof:** *A. First moment equilibrium solution*

We show that (5.10) is an equilibrium solution for the first moment dynamics (5.4). It is equivalent to show

$$H(x\mathbb{1}) = (1 - \zeta)Bq. \quad (5.11)$$

We show that (5.11) is true by substituting the definitions of  $A$  (5.5),  $B$  (5.6), and  $x$  (4.1).

■

**Proof:** *B. Uniqueness for Equilibrium Solution*

To prove that (5.10) is a unique solution, we show that  $H$  (5.9) is invertible in the parameter region of interest. The matrix  $H$  is singular when

$$H = u\mathbb{1}\mathbb{1}^T \text{ and } H = -(n-1)I + (\mathbb{1}\mathbb{1}^T - I),$$

where the eigenvalues are

$$v = 0 \text{ and } u = 1, v = -n.$$

The values that make  $v = 0$  are  $n = 1$ , which is a trivial estimation problem, and  $\zeta a = -1$ , which is not allowable by our parameter choices. To examine the second condition for singularity, we compute  $H\mathbb{1} = (n-1)(1-\zeta)\mathbb{1}$ . This is true when  $v = -n$  and  $u = 1$ , the

parameter  $\zeta = 1$ , which we already know reduces to the traditional consensus case and doesn't converge to our desired fixed point. Thus,  $H$  is invertible in the parameter region of interest. ■

**Theorem 3** *The fixed point  $\langle \mathbf{x} \rangle^* = x\mathbf{1}$  is stable.*

**Proof:** By (5.7), it suffices to show that  $H$  is negative semi-definite. Note that each term in the sum  $A = \sum (A_{ij} - I)$  has the same eigenvalues since the  $A_{ij}$  matrices are permutations of each other. To show that  $A_{ij} - I$  has negative eigenvalues it suffices to show that the eigenvalues of  $A_{ij}$  are less than 1. Thus without loss of generality we can examine

$$A_{12} = \left( \begin{array}{cc|c} v & u & \mathbf{0} \\ u & v & \\ \hline \mathbf{0} & & I \end{array} \right),$$

whose eigenvalues are  $\{1, v - u, v + u\}$ . The eigenvalue 1 has multiplicity  $n - 2$ . Finally, all eigenvalues are less than or equal to 1, based on our assumptions on  $a$  and  $\zeta$ . ■

#### 5.4.2 Second Moment Dynamics

Similarly, we derive the dynamics for the second moment  $\langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle$ . Using the extended generator  $\mathcal{L}$  (4.6) gives

$$\frac{d}{dt} \langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle = k \left\langle \sum_{i < j} (C_{ij} \hat{\mathbf{x}} + D_{ij} \mathbf{q}) (C_{ij} \hat{\mathbf{x}} + D_{ij} \mathbf{q})^T \right\rangle - k \binom{n}{2} \langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle \quad (5.12)$$

where  $C_{ij} = \zeta A_{ij}$  and  $D_{ij} = (1 - \zeta) B_{ij}$ . The second moment dynamics can be expressed as

$$\begin{aligned} \frac{d}{dt} \text{vec}(\langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle) &= k \sum_{i < j} \left[ C_{ij} \otimes C_{ij} \text{vec}(\langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle) + C_{ij} \otimes D_{ij} \text{vec}(\langle \mathbf{x} \mathbf{x}^T \rangle) \right. \\ &\quad \left. + D_{ij} \otimes C_{ij} \text{vec}(\langle \mathbf{x} \mathbf{x}^T \rangle) + D_{ij} \otimes D_{ij} \text{vec}(\mathbf{q} \mathbf{q}^T) \right] - k \binom{n}{2} \text{vec}(\langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle) \end{aligned}$$

where  $\otimes$  is the Kronecker product and  $\text{vec}(\cdot)$  is the vector representation of a matrix where matrix columns are concatenated vertically.

The equilibrium value of the second moment is a tedious function of  $\hat{\mathbf{x}}$  and  $\mathbf{q}$  and the parameters  $\zeta$ ,  $a$ , and  $n$ . However, a simple expression for the equilibrium can be obtained when  $\zeta = 1$ , which corresponds to linear average consensus. Our argument amounts to an alternative proof that LAC consensus in this setting converges to the average of the initial conditions of the estimates with probability one. Said differently, when  $\zeta = 1$  the variance at equilibrium is 0 (even if the estimates do not equal the population fraction).

When  $\zeta = 1$ , the expression for the second moment dynamics reduces to

$$\frac{d}{dt} \langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle = \sum_{i < j} \mathbf{A}_{ij} \langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle \mathbf{A}_{ij}. \quad (5.13)$$

Setting the derivative to zero and solving for  $\langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle$  yields  $\langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle = w \mathbf{1} \mathbf{1}^T$  where  $w$  is a scalar. Now, define

$$V = \mathbf{1}^T \langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle \mathbf{1}.$$

Multiplying (5.13) by  $\mathbf{1}^T$  on the left and by  $\mathbf{1}$  on the right, noting that  $\mathbf{1}^T \mathbf{A}_{ij} = \mathbf{1}^T$ , and noting that  $\mathbf{A}_{ij} \mathbf{1} = \mathbf{1}$  shows that  $\frac{d}{dt} V = 0$ .

Now, initially  $\langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle = \hat{\mathbf{x}}(0) \hat{\mathbf{x}}^T(0)$  (i.e. the variances are zero initially since we start with deterministic values for  $\hat{\mathbf{x}}$ ). Thus,  $V^* = V(0)$ . Also, the equilibrium value for  $\langle \hat{\mathbf{x}} \rangle$  (only when  $\zeta = 1$ ) is

$$\eta \triangleq \frac{1}{n} \mathbf{1}^T \mathbf{x}(0)$$

so that  $V(0) = \mathbf{1}^T \hat{\mathbf{x}}(0) \hat{\mathbf{x}}^T(0) \mathbf{1} = \eta^2 n^2$ . Thus, writing out  $V(0) = V^*$  we have

$$\eta^2 n^2 = w n^2$$

so that  $w = \eta^2$ . Thus, the covariance matrix at equilibrium is

$$\mathbf{C} = \langle \hat{\mathbf{x}} \hat{\mathbf{x}}^T \rangle^* - \langle \hat{\mathbf{x}} \rangle^* \langle \hat{\mathbf{x}}^T \rangle^* = w \mathbf{1} \mathbf{1}^T - \eta^2 \mathbf{1} \mathbf{1}^T = \mathbf{0}.$$

It is also evident that as  $\zeta$  decreases, the variances increases with the correct expected value of the estimate. This is illustrated in Fig. 5.2 where the estimator tracks a changing



population fraction  $x(t)$ . When  $\zeta = 1$ , the estimator cannot track the changing value but converges with zero variance. When  $\zeta < 1$ , the estimate can track  $x(t)$ , but with a non-zero variance at steady state.

**Theorem 4** *The equilibrium of the second moment dynamics is stable.*

Since we consider  $\mathbf{q}$  to be constant, we examine the following matrix to determine stability of  $\langle \hat{\mathbf{x}}\hat{\mathbf{x}}^T \rangle$ :

$$\mathbf{M} \triangleq k \sum_{i < j} \mathbf{A}_{ij} \otimes \mathbf{A}_{ij} - k \binom{n}{2} \mathbf{I} = k \sum_{i < j} (\mathbf{A}_{ij} \otimes \mathbf{A}_{ij} - \mathbf{I}).$$

Call  $\lambda_i$  the eigenvalues of  $\mathbf{A}$  and  $\mu_i$  the eigenvalues of  $\mathbf{B}$ . Then the eigenvalues of  $\mathbf{A} \otimes \mathbf{B} = \lambda_i \mu_i$ . We showed in Theorem 3 that the eigenvalues of  $\mathbf{A}_{ij}$  were  $\{1, v - u, v + u\}$ . Thus, the eigenvalues  $\mathbf{A}_{ij} \otimes \mathbf{A}_{ij}$  are  $\{1, v - u, v + u, (v - u)^2, (v - u)(v + u), (v + u)^2\}$ , which are all less than 1, based on our assumptions on  $a$  and  $\zeta$ .

As in the proof for Theorem 3 each term in this sum is symmetric and negative semi-definite, from which it can be concluded that  $\mathbf{M}$  itself is negative semi-definite.

#### 5.4.3 Simulation Results

We demonstrate the algorithm by directly simulating the system using the *Stochastic Simulation Algorithm* [42]. Figure 5.2 shows simulations of the estimator with various parameters. Note that when  $\zeta = 0$  our formulation is reduced to LAC, as seen in Fig.5.2(a). Choice of parameter affects the rate of convergence rate and variance. We also verified these results experimentally.

Figures 5.2(a), 5.2(b), and 5.2(c) show a step change of the discrete states from a population fraction of  $\frac{3}{7}$  to  $\frac{4}{7}$  indicated in green. The purple regions indicate the mean plotted with a standard deviation window. The blue lines indicate single estimate trajectories. Figures 5.2(d), 5.2(e), and 5.2(f) depict histograms for the plots above and the correct state is plotted as a dotted blue line. Figure 5.2(a) shows IBC with  $\zeta = 1$ , where the estimator does not track the population fraction. Figure 5.2(d) shows the histogram of the estimates converging with zero variance to the average of initial states but unable to track the new average. Figure 5.2(b) shows IBC with  $\zeta = 0.9$ , where the estimator tracks the population

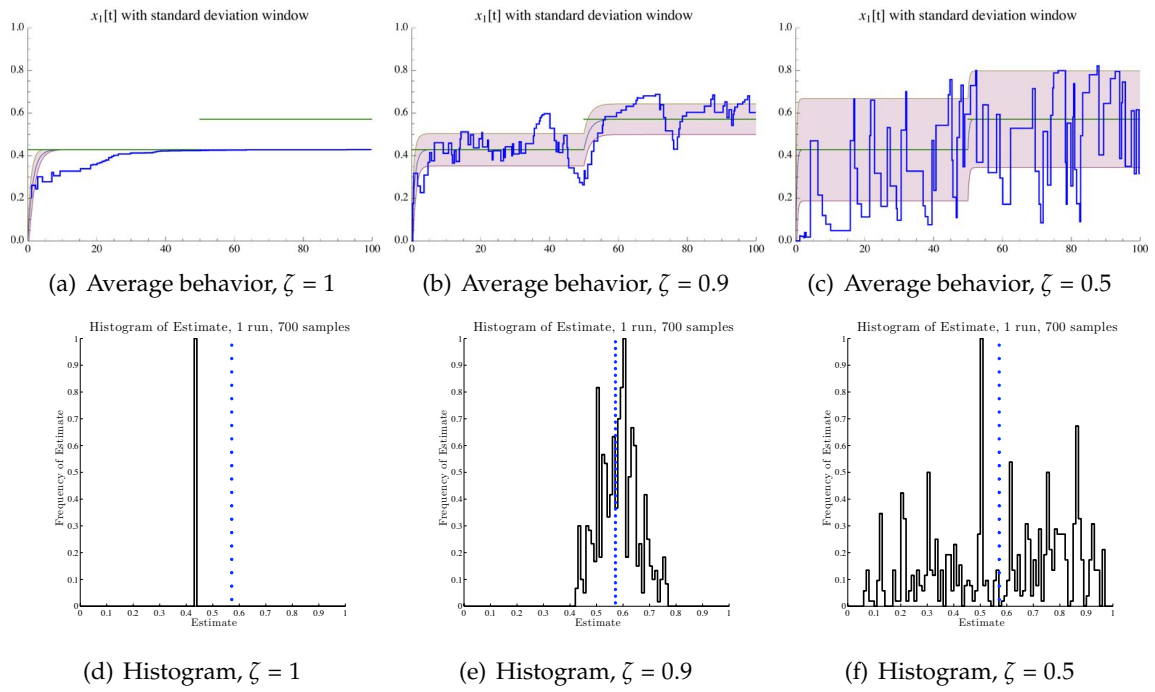
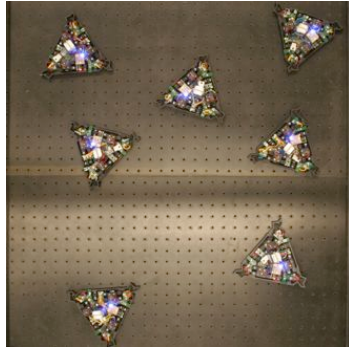
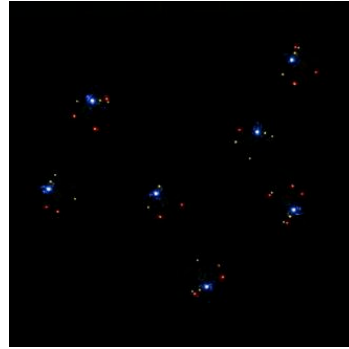


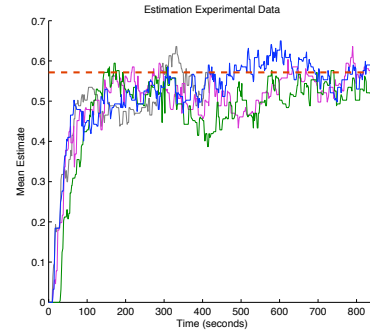
Figure 5.2: Comparison of simulated data with varying consensus parameter  $\zeta$  for 7 robots. Figures 5.2(a), 5.2(b), and 5.2(c) show a step change of the discrete states from a population fraction of  $\frac{3}{7}$  to  $\frac{4}{7}$  indicated in green. Figures 5.2(d), 5.2(e), and 5.2(f) depict histograms for the plots above and the correct state is plotted as a dotted blue line.



(a) Photo of the Programmable Parts



(b) Photo of Programmable Parts with LEDs indicating estimates.



(c) Experimental Results.

Figure 5.3: Experimental setup for the Programmable Parts Testbed. a) PPT with lights on. b) PPT with LEDs indicating state. c) Estimate data collected on PPT.

fraction change. Figure 5.2(e) shows the estimate centered about the population fraction at the end of the simulation with a finite variance. Figure 5.2(c) shows IBC with  $\zeta = 0.5$ . It converges faster than the case where  $\zeta = 0.9$  (Fig.5.2(b)), but with a higher variance as seen when comparing Fig. 5.2(e) with Fig. 5.2(f).

#### 5.4.4 Experimental Results

The PPT has been adapted to make internal estimation states observable by an overhead camera. Each robot computes its estimate with 7-bit accuracy and displays it as a 5-bit number using LEDs. Each robot has a bright blue LED in its center and a green LED to indicate the lowest bit of the estimate, and continuing clockwise to indicate the binary estimate. The estimates indicated by the LEDs were automatically extracted from images using MATLAB. The robots display a quantization error of  $2^{-7}$  in Figure 5.3(c), caused by the discretization of the measurement (output LEDs).

Fig 5.3(a) shows the robots and Fig. 5.3(b) shows the lighting conditions for runs. On each robot the blue LED points to the lowest significant bit to indicate a binary estimate. Figure 5.3(c) shows experimental data on PPT. Each line represents a run with all estimates averaged together. (Current setup does not allow for individual tracking). The red dashed line indicates the average discrete state in all runs. The parameters for these runs are  $\zeta = 0.8$

and  $a = 0.5$ . The runs are fairly accurate given the quantization.

In this section, we introduced the estimation problem and introduce IBC. The estimator was analyzed and proven to converge to the population fraction. The estimator was then demonstrated in simulation and experimentation.

### 5.5 Distributed Control of Stoichiometry

We now address the *control problem* discussed in the problem statement (Sec. 5.1). We define a rate at which the robots flip their discrete states from 0 to 1 and *vice versa* so that (a) the population fraction  $x(t)$  approaches the reference  $r$  and (b) the robots eventually stop switching. For now we assume that the robots have perfect knowledge of  $x(t)$ . Later, we replace this knowledge with the estimated value computed in the previous section. The update rule for robot  $i$  when changing its discrete state is simply

$$q_i(t^+) = 1 - q_i(t^-). \quad (5.14)$$

There are many possible control schemes. Here is a simple one: robot  $i$  toggles its state according to (5.14) at the rate

$$K_i \triangleq |q_i - r| |\hat{x}_i - r|. \quad (5.15)$$

We choose rate  $K_i$  as this product so that robot  $i$  is in state  $q_i = 1$  for the fraction  $r$  of the time. The second term in the product  $|\hat{x} - r|$  makes the rate  $K_i = 0$  when the estimate of the population fraction  $\hat{x}$  is equal to the reference  $r$ .

#### 5.5.1 Model

We examine the convergence of our controller by examining it without the estimator. Here, we assume that the controller has perfect knowledge of the population fraction  $x$ , so let  $\hat{x}_i = x$  in (5.15). Figure 5.4 shows the model for the states of robot  $i$ .

We use a birth-death chain to describe robots switching between two states. Consider the random variable  $N(t) = \sum_{i=1}^n \frac{1}{n} q_i(t)$ . Define  $\mu_i$  to be the rate at which  $N$  transitions from  $N = i$  to  $N = i + 1$ . Similarly, let  $\lambda_i dt$  to be the rate at which  $N$  transitions from  $N = i$  to

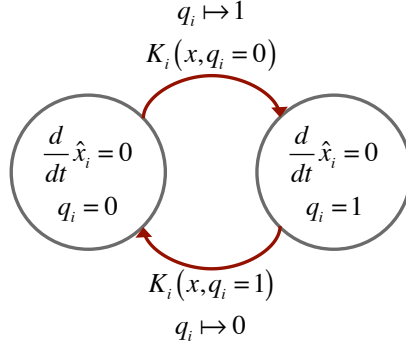


Figure 5.4: State switching controller from the point of view of robot  $i$ . Robot  $i$  toggles its state at rate  $K_i$ .

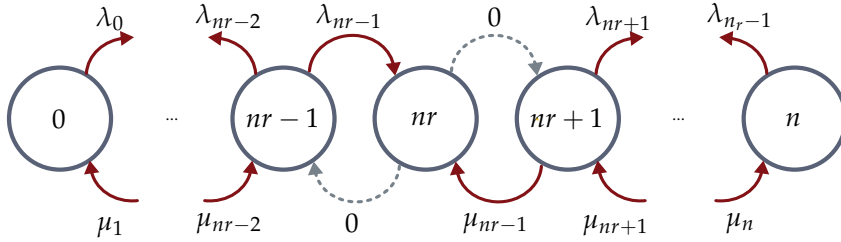


Figure 5.5: Birth-death process. States are labeled  $N = nx$ , the number of robots with discrete state 1. Rates  $\mu_i$  indicate increases of this number while  $\lambda_i$  indicate decreases. Note that state  $nr$  is a sink state since rates  $\mu_{nr} = \lambda_{nr} = 0$ .

$N = i - 1$ . Then, we have a birth-death chain where

$$\mu_i = i(1-r) \left| \frac{i}{n} - r \right| \quad \text{and} \quad \lambda_i = (n-i)r \left| \frac{i}{n} - r \right|.$$

To understand the expression for  $\mu_i$  note that when  $N(t) = i$ , there are  $i$  different robots that can transition from 0 to 1 and they each do so at the same rate  $\lambda_i = |q_i - r||x - r|$ . Since  $q_i = 0$  and  $x = \frac{i}{n}$ , this expression can be reduced to  $\lambda = (1-r) \left| \frac{i}{n} - r \right|$ . The expression for  $\mu_i$  is similar.

Now, choose  $r = n_0/n$  for some  $0 \leq n_0 \leq n$ . Then  $\mu_{n_0} = \lambda_{n_0} = 0$  and the state  $n_0$  is absorbing: all of the probability mass of  $N(t)$  eventually flows to  $n_0$ . Since  $n_0 = rn$ ,  $x(t)$

approaches the reference  $r$ . This is illustrated in Fig. 5.5. In simulation, the system reaches the sink state immediately, so we omit the results here.

## 5.6 Estimation and Control

We have shown that the estimator converges (proof, simulation, experiment) and that the controller converges (proof), and now we address the simultaneous estimation and controller problem by demonstrating that they converge in simulation. Using the estimator for the controller gives control rate  $K_i = |q_i - r||\hat{x}_i - r|$ , where the estimate of the population fraction  $\hat{x}_i$  is used instead of perfect knowledge of the population fraction  $x$ . Figure 5.6 shows the model of the estimator and controller together.

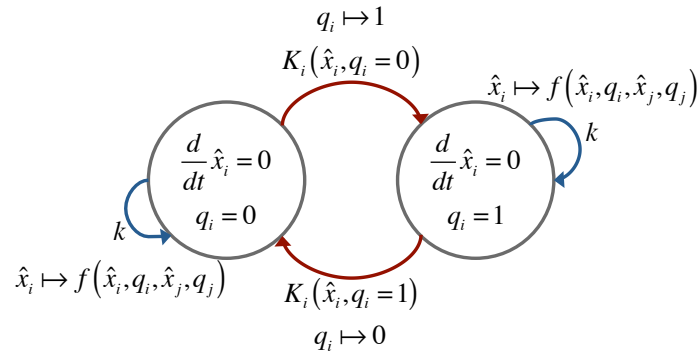


Figure 5.6: Estimator and controller combined. Estimator events are shown in blue and controller events are shown in red.

We simulate the combined system and show the results in Fig 5.7. Figure 5.7(a) shows of 50 runs with 10 robots with parameters  $\zeta = 0.9$  and  $a = 0.9$ . The red dashed line is the reference ( $r = 0.2$ ), the green line indicates the average of discrete states  $x$ , the blue line is the average of estimates over all robots. Figure 5.7(b) plots the controller rate  $K_i$  as a function of time. The control effort decreases over time, but does not stop like the perfect controller. Figure 5.7(c) shows a histogram for 50 runs for the last hundred timesteps over all runs with the peak about the reference.

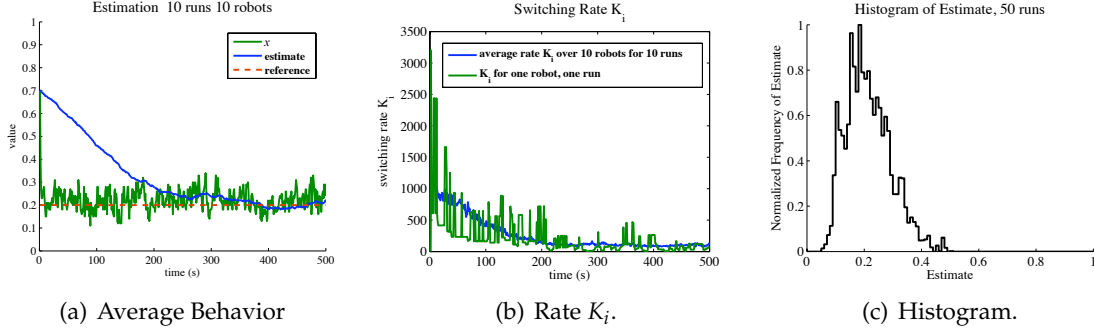


Figure 5.7: Controller with estimator. a) Estimate trajectory averaged over 10 runs. b) Controller rate  $K_i$  decreases as a function of time

### 5.7 Robustness to Dropped Messages

Agreement algorithms are widely used on multi-agent systems to diffuse information and combine local measurements of a global quality. An example of this type of algorithm is LAC [86], in which agents compute the global mean of a locally measured quantity by repeatedly computing pairwise averages between agents. Assuming no communications messages are lost and no part of the system is disconnected indefinitely, each agent's local estimate will converge to the global mean. Agents are in agreement when the variance of the estimates on between agents is small. However, in the presence of intermittent communications failures, information about the global mean can be lost permanently, making agreement to the correct global value impossible to reach.

In this section we extend IBC by examining the algorithm in the event of dropped messages; we demonstrate that the agents converge with bounded error. Second, we use LAC as a benchmark and compare its performance via root mean square error. We compare the two algorithms analytically, via simulation, and experimentally on a system of 20 mobile robots.

#### 5.7.1 Related Work

There is a great deal of literature on agreement in multi-agent systems and on stochastic systems in general. We focus here on faulty agreement. Although LAC is not robust to

message loss, we use it as a baseline because it is representative of a large class of consensus algorithms; those in which agents converge to the convex combination of initial states and a globally invariant quantity is maintained through local interactions. Olfati-Saber *et al.* [86] refer to the class of algorithms with this invariance property as *average-consensus* algorithms. To our knowledge, computing global averages is the simplest such consensus application, thus, we found it suitable to use LAC to compute the global average as a baseline for comparison.

Consensus problems have attracted the interest of many authors particularly under the condition of faulty communication. Often, consensus is considered with additive noise representing sensor error [96, 130, 53] and also as packet losses [52, 51]. Huang assumes an observer under the event of a known packet loss [52]. Our work is similar to the latter, in that we can also model packet loss as a Markov process. However, we assume that packet losses are not observable.

Our experimental multi-robot system generates random graphs through the mobility of the robots. We assume the system is well-mixed; that is, every agent is equally likely to interact with any other agent in the system. Thus, we can associate a rate  $k$  for interactions, where  $k dt$  is the probability in the next  $dt$  seconds of an event occurring.

However, in the presence of intermittent communications failures, information about the global mean can be lost permanently, making agreement to the correct global value impossible to reach. For robots  $i$  and  $j$  that interact and robot  $i$  drops a message, the update function is

$$\begin{aligned}\hat{x}_i(t+1) &= \hat{x}_i(t) \\ \hat{x}_j(t+1) &= f(\hat{x}_j(t), q_j, \hat{x}_i(t), q_i) \\ \hat{x}_l(t+1) &= \hat{x}_l(t) \text{ for all } l \neq i, j,\end{aligned}$$

where  $f(\hat{x}_i(t), q_i, \hat{x}_j(t), q_j)$  is defined in (5.1). This results in the following update matrices where



$$\begin{aligned}
C_{iji}(j, j) &= a & D_{iji}(j, j) &= \frac{1}{n} \\
C_{iji}(j, i) &= (1 - a) & D_{iji}(j, i) &= \frac{n-1}{n} \\
C_{iji}(i, i) &= C_{iji}(l, l) = \frac{1}{\zeta}
\end{aligned} \tag{5.16}$$

for a message dropped by robot  $i$ , with all the remaining matrix entries 0. Note that matrices  $C_{iji}$  and  $D_{ij}$  are applied when a message dropped by robot  $i$  and matrices  $C_{ijj}$  and  $D_{ijj}$  are applied when a message dropped by robot  $j$ . We introduce  $\gamma$  to represent the fraction of successful messages, which is uniform across all robots. For any interaction between robots  $i$  and  $j$  where a message is dropped by robot  $i$ , we revise the estimator update rule to be

$$\hat{\mathbf{x}} = \gamma (\zeta \mathbf{A}_{ij} + (1 - \zeta) \mathbf{B}_{ij}) + (1 - \gamma) (\zeta \mathbf{C}_{iji} + (1 - \zeta) \mathbf{D}_{iji}) \tag{5.17}$$

If we set  $\gamma$  to zero, then we get standard IBC equation as before 5.2.

### 5.7.2 Moment Dynamics

Because the system is a stochastic process, we reason about the moments of the system, particularly, the mean for the estimate  $\hat{\mathbf{x}}$ . The first moment of the estimator process is examined by using the extended generator  $\mathcal{L}$  in (4.7). Substituting our new update into (4.7), gives

$$\begin{aligned}
\frac{d}{dt} \langle \hat{\mathbf{x}} \rangle &= \gamma k \left\langle \left( \sum_{i < j} (\zeta \mathbf{A}_{ij} - \mathbf{I}) \right) \hat{\mathbf{x}} + (1 - \zeta) \sum_{i < j} \mathbf{B}_{ij} \mathbf{q} \right\rangle \\
&+ (1 - \gamma) k \left\langle \left( \sum_{i < j} \sum_{m \in i, j} (\zeta \mathbf{C}_{ijm} - \mathbf{I}) \right) \hat{\mathbf{x}} + (1 - \zeta) \sum_{i < j} \sum_{m \in i, j} \mathbf{D}_{ijm} \mathbf{q} \right\rangle
\end{aligned} \tag{5.18}$$

where the indices  $i < j$  refer to the possible interactions between robots  $i$  and  $j$ . The second sum where  $m \in i, j$  indicates that either robot  $i$  or  $j$  can drop a message.

Equation (5.18) can be simplified as follows. Define

$$\begin{aligned}
\mathbf{A} &\triangleq \sum_{i < j} \mathbf{A}_{ij}, & \mathbf{B} &\triangleq \sum_{i < j} \mathbf{B}_{ij}, \\
\mathbf{C} &\triangleq \sum_{i < j} \mathbf{C}_{iji} + \sum_{i < j} \mathbf{C}_{ijj} & \mathbf{D} &\triangleq \sum_{i < j} \mathbf{D}_{iji} + \sum_{i < j} \mathbf{D}_{ijj}.
\end{aligned}$$

For now, we consider the estimate independent from any changing discrete values and thus, assume that the discrete state  $q$  is constant. Matrices  $A$  and  $B$  are defined in (5.5) and (5.6), respectively. It can be shown that

$$\begin{aligned} C &= A + \frac{1}{\zeta} \binom{n}{2} I. \\ D &= B = \frac{n-1}{n} \mathbf{1}\mathbf{1}^T. \end{aligned} \tag{5.19}$$

Equation (5.18) reduces to (5.7). the same as IBC without dropped messages. This yields the same desired equilibrium solution  $\langle \hat{x} \rangle^* = x\mathbf{1}$  for the first moment as above (5.10). Note that as the consensus parameter  $\zeta$  approaches 1, the variance decreases. This is observable in our data.

### 5.7.3 Analysis

We numerically solve the differential equation (5.18) for the mean  $\langle \hat{x} \rangle$  and second moment  $\langle \hat{x}\hat{x}^T \rangle$  for LAC and IBC for  $\gamma = 0.5$  and plot them in Fig 5.9. Figures 5.8(a) and 5.8(b) show the analytical solutions for the expected value of the estimate for success fraction  $\gamma = 0.25$ . The standard deviation window around the mean is displayed in light purple. The reference population fraction is plotted with a dashed green line. An example robot trajectory for one experimental run is plotted in orange. Note that for LAC, Fig. 5.8(a), the single trajectory converges to an incorrect steady state value, while the single robot for IBC, Figure 5.8(b), oscillates about the correct value. These results are further supported by simulation results.

Figure 5.8 shows the standard deviation at equilibrium of LAC to IBC as the function of communications failures,  $1 - \gamma$ . The numerical solutions are found at the steady-state of the mean estimate dynamics (5.7) and the second moment dynamics. Effectively, this evaluates the standard deviation across multiple executions of each algorithm, not within a single run. Note that the standard deviation of IBC is almost unaffected by communications loss, but the standard deviation of LAC approaches the maximum value – the initial conditions.

Both algorithms require pairwise updates between neighbors, which required a com-

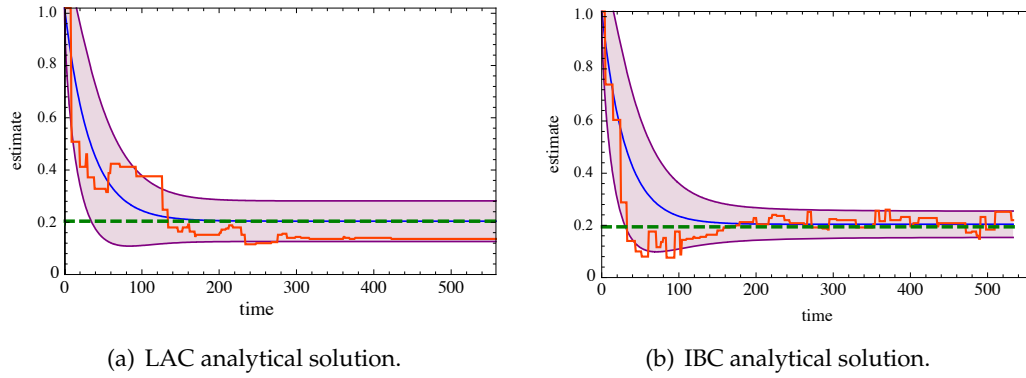


Figure 5.8: Analytical solutions plotted with experimental data for LAC and IBC. Parameters chosen  $\zeta = 1, \gamma = 0.25$  for LAC and  $\zeta = 0.95, \gamma = 0.25$  for IBC.

munications handshaking protocol. First, if a robot is not updating, it selects a neighbor at random and transmits a partner request message. Then, if that neighbor is not updating, it replies with a partner acknowledge message and runs the update rule. Upon receipt of the acknowledgement, the original robot runs its update rule with probability  $\gamma$ . This success probability parameter  $\gamma$  allows us to simulate communication losses much larger than the actual loss of the system, which is less than 1% [79].

#### 5.7.4 Simulated Results

We simulated the system using the *Stochastic Simulation Algorithm* [42]. At every time step in the simulation, a pair of robots is chosen to interact and update their estimates. This produces a well-mixed system, *i.e.* interactions between any pair is uniform. Two representative trajectories for the simulated results are shown in Figure 5.10(a) and Figure 5.10(b), for LAC and IBC respectively. The communications success rate is  $\gamma = 0.25$ . LAC converged with variance 0 to a value that is not the correct value while IBC oscillates about the correct value with a finite variance. The results from simulation are confirmed by experimental results.

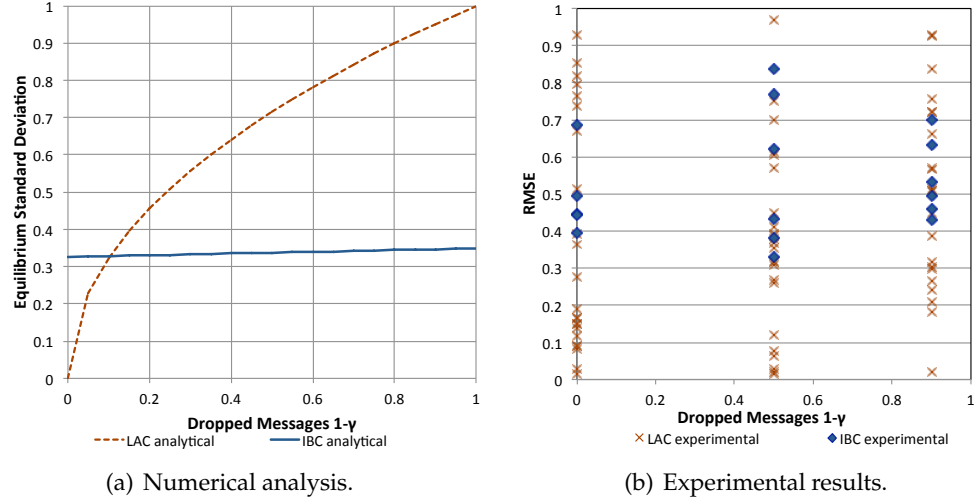


Figure 5.9: LAC compared to IBC numerically for dropped messages. (a) We plot the standard deviation as a function of communication failures ( $1 - \gamma$ ) for 20 robots. The red (dashed) line is the numerical standard deviation for LAC and the blue (solid) line is the standard deviation for IBC. (b) LAC compared to IBC experimentally. We conducted a total of 96 runs. The experimental data shows a much larger variance for the LAC RMSE than the IBC RMSE.

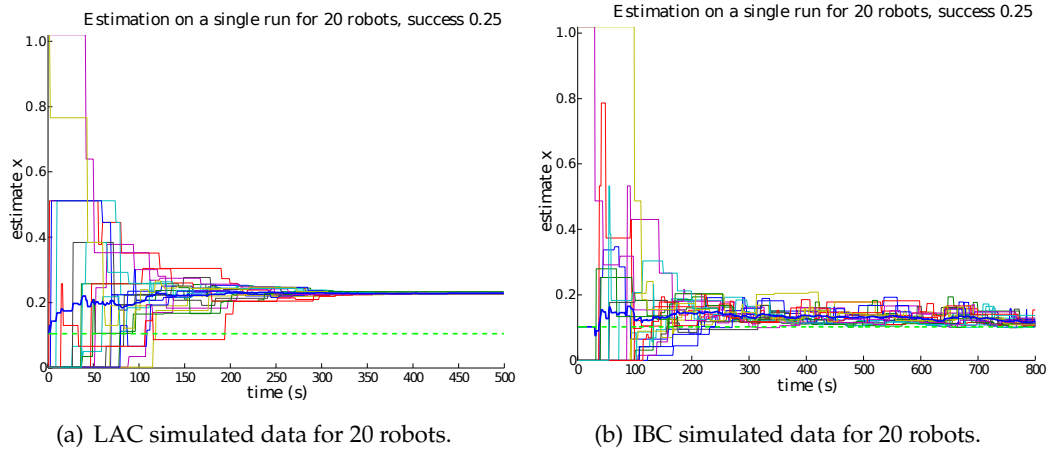


Figure 5.10: Simulated data for LAC and IBC. Parameters chosen  $\zeta = 1, \gamma = 0.25$  for LAC and  $\zeta = 0.95, \gamma = 0.25$  for IBC.

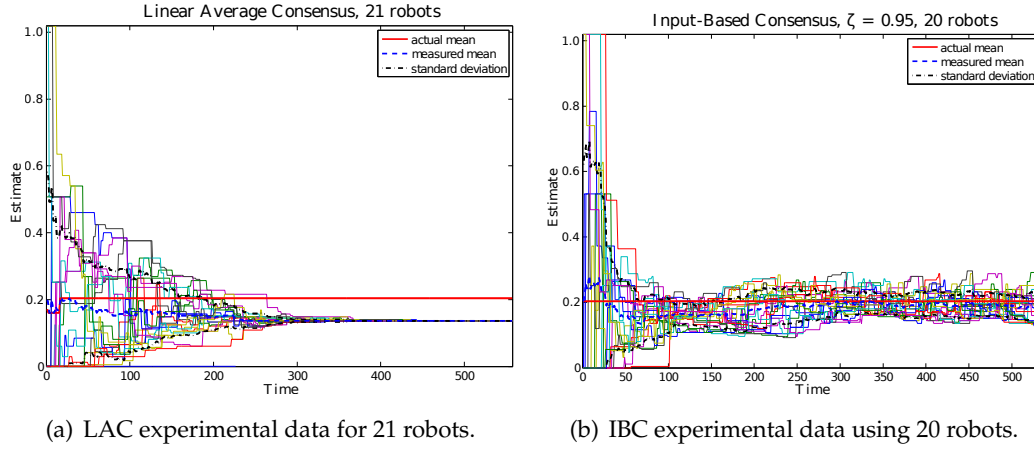


Figure 5.11: Experimental data comparison for LAC and IBC. a) Experimental Data for 21 robots using LAC. The solid red line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories. b) Experimental Data for 20 robots using IBC. The red line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories.

### 5.7.5 Experimental Results

The SwarmBot Testbed [79], as introduced in Ch. 2 was used to validate algorithm performance. The robots are autonomous, using only local computation and communication to run the algorithms. Each robot has an infra-red communication system that allows robots to communicate to neighbors within approximately a 1.0 meter radius. This produces multi-hop networks within the confines of our experimental workspace, which is a  $2.43 \text{ m} \times 2.43 \text{ m}$  ( $8' \times 8'$ ) square.

We generated dynamic network topologies by moving the robots randomly throughout the workspace. Since their communication range is smaller than the workspace dimension, this will force frequent changes in neighbors, producing random interactions. The robots drive straight until they contact an obstacle, then use their bump sensors to estimate the angle of incidence and rotate to “reflect” themselves back into the environment. Each trial was 5 minutes long and initialized with 20 robots. Robots would occasionally run out of batteries during the experiment and were not replaced.



Figure 5.12: A SwarmBot experiment. Each trial started with 20 robots, but often robots would run out of batteries during a trial. These robots were not replaced until the next trial.

A total of 95 experimental runs were performed with the variables  $\gamma = \{0.1, 0.5, 1\}$  and  $\zeta = \{0.95, 1.0\}$ . Figure 5.8(a) shows the analytical solution of the first moment dynamics of the estimate. Figure 5.10(a) shows robots simulated using the Stochastic Simulation Algorithm. Figure 5.11(a) shows data collected from the Swarmbot testbed. Analytical, simulated, and experimental data for IBC  $\zeta = 0.95, \gamma = 0.25$ . Figure 5.8(b) shows the analytical solution of the first moment dynamics of the estimate. Figure 5.10(b) shows robots simulated using the Stochastic Simulation Algorithm. Figure 5.11(b) shows data collected from the Swarmbot testbed.

Figures 5.8(a) and 5.8(b) each show one sample trajectory from the experimental data plotted with the analytical results. We estimate  $k = 0.22$  from our experimental results and see that our data compares nicely to our analytical solution. We compute the root mean square errors (RMSE) for experiments, where RMSE is  $\sqrt{\langle \hat{x} - x \rangle^2}$ .

Figure 5.11 shows the experimental results for 96 trials where the RSME of LAC to IBC are plotted as a function of communications failures,  $1 - \gamma$ . The data supports the analyti-

cal results and simulations. The LAC trials have a much larger RSME across multiple runs. Although LAC produces runs with small RSME, it does not do so consistently. The IBC algorithm produces more consistent RSME, even at which is 90% message loss.

Figure 5.9 shows LAC compared to IBC solved numerically, in simulation, and in experiment. The left side shows LAC and the right side shows IBC; the top row compares numerical solutions, the middle simulation, and the bottom row is experimental data.

Starting with LAC, Figure 5.8(a) shows the analytical solution (blue) of first moment dynamics of the estimate with LAC parameters and standard deviation window (purple). Desired mean (green dashed) plotted with experimental data for one robot (orange). In Figure 5.10(a) LAC is simulated with 20 robots. The green dashed line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories. Lastly, in Figure 5.11(a), experimental data is taken for 21 robots using LAC. The solid red line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories. The trajectories qualitatively behave the same for numerical, simulated, and experimental data.

In Figure 5.8(b) the numerical solution for IBC is shown. The analytical solution (blue) of first moment dynamics of the estimate with IBC parameters and standard deviation window (purple). Desired mean (green dashed) plotted with experimental data for one robot (orange). In Figure 5.10(b) IBC simulated data for 20 robots. The green dashed line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories. Lastly, in Figure 5.11(b) experimental data for 20 robots is collected. The red line indicates the desired mean state. Light lines indicate individual robot trajectories and the solid blue line is the average of trajectories. The trajectories qualitatively behave the same for numerical, simulated, and experimental data.

## 5.8 Extensions for Estimation

### 5.8.1 Problem Statement

We want to demonstrate that estimation can be extended to other applications. For example, we may want to perform assembly on the PPT and estimate population fractions of

all sub-components of an assembly. Thus, we extend IBC to estimate a vector of population fractions. A robot  $i$  now has an *estimate vector*  $\hat{\mathbf{x}}_i = (\hat{x}_a \ \hat{x}_b \ \hat{x}_c)$  representing the population fractions for types  $a, b$ , and  $c$ . The population fractions sum to 1.

### 5.8.2 Algorithm

Much like in 5.1 if two robots  $i$  and  $j$  collide, they update their estimates according to the rule

$$f(\hat{\mathbf{x}}_i, \mathbf{q}_i, \hat{\mathbf{x}}_j, \mathbf{q}_j) = \zeta (a\hat{\mathbf{x}}_i + (1-a)\hat{\mathbf{x}}_j) + (1-\zeta) \left( \frac{1}{n} \mathbf{q}_i + \left( \frac{n-1}{n} \right) \mathbf{q}_j \right).$$

The main difference between this update equation and (5.1) is that  $\hat{\mathbf{x}}_i$  and  $\mathbf{q}_i$  are vectors rather than scalars since each robot estimates a vector of species. Written as matrices, the update rule becomes

$$\hat{\mathbf{x}}(t^+) = \zeta \begin{pmatrix} a\mathbf{I} & (1-a)\mathbf{I} & \mathbf{0} \\ (1-a)\mathbf{I} & a\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{1}{\zeta}\mathbf{I} \end{pmatrix} \hat{\mathbf{x}}(t^-) + (1-\zeta) \begin{pmatrix} \frac{1}{n}\mathbf{I} & (1-\frac{1}{n})\mathbf{I} & \mathbf{0} \\ 1-\frac{1}{n}\mathbf{I} & \frac{1}{n}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{q}(t^-), \quad (5.20)$$

or more compactly and generally

$$\hat{\mathbf{x}}(t^+) = \zeta \mathcal{A}_{ij} \hat{\mathbf{x}}(t^-) + (1-\zeta) \mathcal{B}_{ij} \mathbf{q}(t^-).$$

where  $\mathcal{A}_{ij} = \mathbf{A}_{ij} \otimes \mathbf{I}_n$  and  $\mathcal{B}_{ij} = \mathbf{B}_{ij} \otimes \mathbf{I}_n$ . Matrices  $\mathbf{A}_{ij}$  and  $\mathbf{B}_{ij}$  are defined as in (5.3).

### 5.8.3 Analysis

We can obtain the dynamics for the expected value of the estimate vector for a particular species  $\langle \hat{x}_a \rangle$  using the extended generator (4.7):

$$\frac{d}{dt} \langle \hat{x}_a \rangle = k \left\langle \left( \zeta \sum_{i < j} \mathbf{A}_{ij} - \binom{n}{2} \mathbf{I} \right) \hat{\mathbf{x}}_a + (1-\zeta) \sum_{i < j} \mathbf{B}_{ij} \mathbf{q} \right\rangle \quad (5.21)$$

The expressions for matrices  $\mathbf{A}$  and  $\mathbf{B}$  are the same as (5.5) and (5.6).



Therefore, equation (5.21) becomes

$$\frac{d}{dt}\langle x_a \rangle = k(\mathbf{H}\langle \hat{\mathbf{x}}_a \rangle + (1 - \zeta)\mathbf{B}\mathbf{q}) \quad (5.22)$$

When the dynamics of each estimate are analyzed separately, they reduce to

$$\langle x_i \rangle^* = x\mathbb{1},$$

This has yet to be proven analytically.

#### 5.8.4 Simulation Results

The robots interactions are simulated using the stochastic simulation algorithm [42] and the results are plotted in Figure 5.13(a). Figure 5.13(a) shows one run with the estimate of each of the three different species averaged over ten robots. Figures 5.13(b), 5.13(c), and 5.13(d) show histograms of all the robots for each estimate. The parameters  $a = 0.9$  and  $\zeta = 0.9$ .

The simulation is initialized with one robots of type 1, two robots of type 2, and seven of type 3. The simulations show that the averaged estimate across all robots for any particular species converges to the correct estimate.

### 5.9 Discussion

Much of the work in distributed averaging guarantees convergence for systems that are jointly connected, but IBC drives the system to a value different than the population fraction if the graph induced by the random collisions is not completely connected. We will look at the bounds as to why this is true.

We also examine the topology requirements for linear average consensus and input-based consensus. Linear average consensus for a deterministic switching topology requires that the agents' communication events induce a jointly connected graph over time [58], that is, the union of communication graphs over time is connected. For a stochastic setting one examines the expected value of the graph instead. Hatano and Mesbahi [47] and Tahbaz-

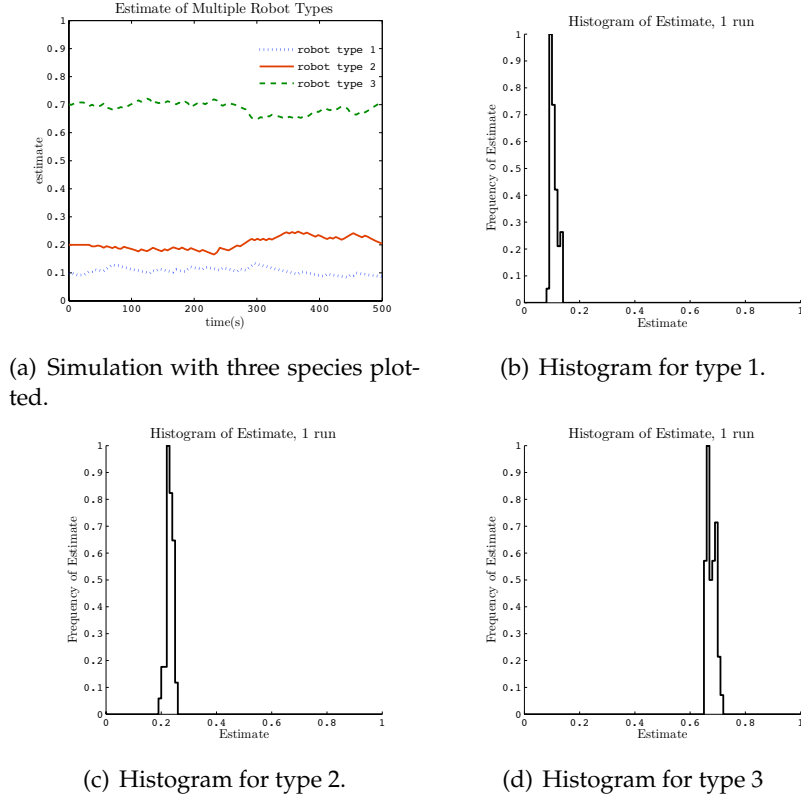


Figure 5.13: Estimation for multiple species. Ten robots estimate 3 types of robots, ten runs averaged and associated histograms. Parameters are chosen to be  $a = 0.9$  and  $\zeta = 0.9$ .

Salehi and Jadbabaie [114] examine this case. In IBC the expected value of the network is a complete graph. However, IBC does not have the same guarantee if the expected graph is not complete; in that event the agents converge to a value.

We consider a “fixed” topology graph in that agents interact stochastically with a set of neighbors instead of all of the other agents.

We also examine the robustness of IBC to lost robots. We can calculate the effect of losing a robot on the global estimate. We use the equilibrium of the estimate in (5.10),  $\langle \hat{x} \rangle = x\mathbb{1}$ . Suppose that one robot has a dead battery and the rest of the robots still believe that there are  $n$  robots in the system, but instead there are actually  $n - 1$  robots. The error

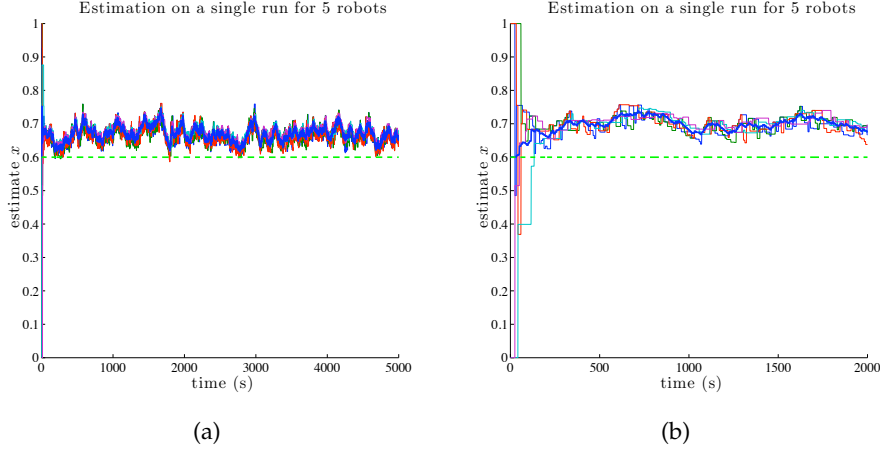


Figure 5.14: IBC for fixed topology graph. All previous plots were generated for a well-mixed environment, where robots communicate at rate  $k$ , and the union of the pairwise interactions is a complete graph. This plot shows a “fixed” topology where robots communicate at rate  $k$ , but the union of the pairwise interactions is one edge short of a complete graph. The error is more severe for graphs of fewer edges.

of the equilibrium is  $\langle \hat{x} \rangle_{actual} - \langle \hat{x} \rangle_{measured} = e\mathbb{1}$ . We can solve this error as

$$e = -(1 - \zeta)H^{-1}(B_{actual} - B_{measured})q$$

where  $B_{actual} = \frac{n-2}{n-1}\mathbb{1}\mathbb{1}^T$  and  $B_{measured} = \frac{n-1}{n}\mathbb{1}\mathbb{1}^T$ . As the number of robots increases, the effect of an incorrect coefficient decreases. For example, parameters for a representative experiment are  $n = 20$  robots, two robots with state 1,  $a = 0.5$  and  $\zeta = 0.95$ . If there are 19 active robots at the end of an experiment, the error is 0.27%. For 18 active robots, the error is 0.58% and so on.

In this chapter, we presented input-based consensus (IBC), model the system as a stochastic hybrid system, examine the first and second moment solutions of the equilibrium and use those analytical solutions to compare with simulations and experimentation. We find that our algorithm converges in mean and variance. Then we examine IBC in the presence of severe message loss and extend it to multiple species.

Looking forward, we would like to find bounds on performance and derive an expression for the error due to dropped messages. In this chapter, we have only demonstrated

recovery of a mean initial condition, but this work can be extended to tracking a changing quantity. Previous work leads [107, 108] us to believe that agreement upon the mean is sufficient for control. Thus, we can extend this work to consensus and control for coordination, such a task assignment, in the presence of lossy communications or with a vector estimate.

## Chapter 6

### FEED-FORWARD CONSENSUS

#### 6.1 Introduction

We address the problem of distributed estimation and control of agents in a stochastic environment. In this chapter, distributed estimation refers to ensemble tracking of a changing value and distributed control refers to asynchronous and simultaneous state assignment to agents.

This work is motivated by, but not limited to, the PPT [63], introduced in Ch. 2. Like input-based consensus, each agent can be one of two discrete states, and calculates an estimated *population fraction* and switch from one discrete state to another. Based on their estimates of the population fraction, agents switch their type to match a desired *reference* population fraction.

The contribution of this chapter is a distributed estimator of the population fraction that informs a stochastic controller and together results in a zero-variance algorithm, an improvement on IBC. We introduce an estimator and controller, prove their convergence separately and together, and demonstrate our results in simulation. While we present this work with the example of stochastic self-assembly, the theory can be generalized to be used in other examples such as robotic swarms [46].

In Ch. 5, we derived a stochastic controller and estimator for robots of two types. We proved that the estimation and control processes were separately stable and demonstrable together. However, the estimator converged to the population fraction with a finite variance. In the current work, we have modified the estimator to more accurately track the population fraction and provide a proof of the estimator and controller working together.

## 6.2 Problem Statement

The estimation problem is to define an estimator function  $(\hat{x}_i, \hat{x}_j) \mapsto f(\hat{x}_i, q_i, \hat{x}_j, q_j)$  so that  $\hat{x}_i(t)$  converges to  $x(t)$  as  $t \rightarrow \infty$  with high probability. Feed-Foward Consensus (FFC) is similar to *linear average consensus*, where estimates are updated in a convex combination. Also, we retain the knowledge of  $q_i$  to add a *feed-forward injection* in the estimate  $\hat{x}$  when a robot toggles  $q$ . In particular, if robot  $i$  interacts with robot  $j$  at time  $t$  then the robots update their estimates according to

## 6.3 Algorithm

$$\begin{aligned}\hat{x}_i(t^+) &= a\hat{x}_i(t^-) + (1-a)\hat{x}_j(t^-) \\ \hat{x}_j(t^+) &= (1-a)\hat{x}_i(t^-) + a\hat{x}_j(t^-) \\ \hat{x}_k(t^+) &= \hat{x}_k(t^-) \text{ for all } k \neq i, j.\end{aligned}\tag{6.1}$$

Here  $a \in (0, 1)$  is a design parameter defining the weight of a robot's own estimate. The symbols  $t^-$  and  $t^+$  denote the times immediately before and after the interaction, respectively. The last line of the above update rule represents the fact that robots not participating in the interaction do not update their estimates. The update equations can be written using matrices:

$$\hat{\mathbf{x}}(t^+) = \mathbf{A}_{ij}\hat{\mathbf{x}}(t^-),\tag{6.2}$$

where  $\mathbf{A}_{ij}$  is defined according to equation (6.1). The update rule results in linear average consensus.

For the controller, we define a rate  $K_i$  at which the robots should flip their discrete states from 0 to 1 and *vice versa* so that (a)  $x(t)$  approaches  $r$  and (b) the robots eventually stop switching. The update rule for robot  $i$  when changing its discrete state is

$$\begin{aligned}q_i(t^+) &= 1 - q_i(t^-) \\ \hat{x}_i(t^+) &= \hat{x}_i(t^-) + q_i(t^+) - q_i(t^-) \\ &= \hat{x}_i(t^-) + 1 - 2q_i(t^-).\end{aligned}$$

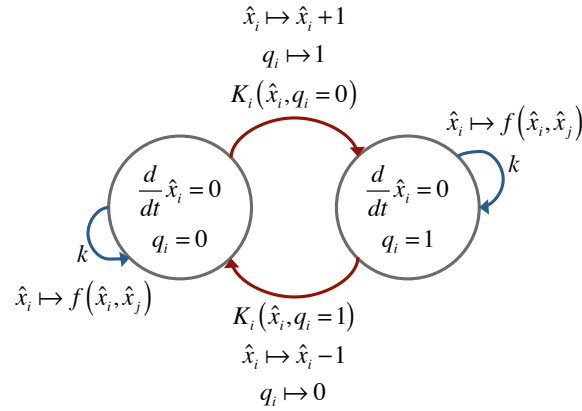


Figure 6.1: FFC modeled for robot  $i$ . Estimator events occur at rate  $k$  and are shown in blue. Controller events occur at rate  $K_i$  and are shown in red.

Note the feed-forward injection contribution in the update rule for the estimate  $\hat{x}$ . The estimate update is also used by Tsitsiklis *et al.* [119] to describe an exogenous measurement. The update equation for the estimate  $\hat{x}_i$  preserves the relationship for the population fraction  $x$ , so that

$$\sum_i q_i = \sum_i \hat{x}_i. \quad (6.3)$$

This allows the estimator to track the population fraction of the states.

## 6.4 Model

We model our system as a stochastic process that occurs at rate  $k$  for estimation updates and  $K_i$  for controller updates. It is similar to the model for IBC in Ch. 5. The differences are that the estimator updates estimates only and the controller updates both the discrete states and estimates. Figure 6.1 shows the estimator (blue arrows) and controller (red) for robot  $i$ .

## 6.5 First Moment Dynamics

The first moment of the estimator process are examined uses the Stochastic Hybrid Systems (SHS) formalism. To reason about the dynamic behavior of a SHS, we use the

extended generator theorem (4.7).

A SHS consists of a set of discrete and continuous states governed by a state-dependent stochastic differential equation. Our system has neither continuous flow nor discrete update, thus we do not use the entire formalism described by Hespanha [48]. In the present case, the extended generator  $\mathcal{L}$  is defined as in (4.6)

$$\mathcal{L}\psi(\hat{\mathbf{x}}, \mathbf{q}) = \sum_{i < j} \lambda_i (\psi(\phi_i(\hat{\mathbf{x}}, \mathbf{q})) - \psi(\hat{\mathbf{x}}, \mathbf{q})),$$

where  $\psi$  is the test function,  $\phi_i$  is the update function for interaction  $i$ , and  $\lambda_i$  is the associated rate. where the indices  $i < j$  refer to the possible interactions between robots  $i$  and  $j$ . We are interested in the expected value of the estimates in the ensemble behavior of the estimation algorithm. We choose  $\psi(\hat{\mathbf{x}}, \mathbf{q}) = \hat{\mathbf{x}}$  so that

$$\frac{d}{dt} \langle \hat{\mathbf{x}} \rangle = k \left\langle \sum_{i < j} f_{i,j}(\hat{\mathbf{x}}) - \hat{\mathbf{x}} \right\rangle + \left\langle \sum_{i=1}^n K_i (g_i(q_i) - q_i) \right\rangle, \quad (6.4)$$

where  $f_{ij}$  indicates the estimator update and  $g_i$  indicates the controller update. In our example, the estimate update is  $f_{ij}(\hat{\mathbf{x}}) = A_{ij}\hat{\mathbf{x}}$ , which occurs at rate  $k$ . The update for the discrete value is  $g_i(q_i) = 1 - q_i$ , which occurs at rate  $K_i$ .

Equation 6.4 simplifies to

$$\frac{d}{dt} \langle \hat{\mathbf{x}} \rangle = k \left( A - \binom{n}{2} I \right) \langle \hat{\mathbf{x}} \rangle + \left\langle \sum_{i=1}^n K_i (1 - 2q_i) \right\rangle \quad (6.5)$$

## 6.6 Analysis

### 6.6.1 Estimator Stability in Isolation

We show that the dynamics of the estimator are equivalent to Laplacian dynamics, therefore is stable.

**Theorem 5** *The dynamics of the estimator, without control, is equivalent the Laplacian matrix for a complete graph. As a result,*



1. The unique fixed point of  $\frac{d}{dt}\langle\hat{\mathbf{x}}\rangle$  is  $\langle\hat{\mathbf{x}}\rangle^* = x\mathbb{1}$ .
2. The Laplacian matrix has a unique zero and negative eigenvalues, thus the fixed point  $\langle\hat{\mathbf{x}}\rangle^* = x\mathbb{1}$  is stable.

In other words, the estimates converge to the population fraction, assuming the discrete states are constant. Furthermore, this system is stable.

If the graph induced by the random interactions of the robots are considered together, the graph is complete and the degree of each node is  $n - 1$ . Also, since the graph is complete, each node is connected to every other node and the adjacency matrix has a 1 in every entry except along the diagonal. Thus, the adjacency matrix is  $\mathbb{1}\mathbb{1}^T - \mathbf{I}$ . For a complete graph, the Laplacian matrix is

$$\mathbf{L} = (n - 1)\mathbf{I} + \mathbb{1}\mathbb{1}^T - \mathbf{I}. \quad (6.6)$$

To examine the estimator stability, we consider the estimator without control. The discrete value  $q_i$  is constant for all  $i$  and  $K_i = 0$ . Equation (6.5) becomes

$$\frac{d}{dt}\langle\hat{\mathbf{x}}\rangle = k \left( \mathbf{A} - \binom{n}{2} \mathbf{I} \right) \langle\hat{\mathbf{x}}\rangle.$$

Thus, to show Theorem 5, it is equivalent to show that

$$\mathbf{H} = w\mathbf{L}, \quad (6.7)$$

where  $w$  is a weight and  $\mathbf{L}$  is the Laplacian. We define

$$\mathbf{H} \triangleq \mathbf{A} - \binom{n}{2} \mathbf{I} \quad (6.8)$$

where

$$\mathbf{A} = \left[ (n - 1)a + \binom{n}{2} - (n - 1) \right] \mathbf{I} + (1 - a) (\mathbb{1}\mathbb{1}^T - \mathbf{I}). \quad (6.9)$$

The diagonal entries for  $A$  represent the  $n - 1$  interactions weighted  $a$  and the  $\binom{n}{2} - (n - 1)$  non-interactions weighted 1. Equation (6.9) can be simplified to

Using the definitions for  $H$  (6.8) and  $A$ (6.9) gives

$$H = (1 - a) \left( -(n - 1)I + \mathbb{1}\mathbb{1}^T - I \right).$$

From (6.7) and the definition for the Laplacian matrix for a complete graph (6.6) gives weight  $w = (1 - a)$ ; the results are consistent with linear average consensus algorithms, which obey (4.3). Similar work is done over random graphs by Hatano and Mesbahi [47] and Tahbaz-Salehi and Jadbabaie [114]. Because we assume that our system is well mixed, the expected value of our network is complete; we find the Laplacian for a complete graph in our derivation. ■

As above, the dynamics of the estimator, without control, is equivalent the Laplacian matrix for a complete graph. From here, we analyze the equilibrium behavior for the estimator and controller.

### 6.6.2 Simultaneous Estimation and Control Stability

To combine the estimator and controller and ensure stability, we must adjust the controller. The controller stability in the previous section is dependent upon the estimates  $\hat{x}_i$  being equal to the actual  $x$ .

Since the space of  $\hat{x}$  is continuous, this will never occur. Thus, we alter our controller so that rate  $K_i$  becomes 0 when  $\hat{x} \approx r$ :

$$K_i = \begin{cases} 0 & \text{if } r - \epsilon \leq x_i \leq r + \epsilon, \forall i; \\ |q_i - r||x_i - r| & \text{otherwise.} \end{cases} \quad (6.10)$$

Note that  $\epsilon < \frac{1}{2n}$  should be chosen to be sufficiently small to preserve the conservation property (6.3).

We introduce the notation and theorem to prove that our controllers and estimators can be designed separately but work together. The system is modeled as a continuous-time

Markov Process over the compact state space  $Z$  and examine its embedded discrete-time Markov chain. The set  $\mathcal{S} \subset Z$  is measurable and invariant,  $T_{\mathcal{S}}$  is the hitting time of  $\mathcal{S}$ , and  $P(\cdot)$  indicates probability. We write  $\sigma = z_0 z_1 z_2 \dots z_n$  as a *path* of length  $n$ , or series of  $n$  states, that begins at state  $z_0$  and ends at  $z_n$ .  $K_{z_i z_j}$  is the transition rate from state  $z_i$  to  $z_j$ .

We introduce a theorem that proves that if for all states there exists a finite path to an invariant set, the probability of paths that eventually reach the invariant set  $\mathcal{S}$  is 1, provided that the sum of the transition rates out of each state is bounded. The following theorem applies to discrete-time infinite-state Markov Processes. This theorem is used later in a proof of convergence.

**Theorem 6** Assume  $K_{z_i z_{i+1}} > \epsilon > 0$  and  $\sum_{x \neq y} K_{xy} < K_{max}$ . If  $\forall z \in Z$  there exists a path starting at  $z$  and ending in  $\mathcal{S}$ , whose length is  $n \leq N < \infty$ , then  $P(T_{\mathcal{S}} < \infty) = 1$ .

*Proof of Theorem 6.* Consider a path of length  $N$ ,  
 $\sigma = z_i z_{i+1} \dots z_{i+N}$ . By the assumptions of Theorem 6,

$$P(z_{i+N} \in \mathcal{S} | z_i \notin \mathcal{S}) \geq \delta = \left( \frac{\epsilon}{K_{max}} \right)^N. \quad (6.11)$$

Thus, the probability of its complement must be

$$P(z_{i+N} \notin \mathcal{S} | z_i \notin \mathcal{S}) \leq 1 - \delta. \quad (6.12)$$

We can derive the probability  $P(z_{i+mN} \notin \mathcal{S})$  using conditional statements

$$P(z_{i+mN} \notin \mathcal{S} | z_{i+(m-1)N} \notin \mathcal{S})P(z_{i+(m-1)N} \notin \mathcal{S}) + P(z_{i+mN} \notin \mathcal{S} | z_{i+(m-1)N} \in \mathcal{S})P(z_{i+(m-1)N} \in \mathcal{S}). \quad (6.13)$$

Note that  $P(z_{i+mN} \notin \mathcal{S} | z_{i+(m-1)N} \in \mathcal{S})$  in (6.13) equals zero since the system cannot leave  $\mathcal{S}$ .

By iterating this substitution, we arrive at

$$P(z_{i+mN} \notin \mathcal{S}) = \prod_{j=1}^m P(z_{i+jN} \notin \mathcal{S} | z_{i+(j-1)N} \notin \mathcal{S})P(z_i \notin \mathcal{S}).$$

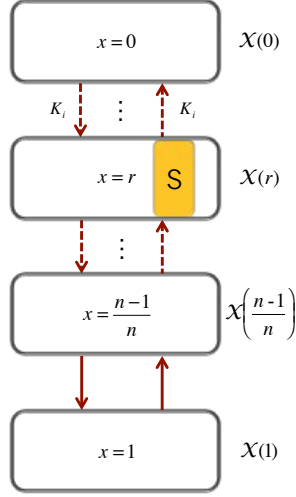


Figure 6.2: State space equivalence classes called levels. Levels  $\chi$  are labeled with corresponding  $x$ . The state where  $x = r$  includes the sink set  $\mathcal{S}$ . Dashed arrows and ellipses indicate omitted states.

Substituting (6.12) gives

$$P(z_{i+mN} \notin \mathcal{S}) < (1 - \delta)^m P(z_i \notin \mathcal{S}).$$

Our condition  $P(T_{\mathcal{S}} < \infty)$  is equivalent to

$$\begin{aligned} & 1 - \lim_{m \rightarrow \infty} P(T_{\mathcal{S}} > mN) \\ & \geq 1 - \lim_{m \rightarrow \infty} \sum_{z \in Z} (1 - \delta)^m P(z_i = z) \\ & \geq 1 - 0 \\ & = 1. \end{aligned} \quad \blacksquare$$

We divide the state space  $X \times Q$  into equivalence classes where  $x$  is constant in each class. We call these classes *levels* and define them as  $\chi(x) = \{(\hat{x}, q) \mid \frac{1}{n} q^T \mathbf{1} = x\}$ . That is, the states in each level have the same number of agents in state 1; these states are permutations of one another. Refer to Figure 6.2 for a graphical description. We prove that there is an invariant  $\mathcal{S}$  in level  $\chi(r)$ , where  $q^T \mathbf{1} = rn$  and  $\hat{x}_i \in [r - \epsilon, r + \epsilon] \forall i$ .

Using the following lemmas with proofs outlined below, we show that there exists an

invariant set  $\mathcal{S}$ , where  $\mathcal{S} = \{(\hat{x}, q) | r - \epsilon \leq \hat{x}_i \leq r + \epsilon, \forall i = 1, \dots, n\}$  for  $\epsilon < \frac{1}{2n}$ .

**Lemma 1**  *$\mathcal{S}$  is invariant with respect to the estimator and controller actions. In particular,*

(a) *No estimation event takes an agent out of  $\mathcal{S}$ .*

(b) *No control event takes an agent out of  $\mathcal{S}$ .*

### 6.6.3 Proof of Lemma 1

For any robots  $i$  and  $j$ , estimates  $\hat{x}_i$  and  $\hat{x}_j$  are updated by the convex combination in (6.1). Without loss of generality, assume  $\hat{x}_i < \hat{x}_j$ . Then

$$\hat{x}_i \leq \hat{x}'_i \leq \hat{x}'_j \leq \hat{x}_j, \quad (6.14)$$

where  $\hat{x}'_i$  and  $\hat{x}'_j$  are the updated estimate values. No estimation event can take the agents' estimates outside of any level  $\chi(x)$ . In particular,  $\mathcal{S} \in \chi(r)$  is invariant with respect to estimation events.

Similarly, by construction, no control event can take an agent's state out of  $\mathcal{S}$ . Refer to the definition of the modified controller (6.10). ■

**Lemma 2** *The rate  $K_i = 0$  for all  $i$  only when  $(\hat{x}_i, q_i) \in \mathcal{S}$ .*

### 6.6.4 Proof of Lemma 2

All invariant states are in the sink set  $\mathcal{S}$ . We prove this lemma by contradiction and show that if  $\hat{x} \in [r - \epsilon, r + \epsilon] \forall i$  then  $x = r$ .

Assume that  $\hat{x} \in [r - \epsilon, r + \epsilon] \forall i$  but  $x \neq r$ . Thus, the sum  $\sum_i \hat{x}_i \in [n(r - \epsilon), n(r + \epsilon)]$ . We note by definitions (something) and (6.3) that  $x = \frac{1}{n} \sum_i \hat{x}_i$  and  $x \in [r - \epsilon, r + \epsilon]$ . The values  $x$  and  $r$  can be written as fractions:  $x = \frac{j}{n}$  and  $r = \frac{k}{n}$  where  $j, k \in \{0, 1, \dots, n\}$ . Since  $\epsilon$  can be no larger than  $\frac{1}{2n}$ ,  $j \in [k - \frac{1}{2}, k + \frac{1}{2}]$ . There is only one integer in that interval, so  $j = k$  and  $x = r$ , which is a contradiction.

Thus, if  $\hat{x} \in [r - \epsilon, r + \epsilon] \forall i$ , then  $K_i = 0 \forall i$  robots and the agents estimates are within the invariant set  $\mathcal{S}$ . ■

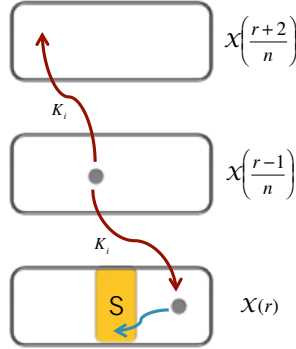


Figure 6.3: From any state not in the sink set, there exist control events (blue arrows) enabled that occur at rate  $K_i$ . Control events bring the state of the system to  $\chi(r)$ . When in the desired level  $\chi(r)$  estimation events (yellow arrows) move  $\hat{x}$  to the sink set  $\mathcal{S}$ .

**Lemma 3**  $\mathcal{S}$  is reachable from any initial condition in finite time.

- a. From (within) any level, there is a path of  $m_c$  steps to  $\chi(r)$ .
- b. From any location in  $\chi(r)$ , there is a path of  $m_e$  steps to  $\mathcal{S}$ .

In other words,  $\mathcal{S}$  is globally attracting.

### 6.6.5 Proof of Lemma 3

Using the results of Lemmas 1 and 2 and Theorem 6 we prove Lemma 3, that our system always reaches a desired state in finite time. We use Lemma 1 to prove that the set  $\mathcal{S}$  is invariant and Lemma 2 to prove that the set  $\mathcal{S}$  appears in only one level.

From Theorem 6, to show that the probability that any infinite path will end up in  $\mathcal{S}$  is 1, it is sufficient to construct a finite path from any state  $(\hat{x}, q)$  to  $\mathcal{S}$  provided bounded transition rates. First, we find a bound,  $K_{max}$ , on the sum of transition rates out of any state by summing the controller and estimator rates.

$$K_{max} > \sum_{i=1}^n |q_i - r| |\hat{x}_i - r| + \binom{n}{2} k.$$

Each control rate  $|q_i - r| |\hat{x}_i - r| < 2$  and the rate  $k$  is measured from the physical testbed, [26]. Thus,  $K_{max} = 2n + \binom{n}{2} k$ .

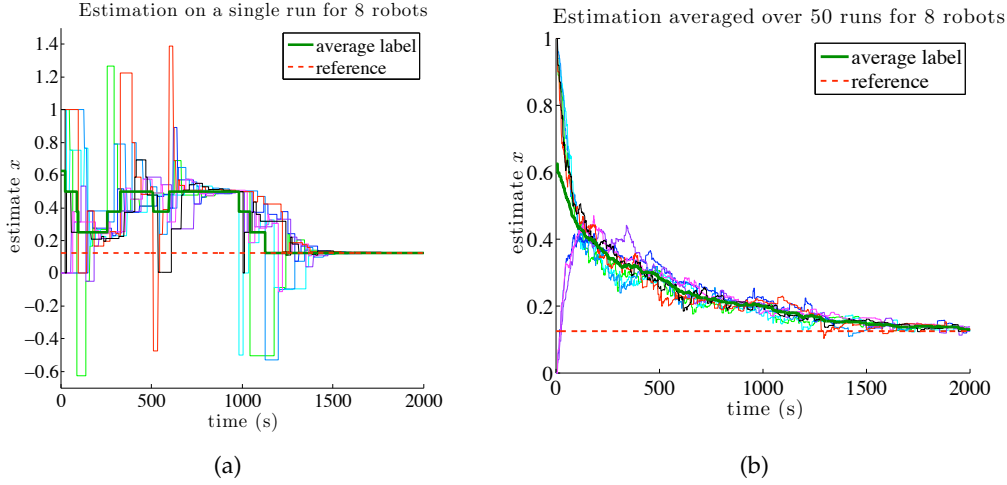


Figure 6.4: Demonstration of the FFC estimator and controller working together. (a) Estimates for a single run. Average over 50 simulations for 8 robots. In both (a) and (b), thin lines represent data for single robots and thick green lines represent the average label. In all three figures, the dashed red lines represent the reference 0.125. (c) Normalized histogram of estimates for 50 runs.

Second, we construct a path from any point in any level to the set  $\mathcal{S}$  using  $m_c$  control actions and  $m_e$  estimation actions. From any level, the minimal path length to level  $\chi(r)$  via control actions is  $m_c = |rn - q^T \mathbf{1}|$ . Similarly, from any location in  $\chi(r)$ , a path can be constructed in  $m_e$  steps via estimation events to  $\mathcal{S}$ . We choose two agents with maximal difference to update their estimates. Since  $\mathcal{S}$  is measurable and estimation events are contracting (6.14), then estimates  $\hat{x}_i$  move closer to each other and closer to  $r$  for all  $i$ . Therefore, all  $\hat{x}_i$  must be within  $\epsilon$  of  $r$  at a finite time thereby reaching  $\mathcal{S}$  in a finite number of steps. ■

## 6.7 Simulation Results

We demonstrate the estimator and controller working together by directly simulating the system using the *Stochastic Simulation Algorithm* (SSA) [42]. Figure 6.4 shows the behavior for 8 robots in a single run and averaged over 50 runs. In Figure 6.4(a), note that all estimates converge to the reference with zero variance. When averaged over many runs as in Figure 6.4(b) we see asymptotic convergence. The series of histograms in Figure 6.5

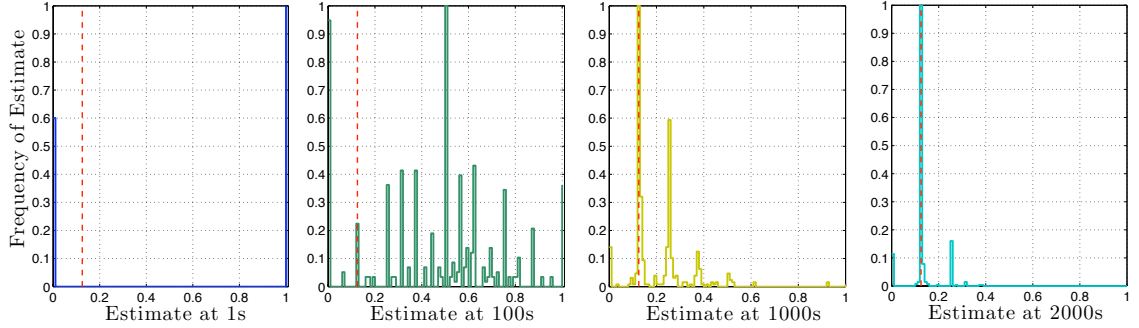


Figure 6.5: Normalized histogram data for FFC at various times. The histogram at the far left shows the initial distribution of estimates where 5 of 8 robots start with discrete state 1. The histogram on the right shows the distribution of estimates at 2000s, with a spike about the reference 0.125.

shows the evolution of the distribution over time. For these simulations the parameter  $\epsilon$  is chosen to be 0.05 for the controller and this fulfills the constraint that  $\epsilon < \frac{1}{2n}$ . The distribution of the estimates approaches the reference over time.

## 6.8 Application: Stochastic Factory Floor

We have thus far designed an estimator of a global value for a well-mixed topology but now we propose an estimator for a local value in a fixed topology. A fixed topology describes the the Stochastic Factory Floor (SFF) (Figure 6.6), is a modular multi-robot system that builds fixed lattice structures made up of raw materials called *nodes* and *trusses*. The hardware is based on the modular robot Ckbot [134] and is being developed at the University of Pennsylvania while the simulator has been developed at the University of Washington.

The SFF is composed of identical tiles each containing a robotic arm, a cradle, and an elevator as shown in 6.6(a). The robotic arms are modular and can pass nodes and trusses to one another. Nodes can also be placed into temporary storage locations, or cradles (not pictured). To create rigid structures, trusses actively connect to ridges on the nodes by using a motorized latch. Upon completion of a level, the elevator lifts connected components and construction of the next level begins.



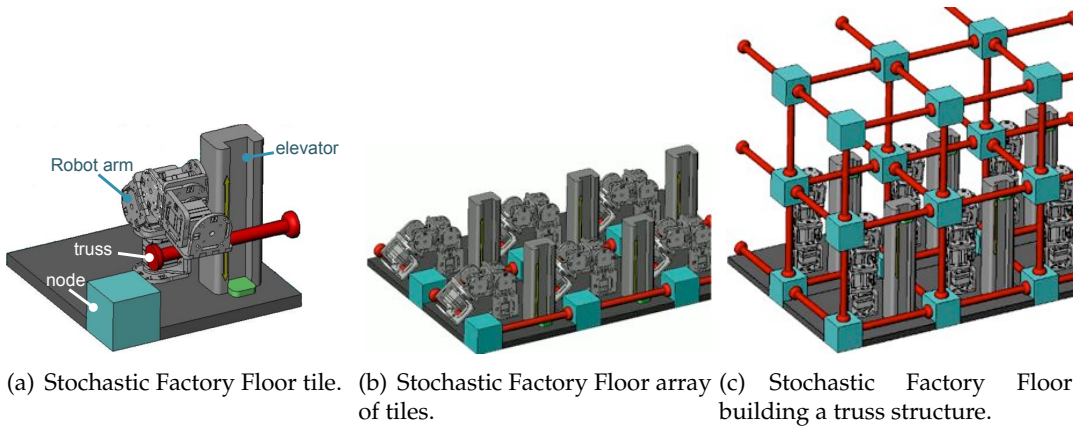


Figure 6.6: Stochastic Factory Floor (SFF) testbed.

For the SFF to function, we develop general principles for distributed assembly and can apply our estimator and controller to several scenarios. One of the coordination problems in this testbed involves lifting a completed level. All of the tiles must have the requisite nodes and trusses in the correct locations before they can decide to lift their elevators synchronously. Other challenges include dynamic routing of parts, automatic program generation, and building arbitrary structures.

An estimator has been designed for the Programmable Parts Testbed (PPT) and is directly applicable to the Swarmbot testbed. However, the algorithm requires some adapting to be applicable to the Stochastic Factory Floor (SFF). Specifically, in the SFF, the *population fraction* refers to the fraction of the nodes in the system. Later, we can also estimate the fraction trusses. However, it may not be enough to simply estimate the population fraction of the entire system; we may want the structure to be built in a particular location. Thus, we introduce a *local* population fraction, the fraction of nodes in a particular location.

In the example in Fig 6.7 each square represents a robot that can interact with its four neighbors (left, right, down, up). The allowable interactions for each tile consist of propagating a message and passing a node to any neighbor.

We set up a similar formalism as in Ch. 5 Consider a set of  $n$  tiles each having a discrete internal state,  $q_i(t) \in \{0, 1\}$ , which represents the presence of a node, depicted a black square in Figure 6.7(a). Also consider a subset of  $m$  tiles that represent an area of interest.

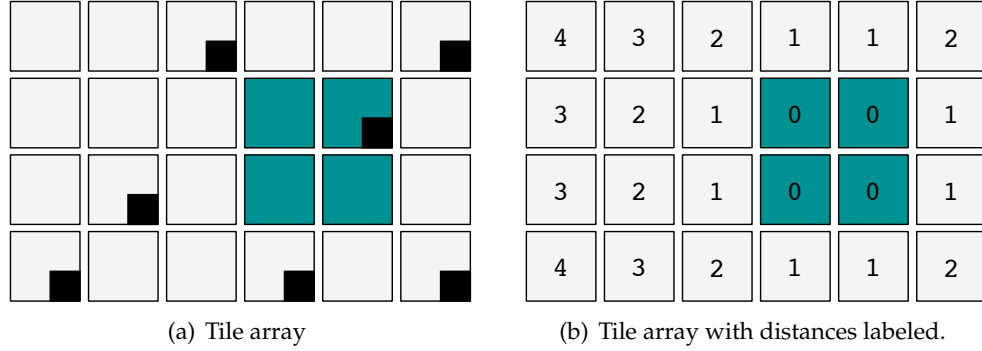


Figure 6.7: Array of robotic tiles. Tiles can communicate with and pass nodes to neighbors. The teal region indicates an area of interest. a) Small black squares indicate that a node is present. b) Tiles are labeled with distance from desirable region.

The population fraction for this area of interest is the fraction of tiles that have nodes.

In Figure 6.7(a), the population fraction of nodes in the area of interest is  $\frac{1}{4}$ . We can achieve this value by initializing the estimates  $\hat{x}_i$  according to the presence of the node and averaging only within the teal square. The estimate update rule for neighbors  $i$  and  $j$  is given by (6.1).

The difficulty lies when we move outside the teal square. It is clear that we can average within the desired region, but how does this information get propagated outward? To address this problem, we create a directed graph rooted in the desirable region and propagate messages away from that region. Thus, we assign 0 to all tiles in the desirable region and label children according to their distance. This is illustrated in Figure 6.7(b).

We set up a similar formalism as in Ch. 5 and earlier in Ch. 6. Consider a set of  $n$  tiles each having a discrete internal state,  $q_i(t) \in \{0, 1\}$ , which represents the presence of a node, depicted a black square in Figure 6.7(a). Also consider a subset of  $m$  tiles that represent an area of interest. The *population fraction* for this area of interest is the fraction of tiles that have nodes.

In Figure 6.7(a), the population fraction of nodes in the area of interest is  $\frac{1}{4}$ . We can achieve this value by initializing the estimates  $\hat{x}_i$  according to the presence of the node and averaging only within the teal square. The estimate update rule for neighbors  $i$  and  $j$  is given by (6.1)

Thus far, we use the same update rule as in [108]. The difficulty lies when we move outside the teal square. It is clear that we can average within the desired region, but how does this information get propagated outward? To address this problem, we create a directed graph rooted in the desirable region and propagate messages away from that region. Thus, we assign 0 to all tiles in the desirable region and label children according to their distance. This is illustrated in Figure 6.7(b).

### 6.8.1 Simulated Results

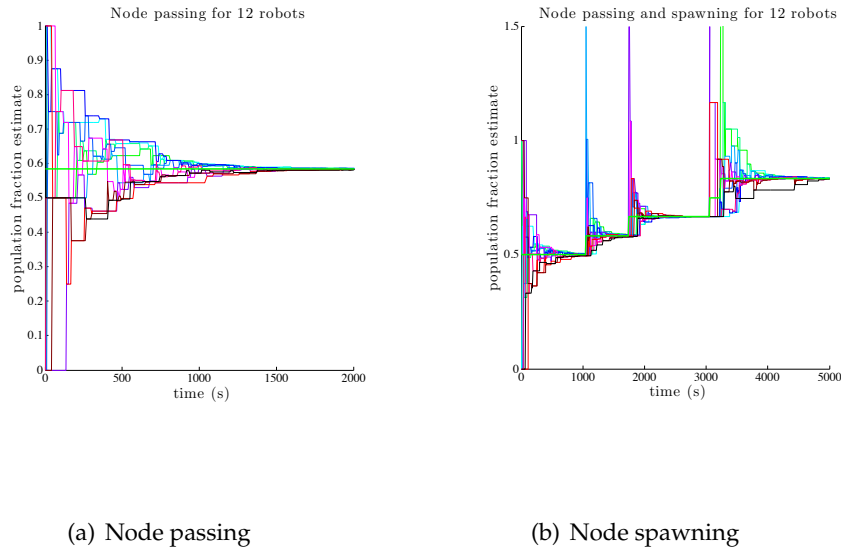


Figure 6.8: Simulations of estimation on fixed topology tile space. a) Nodes are passed between tiles (fixed number of node). b) Nodes spawn from one side and are passed (varying number of nodes).

Figure 6.8 shows the FFC applied directly to the tile testbed with estimation performed on the population fraction of nodes in the system. In Figure 6.8(a) nodes are being passed and the agents converge upon a the ratio. In Figure 6.8(b) nodes are passed around and appear in source locations. The spikes indicate the feed-forward term accounting for the appearance of a new node.

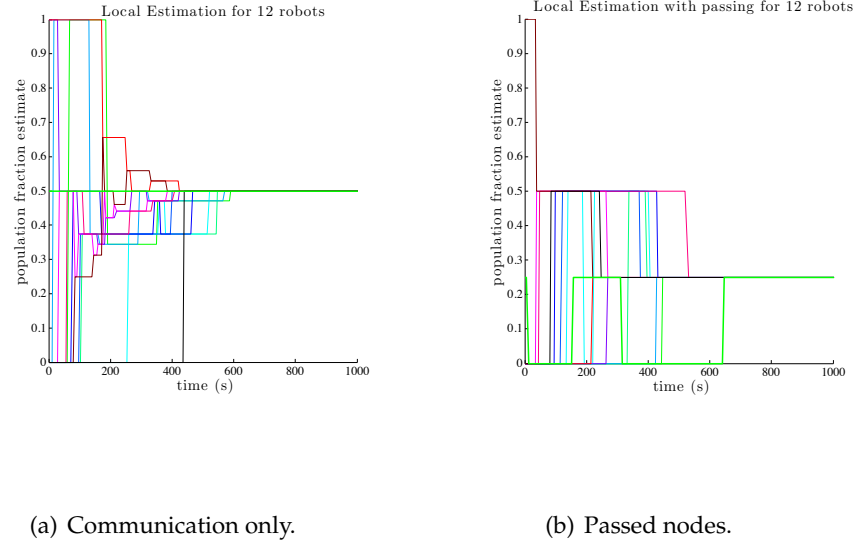


Figure 6.9: Local estimation for 12 tile elements. a) Local estimation with stationary nodes. b) Local estimation with passed nodes.

Figure 6.9 shows simulations for local estimation of nodes in a target region. Earlier, estimation was performed for the whole set of tiles and now, there is estimation for only a subset of the tiles. In 6.9(a), the agents talk to one another only, while in 6.9(b), nodes are passed amongst the tiles. Both come to agreement of the population fraction of the nodes in the particular area. This needs to be extended for nodes appearing at a source.

## 6.9 Discussion

Our scalar estimation algorithms IBC and FFC complement each other nicely. FFC converges with zero inter-agent variance, but with global error for any given execution, whereas IBC converges with bounded inter-agent variance but with zero average global error for a single execution. IBC is best used where the mean topology is completely connected. FFC is better suited than IBC for a fixed topology. Table 6.1 highlights the major similarities and differences.

	Input-based Consensus (IBC)	Feed-Forward Consensus (FFC)
Description	tracks population fraction convex combination of estimates and discrete states  finite variance convergence in probability	tracks population fraction convex combination of estimates; estimation updated during control event zero variance asymptotic convergence
Robust to:	unknown initial conditions dropped messages	unknown number of agents variable communication topology
Sensitive to:	unknown number of agents variable communication topology	unknown initial conditions dropped messages

Table 6.1: Comparison of IBC and FFC

## Chapter 7

### GRAPH CONSENSUS WITH WEARABLE DEVICES

#### 7.1 *Introduction*

Group coordination is required when independent agents set out to achieve a collective goal. Together, group members respond to disasters, rescue victims, and score points [12, 98]. One of the most basic coordination tasks is staying physically close and moving together from one point to another. Examples where people move as groups include schools and tours. Keeping the group together is challenging if the environment is crowded, cluttered, or noisy or if group members are numerous or active. These challenges are exacerbated when group members have special needs. In particular, children with autism are prone to stray [90, 91]; this behavior is among the most stressful for the caregivers [72]. Wandering is also a problem for people with dementia [65, 69, 70, 99].

Current technologies for keeping track of people include visual aids, physical restraints, and electronic tracking. In a school group taking a field trip, students may wear brightly colored shirts and group leaders count them. However, this puts significant cognitive load on the group leader. A small child may be tethered to its parent using a harness [27], but use of harnesses is controversial [15, 45]. Electronic devices seem promising for tracking people [99], but are often reactive. The Pervasive and UbiComp communities are uniquely situated to ease the stress of caregivers and group leaders while providing active monitoring.

We present Grouper, a decentralized system of wearable modules, as a technological aid to this problem. Grouper allows groups to stay together without physical constraints and aims to reduce the cognitive load required by all members of the group. Each module estimates the proximity of the group members and alerts group members if at least one is too far away. The alert continues until the wanderer returns to the group. Users need not continuously consult their devices to check if the group is together, but will be clearly

alerted when straying from the group.

Grouper modules estimate proximity using received signal strength indicator (RSSI) of packets sent by wireless radio. Technical challenges included RSSI noise, packet collision, and concurrent operation. We characterized the behavior of RSSI in three types of environments and used this information to design a decentralized estimation algorithm. We then tested Grouper on groups of five in real-world environments, and characterized the fidelity of the alerts by examining ground truth data. The main contributions of this work are 1) a decentralized system for estimating the graph of a highly dynamic set of nodes, 2) characterization of proximity sensing using RSSI for three types of environments, and 3) evaluation of the system in a real-world environment.

We present our algorithms in Sec. 7.3. The system is modeled in Sec. 7.4 and simulated in Sec. 7.5. The Grouper hardware is described in Ch. 2 and characterized in Sec. 7.6. Sec. 7.7 is a discussion of experiments and results, and we conclude in Sec. 7.8.

## 7.2 Problem Statement

We represent our system as a graph to model proximity. A graph consists of nodes and edges and in this paper we use nodes to represent users or their modules, while edges represent proximity or communication range. Two nodes are *connected* (1) if there exists an edge between them and they are *disconnected* (0) otherwise. A graph is connected if any node can reach any other node via edges.

There are several ways to keep track of members in a group: one might choose a centralized or decentralized method. In a centralized approach, the leader keeps track of the other group members. The group is not together if any user is disconnected from the leader. Thus, in a centralized approach, we define a group as the set of users within certain a distance from the leader. In a decentralized model, a node can communicate with any other node and the group is together if the graph is connected.

Previous work has focused on centralized algorithms between a leader and other users called followers [109]. For the centralized communication protocol, the leader broadcasts messages to followers and the group cohesion is determined by how far the followers are

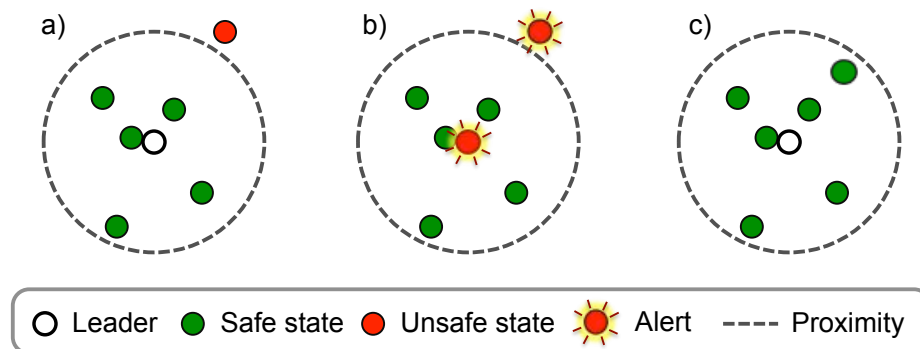


Figure 7.1: Centralized cluster-about-the-leader behavior. a) A follower wanders away from the leader (center, bold outline). b) The leader and the wandering follower are alerted. c) The follower comes back within distance of the leader.

from the leader. While a centralized approach is straight-forward, it may not sufficiently capture how a group actually behaves. First, groups are not always defined with a single leader or a leader at all. Second, it may not be practical to require users to be within a fixed radius from leader. For example, users may be arranged in a line where they are proximal to one another but too far from the leader. We address some of these concerns by designing a decentralized approach.

### 7.2.1 Centralized Approach

Our preliminary work focuses on centralized algorithms between a leader and followers [109]. For the centralized communication protocol, the leader broadcasts messages to followers. A follower will send a message back when it receives a message from the leader. Figure 7.1 shows the *cluster-about-the-leader* behavior. Followers are at varying distances from leader. The center dot is the leader and the dashed circles centered about the leader is the allowable distance. Followers that are within the allowable distance are colored green; those that are beyond the distance are colored red. Figure 7.1 a) shows a follower outside the acceptable distance, b) shows the agents' sensory cues, and c) shows the agent moved closer after receiving their sensory cues. Note that both the leader and follower receive sensory cues. Additionally, if a user wanders too far away from the group and is out of



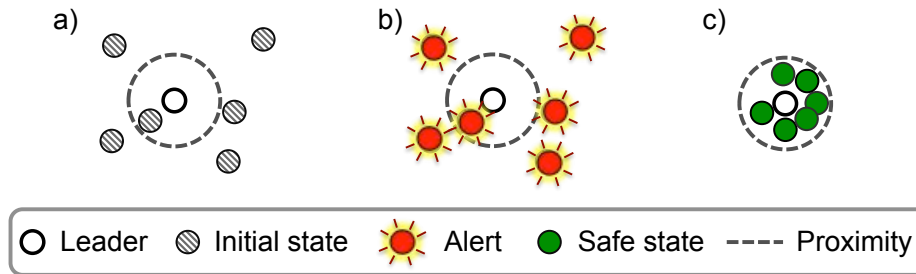


Figure 7.2: Centralized *global alert* behavior. a) All agents are initialized grey, except the leader who is indicated by a solid outline. b) The leader alerts all the followers regardless of distance. c) All followers within a desired radius have stopped alerting.

range for a certain amount of time, the leader also receives an alert.

The second centralized behavior is called *global alert*. The leader presses a button when he or she wishes to collect the group for an announcement or similar event. Figure 7.2 a) shows the followers with uninitialized state, b) shows all the agents getting an alert, and c) shows the followers close to the leader. The allowable distance is indicated by the dashed line. This behavior is useful if a teacher has an announcement or instructions for students.

### 7.2.2 Decentralized Approach

We are mainly interested in keeping track of a group of people in a distributed way and present an approach where nodes estimate the state of the system, represented as a graph. A group is together if the graph is connected. There are several challenges in the decentralized approach: *Who should be alerted? How is information passed?* The decentralized graph estimation problem is more complicated than the centralized problem, but better captures group behavior.

Figure 7.3 depicts the decentralized connected graph behavior. Nodes can move away from each other as long the graph is connected. We are interested in the proximity of each node and represent this with a dashed circle. Nodes that are connected to one another have a *graph edge* between them (solid line). In Figure 7.3 a) a node is outside the range of all other nodes. In Figure 7.3 b) all nodes receive an alert. Lastly, in Figure 7.3 c) the stray node moves within the other nodes' range.

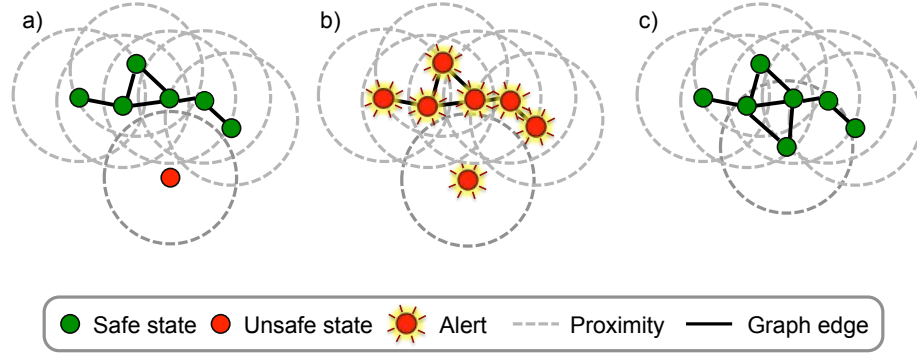


Figure 7.3: Decentralized *connected graph* behavior. a) *Undesired Behavior*: All nodes are connected to one another, except one (red). The red node is considered unsafe. b) *Sensory Cue*: All the nodes receive a signal that a node is missing. c) *Desired Behavior*: The graph is connected and all nodes are safe (green).

### 7.3 Algorithms

In the decentralized approach in Sec. 7.2, users are disconnected from the group when the graph is disconnected. Each node constructs a graph estimate consisting of nodes and edges that is based on the estimates of other nodes. Nodes cannot simple average graphs and instead weigh information based on perspective. Below is an example.

**Example:** Given a graph with four nodes shown in Table 7.1. The list of edges for the graph are given by  $E = \{e_{12}, e_{23}, e_{34}\}$ . The edges from the perspective of each agent  $i$  is given by  $E_i$ . Agents use local information and only receive information from those to whom they are connected. For instance, agent 1 is connected only to agent 2 so does not know (directly) about agent 3.

Graph	Edge list
	$E = \{e_{12}, e_{23}, e_{34}\}$
	$E_1 = \{e_{12}\}$
	$E_2 = \{e_{12}, e_{23}\}$
	$E_3 = \{e_{23}, e_{34}\}$
	$E_4 = \{e_{34}\}$

Table 7.1: Line graph for four nodes, pictorial representation and edge list. Edge list includes the perspective of each node.

Here, no individual agent has complete information about the graph, but the information from agents 2 and 3 together construct the entire graph. Suppose agent 2 sends its edge information  $E_2$  to agent 3. The union of the edges known by agents 2 and 3 is  $E_2(12), E_2(23), E_3(23), E_3(34)$ , where  $E_2(12)$  represents the edge  $e_{12}$  from the perspective of agent 2. Agent 3 processes the edges it has received from agent 2:

- $E_2(12)$  is a new edge for agent 3 and is concatenated to agent 3's edge list.
- $E_2(23)$  is information about the edge shared by agents 2 and 3; agent 3 averages it with its own information  $E_3(23)$ .

The edge list for agent 3 at the end of this interaction is  $E_3 = \{e_{12}, e_{23}, e_{34}\}$ . This algorithm differs from the scalar estimation in that the perspective is taken into account when incorporating information. Each agent cannot sense the entire graph so builds an estimate that is dependent on the communication of other agents. Algorithms 1, 2, and 3 show the example generalized for arbitrary graphs.

Algorithm 1 summarizes the initialization of a node's graph estimate. Each node  $i$  constructs a list of nodes  $N_i$  that includes all nodes that it detects. Node  $i$  then constructs a list of edges  $E_i$  for itself and every other node in  $N_i$ . An edge is represented as a tuple  $E_i(ij) = (i, j, rssi_{ij}, age_{ij})$ , where  $i$  and  $j$  are the nodes,  $rssi_{ij}$  is the RSSI of the most recent packet and  $age_{ij}$  represents the newness of the information. The edge  $E_i(ij)$  can be read as "the edge between nodes  $i$  and  $j$ , from the perspective of node  $j$ ".

---

**Algorithm 1** Graph initialization for node  $i$ .

---

Construct list of nodes  $N_i$ .

$$N_i = \{i, j, k, l, \dots\} \text{ where } j, k, l, \dots \text{ are detected nodes}$$

Construct list of edge data  $E_i$ .

$$\text{where edge data } E_i(ij) = (i, j, rssi_{ij}, age_{ij}) \forall j \in N_i, i \neq j$$


---

After initialization, a node will send and receive edge data to complete its graph estimate. Node  $i$  receives a packet  $E_j$  from node  $j$  with packet RSSI  $p_{rssi}$ . This is not to be

confused with the RSSI measurements contained in the payload of the packet. Node  $i$  parses the data from node  $j$  according to Alg. 2. For each edge  $e$  in a packet  $E_j$  there are four basic operations: average, ignore, update, and concatenate. For distinct nodes  $i, j, k, l$ :

- if the receiver  $i$  and sender  $j$  are part of the edge data ( $e = E_j(ij)$ ), then node  $i$  *averages* the packet RSSI  $p_{rssi}$  with the payload RSSI  $E_j(ij)$ . The *age* of the edge is also reset to 0.
- if the receiver  $i$  is part of the edge data, but the sender  $j$  is not ( $e = E_j(ik)$ ) then the receiver *ignores* the edge data; the receiver has better information than the sender.
- if the receiver  $i$  is not part of the edge ( $e = E_j(jk)$ ), then it *updates* its graph according to the packet.
- if the edge is not in the edge data of  $i$  ( $e = E_j(jl) \notin E_i$ ), the edge is new and the receiver *concatenates* the edge to its list. This allows for dynamic network resizing.

Algorithm 2 provides unequal weights for different information based on perspective. When a node has a “first-person” perspective of an edge, it averages or ignores the received data. When a node has a “third-person” perspective of an edge, it updates or concatenates the received data.

Each node then evaluates if an edge is present by thresholding it via RSSI and age. Algorithm 3 describes this threshold. Variable *isedge* represents if there is an edge present (1) or not (0). An edge is not present when its RSSI is too low (module is too far away) or if the information is too old (module has not communicated recently). This threshold is state-dependent to take into account recent history. The variable *age* is incremented every second until it reaches *timeout*, at which time there is no edge and *isedge* is set to 0.

Our algorithms used for distributed estimation of the network allows for nodes to dynamically join the network. A list of nodes does not need to be preprogrammed, nodes can join at different times, and the network can vary in size. One of the challenges of designing an embedded system is designing modular systems to operate concurrently. Another implementation challenge is scaling an increasing number of nodes leads to larger packet size

---

**Algorithm 2** Packet parsing. Node  $i$  receives edge data from node  $j$ .

---

Given distinct nodes  $i, j, k, l$ .

Edge data  $E_j$  received from node  $j$  with packet RSSI  $p_{rssi}$ .

**for all** edges  $e \in E_j$  **do**

**if**  $e = E_j(ij)$  **then**

$E_i(ij) \leftarrow \text{average}(p_{rssi}, E_j(rssi_{ij}))$

    reset  $age \leftarrow 0$

**else if**  $e = E_j(ik)$ , where  $k \neq j$  **then**

    ignore  $e$

**else if**  $e = E_j(jk)$ , where  $k \neq i$  **then**

    update  $E_i(jk) \leftarrow E_j(jk)$

**else if**  $e = E_j(jl)$ , where  $E_i(jl) \notin E_i$  **then**

    concatenate  $E_i \leftarrow E_i(jl) \cup E_i$

**end if**

**end for**

---



---

**Algorithm 3** State-dependent edge threshold for node  $i$ .

---

Given edge  $e = (i, j, rssi_{ij}, age_{ij})$  with edge connectivity  $isedge_{ij}$

**if**  $isedge$  **and**  $rssi_{ij} < threshUp$  **and**  $age_{ij} < timeout$  **then**

$isedge_{ij} \leftarrow 1$

**else if**  $!isedge_{ij}$  **and**  $rssi_{ij} < threshDown$  **and**  $age_{ij} < timeout$  **then**

$isedge_{ij} \leftarrow 1$

**else**

$isedge_{ij} \leftarrow 0$

**end if**

---

and packet collision. In the next section, we characterize the proximity sensing to address some of the implementation challenges.

#### 7.4 Model

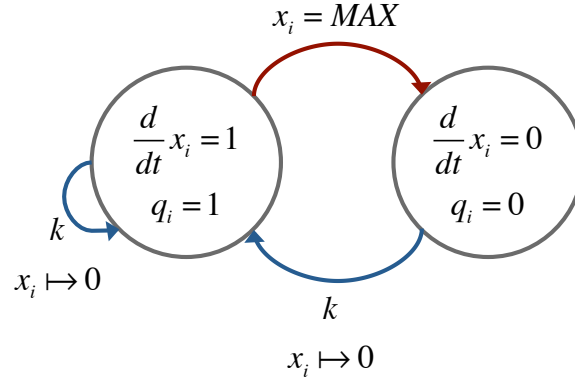


Figure 7.4: Model for edge  $i$ . Communication events (blue) occur at rate  $k$  and reset the variable  $x_i$ . The condition  $x_i = MAX$  (red) deletes an edge from a node's estimate.

We model Grouper as a SHS with deterministic and stochastic components. Figure 7.4 shows the model for edge  $i$  for a node. Each state is labeled by the discrete state and continuous dynamics, where a connected edge exists for  $q_i = 1$ . The continuous dynamics are:

$$\frac{d}{dt}x = \begin{cases} 1 & q_i = 1 \\ 0 & q_i = 0, \end{cases} \quad (7.1)$$

which are state dependent.

The age  $x_i$  of the edge is an abstraction for the measurement and age thresholding in Alg. 3. The continuous dynamics  $\frac{d}{dt}x_i$  represents the deterministic aging of the edge information. At rate  $k$ , an edge is updated when the node communicates with another and the age  $x_i$  is set to zero. The edge is determined to be disconnected when  $x_i = MAX$  (red arrow). This can be interpreted as the edge age reaching the timeout or the edge

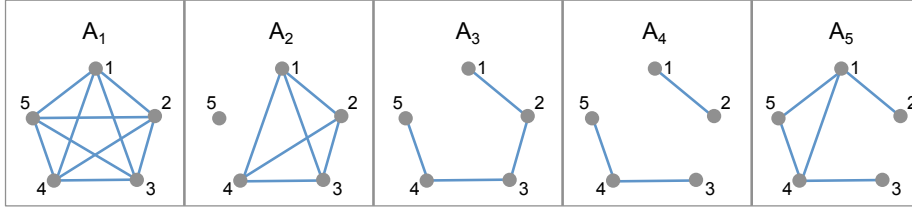


Figure 7.5: Five nodes changing topology, graph trajectory for Grouper simulation. The graphs  $\{A_1, A_2, A_3, A_4, A_5\}$  correspond to state changes at times  $t = \{0, 20, 40, 60, 80\}$ .

measurement reaching a threshold. Each edge can be modeled in this way and a node computes its connectivity by examining the edges.

### 7.5 Simulation

We simulate the system as modeled in Fig. 7.4 using the Stochastic Simulation Algorithm [42]. Edges are aged deterministically and communication between nodes occurs randomly as a function of the adjacency matrix. If there is no edge between nodes, then the communication rate is 0. Figure 7.5 shows the trajectory of graphs  $\{A_1, A_2, A_3, A_4, A_5\}$  for times  $t = \{0, 20, 40, 60, 80\}$ . The graphs alternate between connected and disconnected. The connectivity estimated by each node is shown in Fig. 7.6. In our simulation, the age timeout was 10 s with a rate of communication of  $k = 10$  Hz. In the simulation, nodes detect that they are disconnected about 10 s after state change due to the timeout.

### 7.6 System Characterization

We detect the proximity of users to each other based on the RSSI of packets sent between them. It is debated whether RSSI is a good measure of proximity [112, 16, 135]. Benkić *et al.* characterize RSSI indoors and conclude it to be a poor distance estimator while Srinivasan *et al.* assert that RSSI is underappreciated. We chose to use a RSSI because the XBee can then be used for communication and for sensing. Potentially, RSSI can be used both outdoors and indoors while GPS only works outdoors.

We provide a characterization of the RSSI using XBee for several environments: ideal outdoor (Fig. 7.7), non-ideal outdoor (Fig. 7.8), and indoor (Fig. 7.9). In an ideal envi-

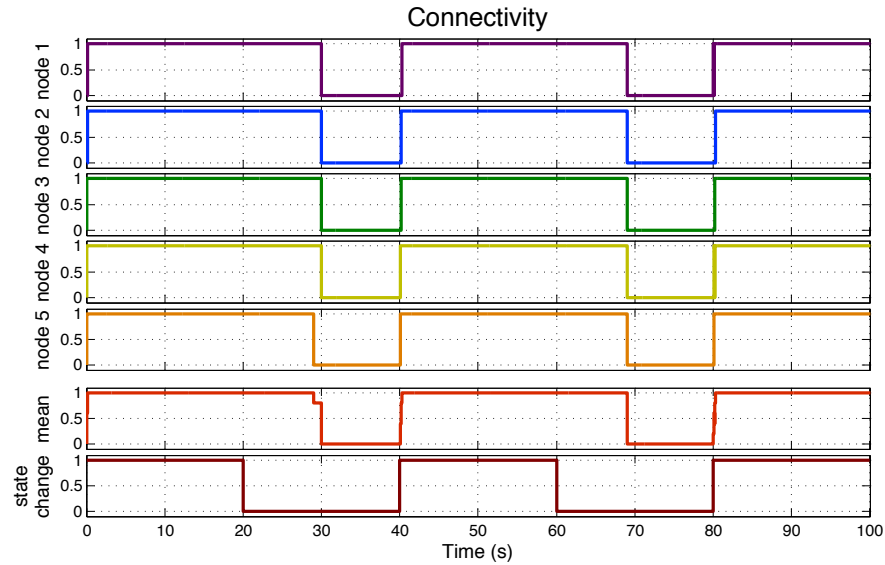


Figure 7.6: Connectivity shown for simulation with five nodes changing topology. Node data is shown in purple, blue, green, yellow, and orange; mean data is red and actual state change is shown in brown. Topologies are shown in Fig. 7.5 and alternate between connected (1) and disconnected (0).

ronment, the RSSI between nodes is inversely proportional to distance. In each of these environments, we set a pair of nodes at varying distances and sent messages at 10 Hz. One node was connected to a computer (serial output) to collect RSSI data. At least 200 samples were taken at each distance in each of the environments. At each distance, we plot the mean and standard deviation for RSSI and message frequency.

Figure 7.7 shows an ideal outdoor environment. The environment is considered ideal because data it was taken in an open are with no obstacles and minimal radio interference. The data is taken at a park and shows a monotonically decreasing relationship between RSSI and distance in Figure 7.7(a). As the nodes move farther apart, the standard deviation for received message frequency increases. There is only significant decrease in message frequency after 10 m.

Figure 7.8 shows RSSI data taken in a non-ideal outdoor setting. Data ranges from 0 m to 4.5 m taken at 0.5 m intervals. At our maximum measurement, 4.5 m, message frequency



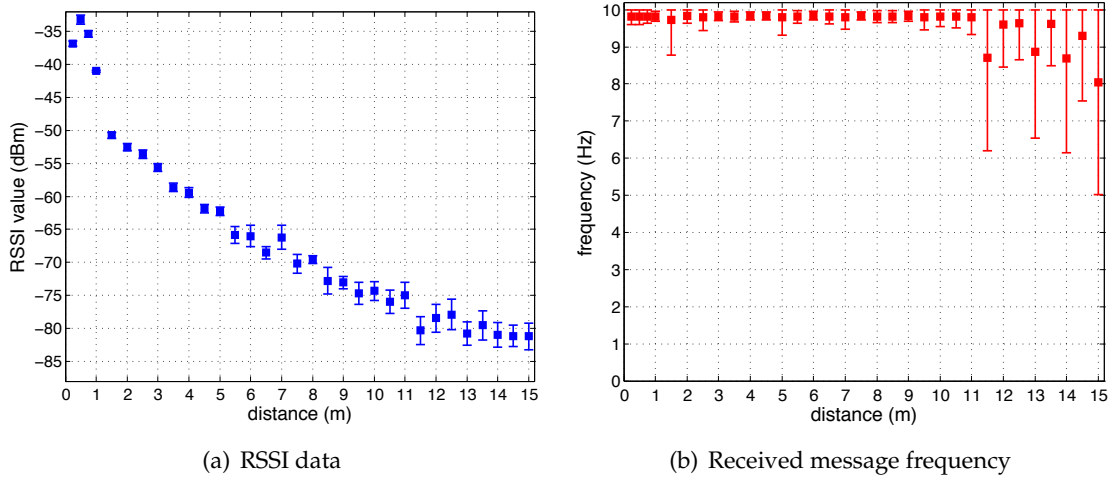


Figure 7.7: RSSI and message frequency for an ideal environment. Values are measured as a function of distance and mean measurements are indicated with standard deviation. This environment is considered ideal because the measurements decrease monotonically as a function of distance. Data was taken with nodes spaced up to 15 m apart. a) The RSSI decays with little variance as a function of distance. b) Message frequency was consistent with the broadcast frequency (10 Hz) until the nodes were spaced 11.5 m apart.

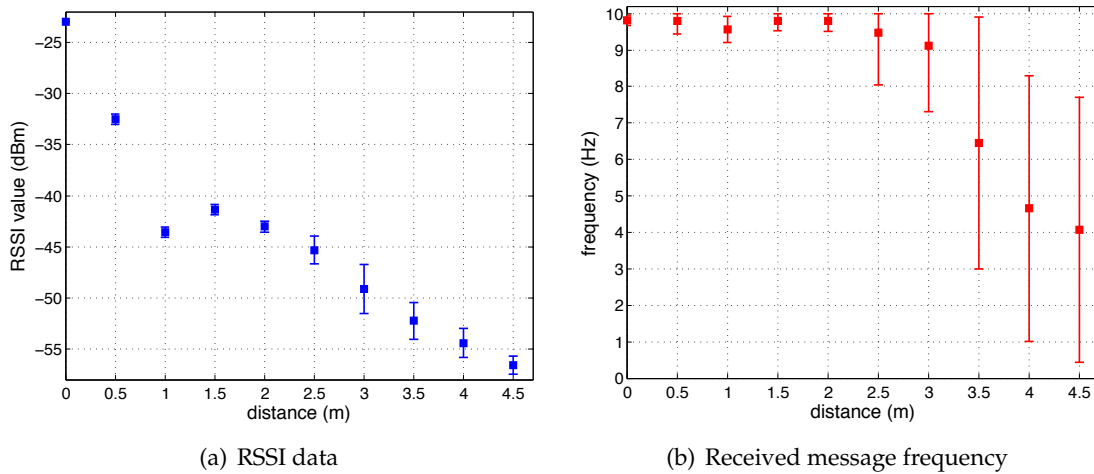


Figure 7.8: RSSI and message frequency for a non-ideal environment. Values are measured as a function of distance and mean measurements are indicated with standard deviation. This environment is considered non-ideal due to severe packet latency. Nodes were spaced up to 4.5 m apart. a) RSSI data decreased as a function of distance. b) Packet arrivals slowed significantly at 3.5 m. At 4.5 m message arrivals took as long as 35 s (0.03 Hz).

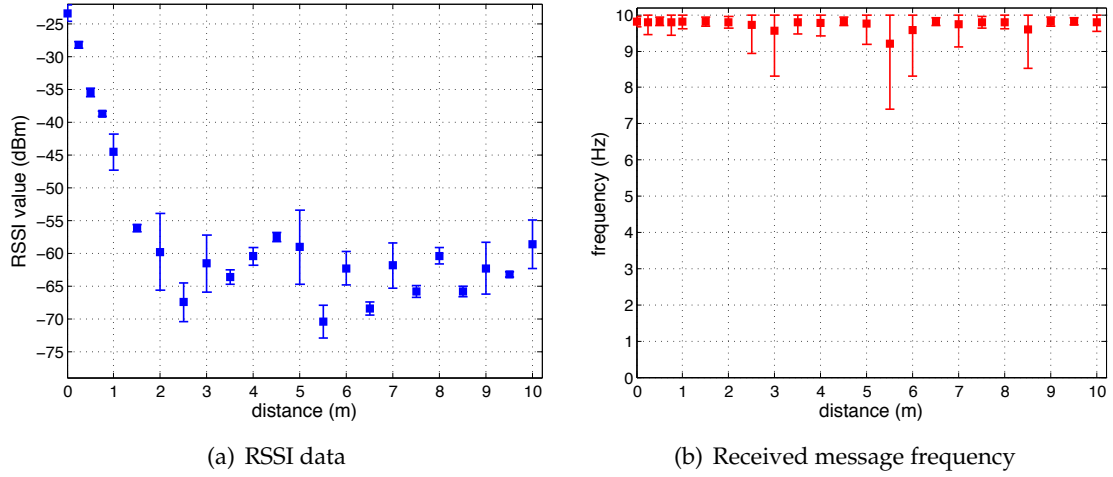


Figure 7.9: RSSI and message frequency for an indoor environment. Values are measured as a function of distance and mean measurements are indicated with standard deviation. Nodes were spaced up to 10 m apart, as limited by the room. a) The RSSI data does not have a monotonic relationship with distance, possibly due to messages bouncing off walls. b) Packets arrived at the broadcast frequency of 10 Hz and was fairly consistent for all distances

slowed significantly (0.03 Hz). User studies attempted in this area yielded poor results.

Figure 7.9 shows data collected indoors in a large room approximately 12 m  $\times$  6 m. The RSSI did not monotonically decay with distance, while received message frequency corresponded to broadcast frequency. Our RSSI measurements lead us to believe that RSSI may not be a reliable measure of distance indoors as others have concluded [16, 135].

## 7.7 Experiments

We evaluated Grouper by collecting ground truth data for the decentralized algorithm. We have designed our system to alert all members of a group when a single member of the group is far away, that is, the received packet is too old or the signal strength crosses below a state-dependent threshold.

The basic experiment was a group walk around the University of Washington campus for 30-60 minutes. Figure 7.10 depicts experiments on campus. Data was taken from five experiments with 23 participants in groups of five (with some repeat participants). Four

trial	notes
1	Exploratory walk, identified friendly environments (many obstacles)
2	Burke Gilman trail (moderate obstacles)
3	Red Square (no obstacles), Quad (many obstacles)
4	Husky Stadium (no obstacles)
5	CSE to Husky Stadium, Burk Gilman (many obstacles)

Table 7.2: Experiment details. Experiments are numbered 1-5 and detailed with the primary location on campus and density of obstacles (none, moderate, many).

participants donned the wearable modules and one held the logger. Video was taken for ground truth and the camera and logger are started simultaneously to synchronize data collection. Environments included areas with no obstacles (experiments 3 and 4), moderate obstacles (experiments 1, 2 and 5), and many obstacles (experiments 1, 3, and 5). Table 7.2 details the locations of the experiments and obstacle density.

### 7.7.1 Alert Classification

Grouper was designed to alert users when the group is not together. We tested this specification by examining the alerts and events. An *alert* is a sensory cue and an *event* is when at least one user is too far from the group. Our aim is to show that the system correctly alerts users, that is, 1) when there is an event, the system alerts users, and 2) all alerts are correct. We do this by classifying alerts:

*A true positive* is a correct alert when a group member is missing.

*A false positive* is an incorrect alert when everyone is present.

*A false negative* is the absence of an alert when a group member is missing.

A group is considered “together” when no two users are greater than approximately 6 m apart or that messages were no older than 10 s. For our experiments, we chose RSSI thresholds for Alg. 3 that corresponded to approximately 7.5 m if a user was leaving the group (*threshUp*) or 6 m if a user was returning to the group (*threshDown*). User study

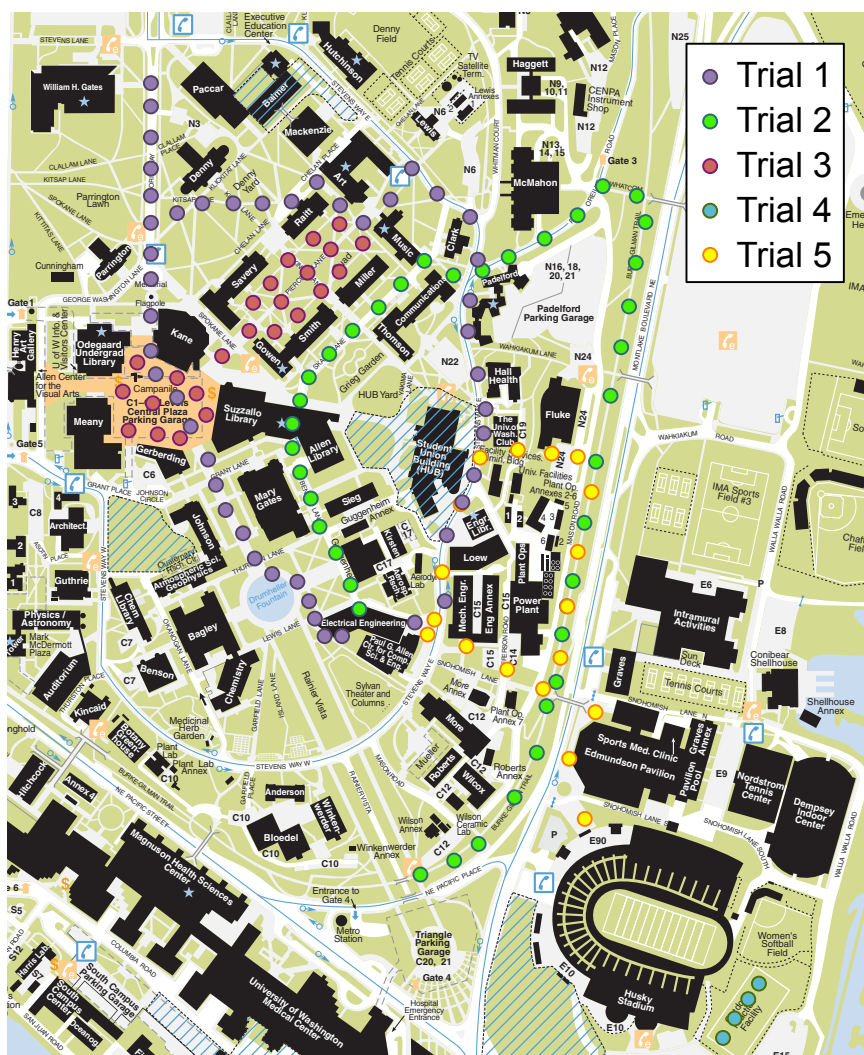


Figure 7.10: Summary of campus experiments. Experiments took place along paths on and the University of Washington campus.

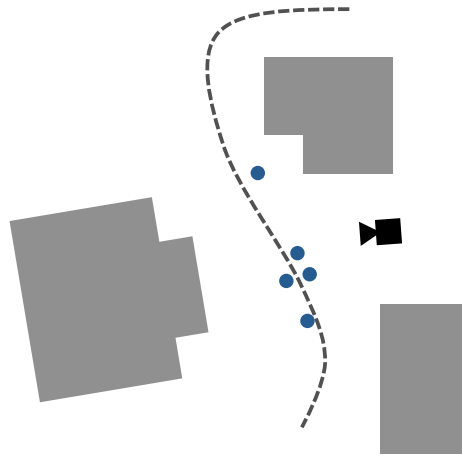


Figure 7.11: Experimental setup. Participants walked around campus as a group (circles) between buildings (blocks). Their path is indicated with the dashed line. Experiments were recorded with a video camera to provide ground truth and compared to logger data .

participants were asked to raise their hands when receiving an alert so that video could be synchronized with logger data. Figure 7.12 shows alerts and events classified in five experiments. The number of true positives includes correctly detected events for all users. The number of false positives includes every event where at least one module alerted errantly. Some false positives were caused by the environment and were noted if they occurred in a characterized non-ideal environment (Fig. 7.8). The number of alerts is the number of true positives added to the number of false positives. The number of events is the number of true positives added to the number of false negatives.

The experiments yielded fairly consistent results with 74% of all alerts correctly classified and 80% of all events detected. In Experiment 1, participants walked around campus to characterize environments for future experiments. There were numerous false positives, many of which were caused by known non-ideal environments.

Figure 7.13 shows connectivity estimated by five nodes, their mean, and ground truth for experiment 4. Experiment 4 was conducted on a football field where one user remained stationary while the rest of the group walked away. Nodes 1-4 are the wearable modules and node 5 is the logger. A node estimates the connectivity of the graph: the value 0 is a disconnected graph and value 1 is a connected graph. Note in the mean plot (red) that there

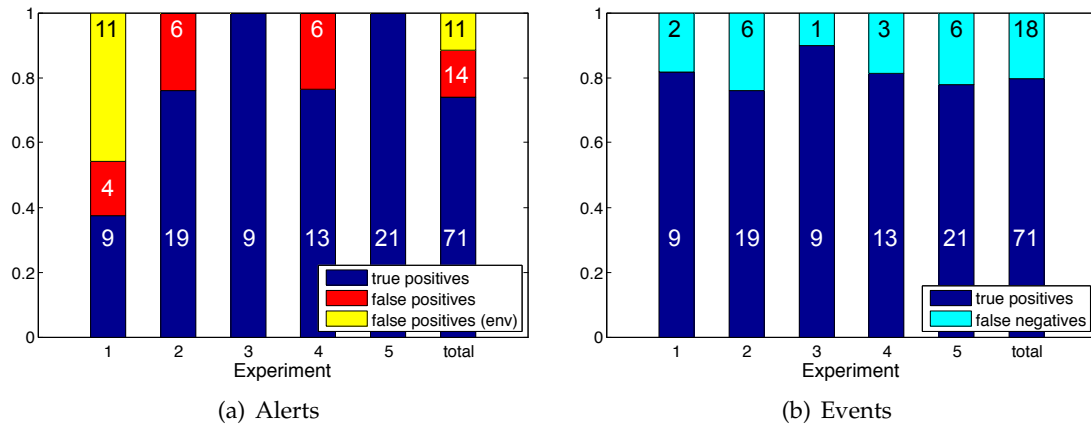


Figure 7.12: Data for five experiments with alerts and events classified. Bar charts are normalized and labeled with the number of occurrences. a) Alerts are comprised of true positives (navy) and false positives (red) including those due to the environment (yellow). b) Events are comprised of true positives (navy) and false negatives (cyan).

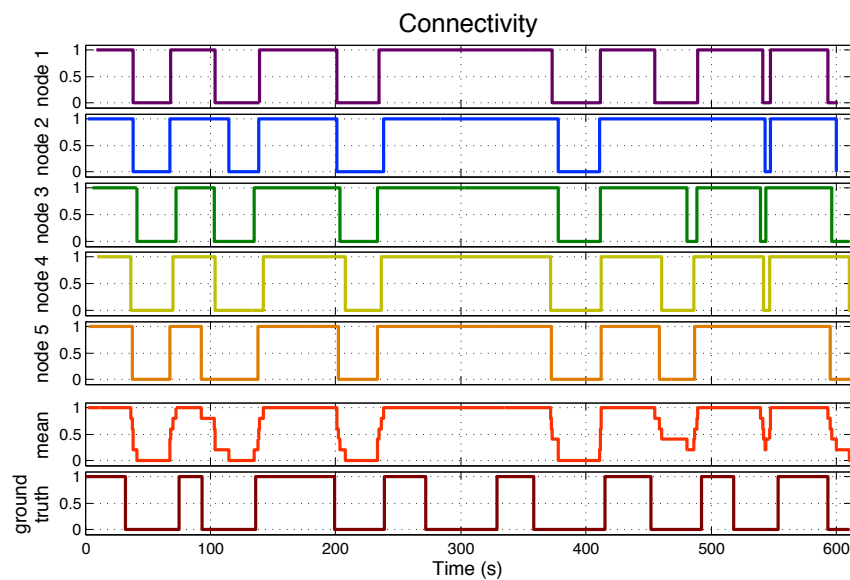


Figure 7.13: Connectivity estimated by each node, experiment 4. Node data is shown in purple, blue, green, yellow, and orange; mean data is red and ground truth is brown. Note that agents agree for majority of the experiment.

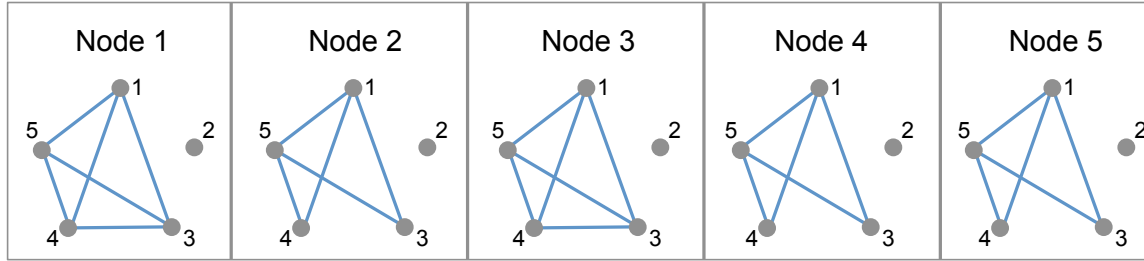


Figure 7.14: Graph estimates for each node, experiment 4 at 45 s. All nodes agree that the graph is disconnected and that node 2 is missing.

is a “ripple” of agreement as nodes detect that the group is connected or disconnected. Experiment 4 was conducted on a football field to collect quantitative ground truth data. We note that while the logger failed to record the event at 275 s, users were in fact alerted. The connectivity data collected resembles the simulated data (Fig. 7.6).

In Figure 7.14, all the nodes have graph estimates of the systems. Nodes are indicated as yellow circles and edges are blue lines. At this point in time, all nodes estimate a disconnected graph with the same node missing. Additionally, estimated edges are similar.

In our experiments, Grouper has done well to correctly alert a group that it is not together with at least 73% of alerts correctly classified. However, these results are very dependent upon the environment. There were several informal experiments we performed hiking where Grouper seemed to yield more false positives than true positives due to the obstacle-filled environment.

## 7.8 Discussion

Grouper has been developed with the concept of group coordination where group members must stay proximal. As a group member it can be difficult to split attention between tracking the group and the task at hand. We introduce a novel testbed with distributed state estimation using proximity sensing and evaluate it in real-world environments.

To our knowledge, our work is the first to evaluate an RSSI-based tracking system for groups in real-world environments. We characterize RSSI and message frequency as a

function of distance in three types of environments and use this data to estimate the proximity of a group. We then collected ground truth data to quantify system alerts and events and compared estimated states to actual state. In user studies, Grouper correctly alerted groups of separation 74% of the time and detected 80% of all separations.

Using RSSI for localization has the benefit over GPS of indoor operation. Our current characterization of indoor RSSI signals are noisy due to indoor geometry (Fig. 7.9), making thresholding difficult. We need to expand our characterization of indoor geometries to hallways, atria, and cluttered rooms. Ideally, we would like to engineer a context-aware device to infer the environment and dynamically adjust its thresholds, possibly by adding complementary sensors. For example, a system using RSSI and GPS can infer an indoor environment by noting the noisiness of RSSI data and absence of GPS signal. Additional sensors would also aid in alerting users of the direction of the majority of the group. Grouper is currently proximity-based and a lost user cannot determine direction for rejoining the group. The alert is designed to cue members of the group to look for one another.

Wearable devices for groups have potential in numerous applications. In addition to keeping a group together, a system like Grouper can be used to track the interactions of sports teams. Previous work inferred social networks [122]; interaction data can be collected from instrumented team players. User-based data can provide insight into strategy, identify successful interaction patterns, and improve team coordination. This type of data collection would be an alternative to motion capture or other vision systems which may be difficult to install. Similar approaches can be imagined for recreation, rescue, and tactical applications for groups of hikers, firefighters, or soldiers. Another example involves search and rescue in disaster situations. People enter unmapped and dangerous buildings to save victims. Rescuers need to know the location of team members and monitor building condition for their own safety. They also need to coordinate with medical responders to best provide care. This is a complex coordination task that requires a significant cognitive load. Instrumented coordination for groups has the potential to improve safety, efficiency, and reaction time, which may critical for saving lives.



## Chapter 8

### CONCLUSION

In this thesis, we presented algorithms for agreement in multi-agent systems that are extensions of linear average consensus. This involves stochastically interacting particles estimating a scalar quantity (Ch. 5, Ch. 6) and agents estimating a graph (Ch. 7). All of the algorithms use local interactions and the work done on these testbeds can be extended to a more general class of systems.

Input-Based Consensus (IBC), which was presented in Ch. 5, is a consensus-like algorithm that weights the current estimate and initial states in its estimate update. The inclusion of the initial states allows the system to be robust to packet loss [106]. IBC works in well-mixed scenarios and has only been tested for a small number of robots. One of the goals of this work is an extension to large multi-agent systems. Reducing the state of the system to a single scalar value is an attempt to address this goal. However, IBC requires that agents be well-mixed, an increasingly difficult condition to enforce as the number of agents increases. Future work includes adapting IBC to scenarios with different topologies. Another drawback is that IBC requires a known and fixed number of agents. This requirement makes it difficult to adapt to a large and changing population. While we have proven convergence and demonstrated it in simulation, we have yet to implement a hardware system with the estimator and controller.

We presented Feed-Forward Consensus (FFC) in Ch. 6, another consensus-like algorithm that averages the current estimates. The estimate is also updated when an agent changes state. We proved convergence for the estimator and state-switching controller. FFC was simulated in a fixed topology environment in the context of the Stochastic Factory Floor. There may be interesting applications using local estimation of a particular region. FFC is much like Linear Average Consensus (LAC). Both update via a convex combination of states, converge to zero variance, and are robust to the number of agents but sensitive

to unknown initial conditions and dropped messages. We have proven convergence and demonstrated FFC in simulation, but have yet to implement a hardware system with the estimator and controller.

Graph consensus, which was presented in Ch. 7, is an algorithm for distributed agreement on a graph without a particular communication protocol. We presented a model, simulation, characterization of hardware, and experiments in outdoor environments. Much work needs to be done to extend this application to indoor operation. One engineering challenge for Grouper include packets collision when too many agents broadcast at once. Different broadcast protocols will need to be introduced for increased number of agents. Another approach for keeping a large group together is dividing the group into subgroups. This incorporates of distributed assignment of group members to subgroups. Each subgroup could communicate on different channels. As we addressed multi-agent problems, we find many more engineering problems along the way.

“The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”

Mark Weiser [125]

In Weiser’s vision, computers will fade into the background so they are no longer noticeable. We already have computers of many sizes and shapes with us at all times. In the home, robots can clean floors and gutters, mow our lawns, and wake us. Deliberate interaction with personal computers will give way to natural interactions with the environment and ordinary objects that are embedded with computers. Already, microprocessors are found in every day objects including cars, thermostats, and kitchen appliances and we use them without thought. Weiser gives writing as an example of a technology that has disappeared into the background [125]. Writing serves as a means to augment memory, record spoken language, and communicate to others. It is now pervasive in our environments but does not require active attention. Instead, it stores information until attention is given.

In contrast, we are inundated with sensory overload, constantly interrupted by email, phone calls, text messages and websites. These interruptions have lead to bad decision

making when addressed in social situations or while driving. Sensory overload is one of the major problems for multi-agent and many component systems; there are simply too many to track. Rather than interaction on an individual level, one approach is to interact with a large number of agents or devices on a system level. Our work in scalar estimation provides a method for *model reduction* by representing the state of the system as a population fraction. Another way to address sensory overload is designing calming technology that will move from our periphery to our attention when necessary [126]. Users need not pay attention at all times; only when required. We have developed Grouper in an attempt to ease the cognitive load when tracking people in a group.

Lastly, to reduce sensory overload, robots and computers must work together making decisions on our behalf. We only need to look at science fiction for how this might be done. For example, in *Back to the Future II*, Marty wears a jacket that is too big and it adjusts to his size. Later, when he falls into the fountain his jacket dries itself after detecting that it is wet. Marty's jacket has complex functions that require many sensors and actuators. There are many engineering challenges for designing these kinds of wearable devices. How will we choose the functionalities of our devices? What sensors and actuators will be useful in garments? Increasing the number of capabilities increases the required number of components, weight of the entire device, and power requirements.

In addition to engineering challenges, there are many ethical issues. Can we allow devices to make decisions for us? Machines making decisions on behalf of humans has inspired plenty of dystopian science fiction. Adam Greenfield writes in his book *Everyware* that devices must do no harm, default to harmlessness, fail gracefully, and "default to a mode that ensures a users' physical, psychic, and financial safety" [44]. I fear that we, in robotics, have already failed to make devices that "do no harm". Unmanned aerial vehicles (UAVs) are often used to strike down military enemies. UAVs provide many functions including surveillance, firefighting, and research, but like much research it is of dual use. At the time of writing, Iran has recently hijacked a US drone and will likely attempt to extract the technology for use against the US. We are now faced with the consequence with developing technology for war. My hope is to contribute to technology that is beneficial for all.

## Appendix A

### GROUPER DESIGN DETAILS

#### A.1 Summary

Grouper consists of three types of modules: wearable, leader, logger, which are outlined in Ch. 2. Here we list the component details, circuit diagram, sample code, and resources used in building Grouper.

#### A.2 Components

The modules are primarily comprised of components from LilyPad hardware library, which was designed for developing wearable electronics.

Of their LilyPad hardware, SparkFun says,

LilyPad is a wearable e-textile technology developed by Leah Buechley and cooperatively designed by Leah and SparkFun. Each LilyPad was creatively designed to have large connecting pads to allow them to be sewn into clothing. They're even washable!

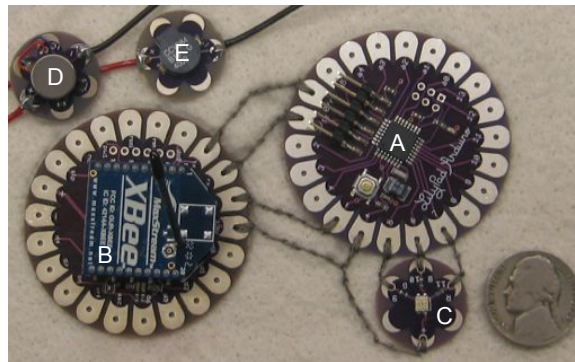


Figure A.1: Early prototype with components sewn. Components are A) Lilypad Arduino, B) XBee, C) tri-LED, D) Vibe motor, E) speaker. Nickel for scale.

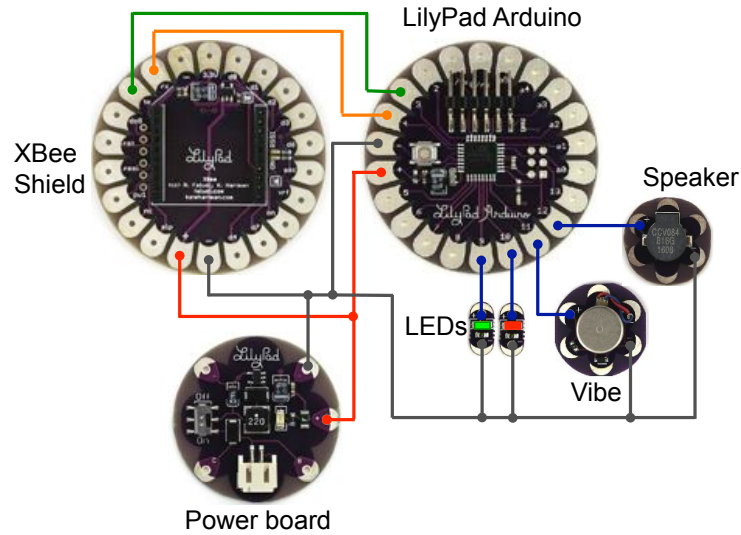


Figure A.2: Circuit schematic for follower module. Components include the XBee shield, LilyPad Arduino, power board, LEDs, vibe motor, and speaker.

Figure A.1 shows components sewn together. In the final prototype, solder was chosen over sewn components. When sewn, the conductive thread becomes both a mechanical and electrical connection for the LilyPad Arduino. This connection becomes loose over after repetitive connection and disconnection of the programming cable into the LilyPad Arduino.

Table A.1 details specific components.

### A.3 Circuit diagram

The circuit diagram for a follower module is shown in Fig. A.2 and includes include the XBee shield, LilyPad Arduino, power board, LEDs, vibe motor, and speaker. The LEDs, vibe motor, and speaker were combined to make the Q board.

### A.4 XBee

Documentation for the XBee Series 1 is found in the XBee manual [31]. The Lilypad XBee shield has pinholes for TX, RX, 3.3V, and ground. We inserted an angled header. and

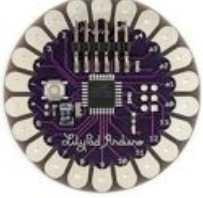
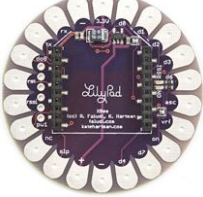




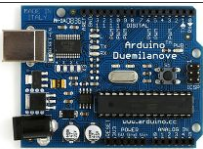


Name	Part number	Description	Picture
LilyPad Arduino	SparkFun DEV-09266	Microprocessor in Grouper system. Arduino is an open source physical computing platform. Designed to be sewing into clothing and runs at 2V to 5V.	
LilyPad XBee Shield	Sparkfun DEV-08937	Breakout board for XBee. Power regulated to operate with the LilyPad Arduino.	
XBee 1mW Wire Antenna	Sparkfun WRL-08665	XBee (Series 1) 2.4GHz radio for communication, operates on 802.15.4.	
LilyPad Power	SparkFun DEV-08786	Power supply using Lithium Polymer batteries at 5V Short circuit protected.	
LilyPad Vibe Board	SparkFun DEV-08468	The vibrating motor is like a cell phone motor. Receives 40mA from output pin.	
LilyPad Buzzer	SparkFun DEV-08463	The buzzer plays sounds at various frequencies. Receives 40mA from output pin.	
Duemilanove Arduino	SparkFun DEV-00666	Arduino microprocessor used for logger module. Arduino is an open source physical computing platform. Operates at 5V.	
Logger Sheild	Adafruit 243	Logger shield plugs into Duemilenove. Writes data to an SD card. 3.3V regulator.	
Battery	SparkFun PRT-00341	Lithium Polymer 3.7V 900mAh battery. Battery includes built-in protection against over voltage, over current, and minimum voltage.	

Table A.1: Grouper components. Pictures from SparkFun and Adafruit.

connected it to a computer to USB port using an FTDI board. We can set configurations for the XBee using any terminal program. X-CTU is a program for Windows used to program XBees.

Table A.4 lists several AT Commands in command mode. Each node was given a unique MY address. The commands ID and CH give the PAN id and channel and must be the same for XBees to communicate. For decentralized operation, we do not have a coordinator so we set CE (Coordinator Enable) to 0; instead, all agents are end devices. End devices are not detected by a coordinator, so A1 is 0. We use XBee.h for node communication, which requires API set to 2. Command API set to 0, sets the XBee in transparent mode and echos the serial.

AT Command	value	purpose
MY	1-12	16-bit address
ID	1111	PAN id
CH	C	channel
CE	0	end device
A1	0	end device association disabled
AP	2	API mode required by XBee.h

Table A.2: Settings for XBees used for decentralized operation.

For the LilyPad Arduino to talk to both the XBee and a computer via serial, we use pins 3 and 4 on the Arduino as software serial using NewSoftSerial.h. Pins 3 and 4 on the Arduino are connected to TX and RX on the XBee, respectively. This leaves pins 0 and 1 on the Arduino to talk to the computer.

### A.5 Sample Code

This program, called RSSItest.pde, was used to gather RSSI data between two modules from measured distances as in Sec. 7.6. A module sends a message every 100 ms to all the other nodes in the system (0xFFFF) . The receiving module prints out the sender address, time, and RSSI of the received packet.

Every Arduino program must contain the functions `setup()` and `loop()`; . The function `setup()` runs once and the function `loop()` runs *ad infinitum*.

```

#include <NewSoftSerial.h>

#include <XBee.h>

//set up serial to xbee
uint8_t txpin = 3;
uint8_t rxpin = 4;
NewSoftSerial xbeeSerial(txpin, rxpin);
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();

uint8_t payload[] = {0, 0};
TxStatusResponse txStatus = TxStatusResponse();
Tx16Request tx = Tx16Request(0xFFFF, payload, sizeof(payload));

// create reusable response objects for responses we expect to handle
Rx16Response rx16 = Rx16Response();

//timing variables in milliseconds
unsigned long lastTimeSent;
unsigned long sendInterval = 100;

//message variables
uint8_t data;
uint8_t rssi;
uint8_t addr;

void setup() {
    Serial.begin(9600);
    xbee.setSerial(&xbeeSerial);
    // start serial

```



```
xbee.begin(9600);  
}  
  
void loop() {  
  parsePacket();  
  
  //send message every sendInterval  
  if (millis() - lastTimeSent > sendInterval)  
    sendMsg();  
  
} // loop  
  
void parsePacket(){  
  // read xbee buffer for 10 ms  
  xbee.readPacket(10);  
  
  while (xbee.getResponse().isAvailable()) {  
    // xbee got something  
  
    if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {  
      // got a rx16 packet  
      getRx16Reponse();  
    }  
  
    long unsigned now = millis();  
    xbee.readPacket(10);  
  
  } // got something  
}
```

```
void sendMsg(){
    lastTimeSent = millis();
    xbee.send(tx);
}

void getRx16Reponse(){
    xbee.getResponse().getRx16Response(rx16);

    uint8_t * data;
    data = rx16.getData();

    rssi = rx16.getRssi();
    addr = rx16.getRemoteAddress16();

    // print sender
    Serial.print(addr, DEC);
    Serial.print(" ");

    // print current time
    Serial.print(millis(), DEC);
    Serial.print(" ");

    // print RSSI
    Serial.println(rssi, DEC);
}
```

## A.6 Resources

We used many online resources to build Grouper. Table A.3 lists suppliers, online tutorials and manuals.

Resource	Purpose
Sparkfun	supplier [34]
Adafruit	supplier [4]
Leah Beuchley	LilyPad tutorial [20]
Lady Ada	XBee tutorial [37]
Digi International	XBee manual [31]
CoolTerm	Terminal for Mac and Windows
FTDi	FTDi driver for board, used with LilyPad [38]
Faludi	Building Wireless Sensor Networks [35]
Igoe	Making Things Talk [57]

Table A.3: Resources for building Grouper.

## BIBLIOGRAPHY

- [1] AAGPS. Amber alert GPS. <https://www.amberalertgps.com/>.
- [2] H. S. AbdelSalam and S. Olariu. Towards enhanced RSSI-Based distance measurements and localization in WSNs. In *IEEE INFOCOM Workshops*, pages 1–2, Apr. 2009.
- [3] G. D. Abowd and E. D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions Computer-Human Interaction*, 7(1):29–58, Mar. 2000.
- [4] Adafruit Industries. Adafruit Industries. <http://www.adafruit.com/>.
- [5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, Mar. 2002.
- [6] C. Alippi and G. Vanini. A RSSI-based and calibrated centralized localization technique for wireless sensor networks. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 1–5, 2006.
- [7] D. E. Altus, R. M. Mathews, P. K. Xaverius, K. K. Engelman, and B. A. D. Nolan. Evaluating an electronic monitoring system for people who wander. *American Journal of Alzheimer's Disease and Other Dementias*, 15(2):121–125, Apr. 2000.
- [8] M. Arai, H. Kawamura, and K. Suzuki. Estimation of ZigBee's RSSI fluctuated by crowd behavior in indoor space. In *Proceedings of SICE Annual Conference 2010*, pages 696–701, 2010.
- [9] A. Awad, T. Frunzke, and F. Dressler. Adaptive distance estimation and localization in WSN using RSSI measures. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, pages 471–478, 2007.
- [10] N. Ayanian, P. J. White, A. Halasz, M. Yim, and V. Kumar. Stochastic control for self-assembly of XBots. *ASME Conference Proceedings*, 2008(43260):1169–1176, 2008.
- [11] K. R. Baghaei and A. Agah. Task allocation methodologies for multi-robot systems. Technical Report, 2002.

- [12] S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V. A. Ziparo. Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from RoboCup rescue. *Journal of Field Robotics*, 24(11-12):943–967, 2007.
- [13] T. Balch, F. Dellaert, A. Feldman, A. Guillory, C. Isbell, Z. Khan, S. Pratt, A. Stein, and H. Wilde. How multirobot systems research will accelerate our understanding of social animal behavior. *Proceedings of the IEEE*, 94(7):1445–1463, 2006.
- [14] B. L. Bassler. How bacteria talk to each other: regulation of gene expression by quorum sensing. *Current Opinion in Microbiology*, 2(6):582–587, Dec. 1999.
- [15] L. Belkin. Are leashes for children? *NYTimes.com*, Mar. 2011.
- [16] K. Benkic, M. Malajner, P. Planinsic, and Z. Cucej. Using RSSI value for distance estimation in wireless sensor networks based on ZigBee. In *15th International Conference on Systems, Signals and Image Processing*, pages 303–306, 2008.
- [17] S. Berman, A. Halsz, M. A. Hsieh, and V. Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937, 2009.
- [18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE ACM Transactions on Networking*, 52(6):2508–2530, 2006.
- [19] Brickhouse Security. Child GPS tracking devices. <http://www.brickhousesecurity.com/about-child-locators-gps-tracking.html>.
- [20] L. Buechley. LilyPad arduino. [web.media.mit.edu/~leah/LilyPad/](http://web.media.mit.edu/~leah/LilyPad/).
- [21] L. Buechley. LilyPad arduino: How an open source hardware kit is sparking new engineering and design communities. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.168.5521>.
- [22] L. Buechley and M. Eisenberg. The LilyPad arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing*, 7(2):12–15, 2008.
- [23] L. Buechley and M. Eisenberg. Fabric PCBs, electronic sequins, and socket buttons: techniques for e-textile craft. *Personal Ubiquitous Computing*, 13(2):133–150, 2009.
- [24] L. Buechley and B. M. Hill. LilyPad in the wild: how hardware’s long tail is supporting new engineering and design communities. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, pages 199–207, New York, NY, USA, 2010.
- [25] F. Bullo, J. Cortes, and S. Martinez. *Distributed Control of Robotic Networks*. Princeton University Press, Princeton, NJ, Sept. 2008.

- [26] S. Burden, N. Napp, and E. Klavins. The statistical dynamics of programmed robotic self-assembly. In *Proceedings of the International Conference on Robotics and Automation*, pages 1469–1476, May 2006.
- [27] M. Canales. Leashes for toddlers: good parenting or bad parenting? <http://www.imperfectparent.com/topics/2011/11/14/leashes-for-toddlers-good-parenting-or-bad-parenting/>, Nov. 2011.
- [28] J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(3):367–385, 2003.
- [29] A. Davids. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, 17(2):81–83, 2002.
- [30] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.
- [31] Digi International Inc. XBee / XBee-PRO RF modules. <http://ftp1.digi.com/support/documentation/90000982B.pdf>.
- [32] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(9):851–871, 2000.
- [33] M. Egerstedt, T. Balch, F. Dellaert, F. Delmotte, and Z. Khan. What are the ants doing? vision-based tracking and reconstruction of control programs. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4182–4187, 2005.
- [34] S. Electronics. Sparkfun electronics. [www.sparkfun.com](http://www.sparkfun.com).
- [35] R. Faludi. *Building Wireless Sensor Networks with ZigBee, XBee, Arduino, and Processing*. O’Reilly Media, 2010.
- [36] R. Freeman, P. Yang, and K. Lynch. Distributed estimation and control of swarm formation statistics. In *Proceedings of the 2006 American Control Conference (ACC)*, pages 749–755, 2006.
- [37] L. Fried. XBee radios. [www.ladyada.net/make/xbee/arduino.html](http://www.ladyada.net/make/xbee/arduino.html).
- [38] Future Devices International Ltd. Drivers. <http://www.ftdichip.com/FTDrivers.htm>.
- [39] GeoEye. GeoEye gallery. <http://geoeye.com/CorpSite/gallery/>, 2009.

- [40] B. Gerkey and M. Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.
- [41] B. Gerkey and M. Mataric. Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 3862–3868, 2003.
- [42] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 1977.
- [43] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer, 2001.
- [44] A. Greenfield. *Everyware: The dawning age of ubiquitous computing*. AIGA, 2006.
- [45] L. Hageman. Do you keep your kid on a leash? *The Philly Post*, Nov. 2011.
- [46] A. Halasz, M. Hsieh, S. Berman, and V. Kumar. Dynamic redistribution of a swarm of robots among multiple sites. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2320–2325, 2007.
- [47] Y. Hatano and M. Mesbahi. Agreement over random networks. *IEEE Transactions on Automatic Control*, 50(11):1867–1872, Nov. 2005.
- [48] J. P. Hespanha. Modeling and analysis of stochastic hybrid systems. *IEE Proceedings of Control Theory and Applications*, 153(5):520–535, Sept. 2006.
- [49] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.
- [50] K. Hosokawa, I. Shimoyama, and H. Miura. Dynamics of Self-Assembling systems: Analogy with chemical kinetics. *Artificial Life*, 1(4):413–427, 1994.
- [51] M. Huang and S. Dey. Kalman filtering with Markovian packet losses and stability criteria. In *Decision and Control, 2006 45th IEEE Conference on*, pages 5621–5626, 2006.
- [52] M. Huang and S. Dey. Stability of Kalman filtering with Markovian packet losses. *Automatica*, 43(4):598–607, Apr. 2007.
- [53] M. Huang and J. H. Manton. Coordination and consensus of networked agents with noisy measurements: Stochastic algorithms and asymptotic behavior. *SIAM Journal on Control and Optimization*, 48(1):134–161, Jan. 2009.
- [54] J. C. Hughes and S. J. Louw. Electronic tagging of people with dementia who wander. *British Medical Journal*, 325(7369), Oct. 2002.

- [55] A. Hussain. Spin on the waltz: Wearable garments as instruments for interactive sound environments in the viennese waltz. Master's thesis, Parsons, The New School For Design, 2009.
- [56] I. Hwang, H. Jang, T. Park, A. Choi, C. Hwang, Y. Choi, L. Nachman, and J. Song. Toward delegated observation of kindergarten children's exploratory behaviors in field trips. In *Proceedings of the 13th international conference on Ubiquitous computing, UbiComp '11*, pages 555–556, New York, NY, USA, 2011.
- [57] T. Igoe. *Making Things Talk: Practical Methods for Connecting Physical Objects*. O'Reilly Media / Make, 2007.
- [58] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48:988–1001, 2003.
- [59] JJCK, LLC. EmFinders. <http://www.emfinders.com/>.
- [60] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. *SIGPLAN Not.*, 37:96–107, Oct. 2002.
- [61] N. G. V. Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, 2007.
- [62] T. Kim, A. Chang, and A. Pentland. Enhancing organizational communication using sociometric badges. In *IEEE 11th International Symposium on Wearable Computing (Doctoral Colloquium)*, page 9, 2007.
- [63] E. Klavins. Programmable Self-Assembly. *Control Systems Magazine, IEEE*, 27(4):43–56, 2007.
- [64] E. Klavins, S. Burden, and N. Napp. Optimal rules for programmed stochastic self-assembly. In *Robotics: Science and Systems II*, pages 9–16, Philadelphia, PA, 2006.
- [65] D. A. Klein, M. Steinberg, E. Galik, C. Steele, J. Sheppard, A. Warren, A. Rosenblatt, and C. G. Lyketsos. Wandering behaviour in community-residing persons with dementia. *International Journal of Geriatric Psychiatry*, 14(4):272–279, 1999.
- [66] K. Konolige, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset, D. Schulz, B. Stewart, and R. Vincent. Centibots: Very large scale distributed robotic teams. In *Experimental Robotics IX*, volume 21, pages 131–140. Springer Berlin / Heidelberg, 2006.



- [67] M. J. B. Krieger, J. Billeter, and L. Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.
- [68] T. H. Labella, M. Dorigo, and J. Deneubourg. Division of labor in a group of robots inspired by ants’ foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):4–25, 2006.
- [69] C. K. Lai and D. G. Arthur. Wandering behaviour in people with dementia. *Journal of Advanced Nursing*, 44(2):173–182, 2003.
- [70] R. Lang, M. Rispoli, W. Machalicek, P. J. White, S. Kang, N. Pierce, A. Mulloy, T. Fragale, M. OReilly, J. Sigafos, and G. Lancioni. Treatment of elopement in individuals with developmental disabilities: A systematic review. *Research in Developmental Disabilities*, 30(4):670–681, July 2009.
- [71] W. W. Lau, G. Ngai, S. C. Chan, and J. C. Cheung. Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students. In *Proceedings of the 40th ACM technical symposium on Computer science education*, pages 504–508, Chattanooga, TN, USA, 2009. ACM.
- [72] P. Law and C. Anderson. IAN resarch report: Elopement and wandering. *Interactive Autism Network*, Apr. 2011.
- [73] P. L.E. ALLIANCE: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 776–783, 1994.
- [74] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc, 1996.
- [75] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97, Atlanta, Georgia, USA, 2002.
- [76] M. J. Mataric, G. S. Sukhatme, and E. H. Ostergaard. Multi-robot task allocation in uncertain environments. *Autonomous Robots*, 14(2):255–263, Mar. 2003.
- [77] L. Matthey, S. Berman, and V. Kumar. Stochastic strategies for a swarm robotic assembly system. In *Proceedings of the International Conference on Robotics and Automation*, pages 1751–1756, 2009.
- [78] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, 2001.

- [79] J. McLurkin. *Analysis and Implementation of Distributed Algorithms for Multi-Robot Systems*. PhD thesis, Massachusetts Institute of Technology, June 2008.
- [80] J. McLurkin and D. Yamins. Dynamic task assignment in robot swarms. In *Proceedings of Robotics: Science and Systems Conference*, 2005.
- [81] M. Mesbahi and M. Egerstedt. *Graph-theoretic Methods in Multiagent Networks*. Princeton University Press, 2010, 2010.
- [82] F. Miskelly. A novel system of electronic tagging in patients with dementia and wandering. *Age and Ageing*, 33(3):304–306, May 2004.
- [83] N. Napp. *Control of Stochastically Interacting Particles: With Applications to Robotics*. PhD thesis, University of Washington, 2011.
- [84] G. Ngai, S. C. Chan, J. C. Cheung, and W. W. Lau. The TeeBoard: an education-friendly construction platform for e-textiles and wearable computing. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 249–258, Boston, MA, USA, 2009.
- [85] R. Olfati-Saber. Distributed Kalman filter with embedded consensus filters. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference*, pages 8179–8184, 2005.
- [86] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.
- [87] S. Park, I. Locher, A. Savvides, M. B. Srivastava, A. Chen, R. Muntz, and S. Yuen. Design of a wearable sensor badge for smart kindergarten. In *Proceedings of the 6th International Symposium on Wearable Computers*, pages 231–238, 2002.
- [88] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12:231–255, 2002.
- [89] L. E. Parker and B. A. Emmons. Cooperative multi-robot observation of multiple moving targets. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 2082–2089, 1997.
- [90] C. J. Perrin, S. H. Perrin, E. A. Hill, and K. DiNovi. Brief functional analysis and treatment of elopement in preschoolers with autism. *Behavioral Interventions*, 23(2):87–95, 2008.
- [91] C. C. Piazza, G. P. Hanley, L. G. Bowman, J. M. Ruyter, S. E. Lindauer, and D. M. Saiontz. Functional analysis and treatment of elopement. *Journal of Applied Behavior Analysis*, 30(4):653–672, 1997.

- [92] S. Pratt, E. Mallon, D. Sumpter, and N. Franks. Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant *leptothorax albipennis*. *Behavioral Ecology and Sociobiology*, 52(2):117–127, July 2002.
- [93] D. Puccinelli and M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. *Circuits and Systems Magazine, IEEE*, 5(3):19–31, 2005.
- [94] J. H. Reif and H. Wang. Social potential fields: a distributed behavioral control for autonomous robots. In *WAFR: Proceedings of the Workshop on Algorithmic Foundations of Robotics*, pages 331–345. A. K. Peters, Ltd., 1995.
- [95] W. Ren, R. Beard, and E. Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005 American Control Conference (ACC)*, volume 3, pages 1859–1864, 2005.
- [96] W. Ren, R. Beard, and D. Kingston. Multi-agent Kalman consensus with relative uncertainty. In *Proceedings of the 2005 American Control Conference (ACC)*, pages 1865–1870 vol. 3, 2005.
- [97] W. Ren and R. W. Beard. *Distributed consensus in multi-vehicle cooperative control: theory and applications*. Springer, 2008.
- [98] Y. Ren, S. Kiesler, and S. Fussell. Multiple Group Coordination in Complex and Dynamic Task Environments: Interruptions, Coping Mechanisms, and Technology Recommendations. *Journal of Management of Information Systems*, 25(1):105–130, July 2008.
- [99] L. Robinson, D. Hutchings, H. O. Dickinson, L. Corner, F. Beyer, T. Finch, J. Hughes, A. Vanoli, C. Ballard, and J. Bond. Effectiveness and acceptability of non-pharmacological interventions to reduce wandering in dementia: a systematic review. *International Journal of Geriatric Psychiatry*, 22(1):9–22, 2007.
- [100] M. Rosenberg and S. Love. Press release: Kiva Systems chosen by Walgreens to double distribution center capacity. [http://www.kivasystems.com/media/30814/kivasystems'walgreens](http://www.kivasystems.com/media/30814/kivasystems%27walgreens)Mar. 2009.
- [101] P. Rudol and P. Doherty. Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery. In *Aerospace Conference, 2008 IEEE*, pages 1–8, 2008.
- [102] SafetyBright.com. <http://safetybright.com/ChildSafetyProds.php>.

- [103] R. Schollmeier, I. Gruber, and M. Finkenzeller. Routing in Mobile Ad-hoc and Peer-to-Peer Networks: A Comparison. In E. Gregori, L. Cherkasova, G. Cugola, F. Panzieri, and G. Picco, editors, *Web Engineering and Peer-to-Peer Computing*, volume 2376 of *Lecture Notes in Computer Science*, pages 172–187. Springer Berlin / Heidelberg, 2002.
- [104] M. Schwager, C. Detweiler, I. Vasilescu, D. M. Anderson, and D. Rus. Data-driven identification of group dynamics for motion prediction and control. *Journal of Field Robotics*, 25(6-7):305–324, 2008.
- [105] T. Seeley. *The Wisdom of the Hive*. Harvard University Press, 1996.
- [106] F. Shaw, A. Chiu, and J. McLurkin. Agreement on stochastic multi-robot systems with communication failures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6095–6100, 2010.
- [107] F. Shaw and E. Klavins. Distributed estimation and control in stochastically interacting robots. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 1895–1901, Dec. 2008.
- [108] F. Shaw and E. Klavins. Distributed estimation and state assignment for stochastically interacting robots. In *IFAC Workshop on Networked Systems and Controls*, Sept. 2009.
- [109] F. W. Shaw and E. Klavins. Grouper: A Proof-of-Concept Wearable Wireless Group Coordinator. In *Proceedings of the 12th ACM International Conference Adjunct Papers on Ubiquitous Computing*, pages 379–380, 2010.
- [110] X. Shen, Z. Wang, P. Jiang, R. Lin, and Y. Sun. Connectivity and RSSI based localization scheme for wireless sensor networks. In D. Huang, X. Zhang, and G. Huang, editors, *Advances in Intelligent Computing*, volume 3645 of *Lecture Notes in Computer Science*, pages 578–587. Springer Berlin / Heidelberg, 2005.
- [111] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Dynamic consensus for mobile networks. In *Proceedings of the 16th IFAC World Congress*, 2005.
- [112] K. Srinivasan and P. Levis. RSSI is under appreciated. In *Proceedings of the 3rd Workshop on Embedded Networked Sensors*, pages 34–40, 2006.
- [113] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and Low-Bandwidth communication for Real-Time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

- [114] A. Tahbaz-Salehi and A. Jadbabaie. A necessary and sufficient condition for consensus over random networks. *IEEE Transactions on Automatic Control*, 53(3), Apr. 2008.
- [115] J. Tanenbaum, K. Tanenbaum, and A. Antle. The reading glove: designing interactions for object-based tangible storytelling. In *Proceedings of the 1st Augmented Human International Conference*, pages 1–9, New York, NY, USA, 2010.
- [116] Tattoos With A Purpose, LLC. Temporary tattoos with a purpose. <http://www.tattooswithapurpose.com/>.
- [117] A. Team. Arduino. <http://www.arduino.cc/>.
- [118] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3):253–271, July 1998.
- [119] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, Sept. 1986.
- [120] M. M. Veloso and D. Nardi. Special issue on multirobot systems. *Proceedings of the IEEE*, 94(7):1253–1256, 2006.
- [121] Vital ID. [http://www.vitalid.ca/store/Scripts/prodView\\_kid.asp?idproduct=1](http://www.vitalid.ca/store/Scripts/prodView_kid.asp?idproduct=1).
- [122] B. N. Waber, D. O. Olgun, T. Kim, A. Mohan, K. Ara, and Y. Pentl. Organizational engineering using sociometric badges. In *International Workshop and Conference on Network Science*, 2007.
- [123] I. Wagner and A. Bruckstein. From ants to a(ge)nts : A special issue on Ant-Robotics. *Annals of Mathematics and Artificial Intelligence*, 31(1):1–5, Oct. 2001.
- [124] R. Want, A. Hopper, V. Falco, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.
- [125] M. Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing Communications Review*, 3(3):3–11, 1999.
- [126] M. Weiser and J. S. Brown. The coming age of calm technology. In *Beyond calculation*, pages 75–85. Copernicus, 1997.
- [127] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 347–356. Springer-Verlag, 2000.

- [128] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–19, 2008.
- [129] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 5, pages 4997–5002, 2003.
- [130] L. Xiao, S. Boyd, and S. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, Jan. 2007.
- [131] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks*, pages 63–70, Apr. 2005.
- [132] Y. Yang, M. Chuang, S. Lou, and J. Wang. Thick-film textile-based amperometric sensors and biosensors. *Analyst*, 135(6):1230–1234, Mar. 2010.
- [133] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics and Automation Magazine*, 14(1):43–52, Mar. 2007.
- [134] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, and C. J. Taylor. Towards robotic self-reassembly after explosion. In *Proceedings of the 2007 IEEE Conference on Intelligent Robot Systems (IROS)*, pages 2767–2772, 2007.
- [135] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st international Conference on Embedded Networked Sensor Systems, SenSys '03*, pages 1–13, New York, NY, USA, 2003.
- [136] M. Zhu and S. Martinez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, Feb. 2010.
- [137] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the 2002 IEEE International Conference Robotics and Automation (ICRA)*, volume 3, pages 3016–3023, 2002.