Distributed Coordination of Multi-agent Based on Estimation
over Ad Hoc Communication Networks

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in
Electrical Engineering

by

Yashan Sun, B.S., M.S.

________________________________________
Michael D. Lemmon, Director

Graduate Program in Electrical Engineering
Notre Dame, Indiana
May 2007
Distributed Coordination of Multi-agent Based on Estimation over Ad Hoc Communication Networks

Abstract

by

Yashan Sun

Distributed coordinated multiple agent system attracts significant interests in the recent decades. Typical applications include unmanned autonomous vehicle formation control, automated highway system and sensor network. A common feature for these applications is that cooperative behaviors are accomplished by interactions among agents wherein information exchange across a wireless communication network. Distributed coordinated control of multiple agent system poses meaningful theoretical and practical challenges. In this work, we consider three important and related issues: communication logic, swarm cohesion under consensus and convergence rate of consensus filter under network throughput limitations.

An optimal communication logic is investigated in Chapter 2 for scheduling the information exchange to achieve the required system performance, at the minimal energy consumption. Different communication logics are studied. A state-independent, or “open-loop” logic is proposed in which each agent transmits periodically. For comparison, a state-dependent, or “closed-loop” logic is also studied, which schedules transmission when the local state estimation error is above a preset threshold. We propose an approach to theoretically analyze the
threshold-based logic performance and compared it with the proposed periodic logic.

Cohesion of multi-agent swarms moving under the control of a consensus filter is studied in Chapter 3. Main result shows that swarming under consensus is cohesive. We establish specific bounds on the degree of cohesion and consensus as a function of the attraction/repulsion fields, the number of swarm members, and connectivity in the communication network. We prove that if the swarm’s communication graph is regular, then the introduction of integral action into the consensus filter achieves perfect consensus regardless of the number of members.

Consensus filters provide a distributed way of computing data aggregates in distributed multiple agent system. Prior work has suggested that the rate at which such filters achieve consensus is proportional to the number of neighbors in the communication network. This conclusion, however, is simplistic because it ignores the intrinsic throughput limitation of multi-hop networks. The convergence behavior of consensus filters under such throughput limitations is examined in Chapter 4. We consider a time-slotted frequency division multiple access (FDMA) network assuming a regular network. Under these assumptions we show that throughput limits can be modeled as delays. We presented two consensus filter schemes, synchronous and asynchronous consensus, associated to different communication protocols. Synchronous consensus obeys the principle wherein individual agents regulate their states only after receiving all neighbors’ message. While, in asynchronous consensus manner, each agent updates its state if and only if receives any delayed message from any neighbors. We exhibit an theoretical approach to analyze the impact message delays have on the convergence rate of two consensus schemes, respectively, and demonstrate the specific advantages
of individual schemes.

In order to test these studied coordinated control algorithm in real multi-robot system, we developed a software multi-robot simulator and hardware testbed of real robots. Our developed simulator and testbed are introduced in Chapter 5. The simulator enables us to rapid develop the efficient control algorithms without accessing to the real hardware and environment, which dramatically cuts down the time and energy-consuming on complicated programming. Our laboratory robotic vehicle testbed is built to experiment on the proposed coordinated control algorithms.
## CONTENTS

**FIGURES** ................................................................. iv

**TABLES** ........................................................................ vi

**ACKNOWLEDGMENTS** ...................................................... vii

**CHAPTER 1: INTRODUCTION** ............................................. 1
  1.1 Background and Motivation ............................................. 1
  1.2 Related Work ............................................................... 3
    1.2.1 MAS Formation Control ........................................... 3
    1.2.2 Multi-agent Swarming .............................................. 9
    1.2.3 Coordinated Behaviors under Consensus ....................... 12
    1.2.4 TestBed ................................................................. 17
  1.3 Statement of Contribution ............................................... 21

**CHAPTER 2: COMMUNICATION LOGICS FOR THE DISTRIBUTED**
  **CONTROL OF MAS** ..................................................... 27
    2.1 Overview ................................................................. 27
    2.2 Introduction ............................................................... 28
    2.3 Control-communication Model ....................................... 30
    2.4 Open-loop Communication Logic .................................... 32
    2.5 Evaluation of Periodic Communication Logic ....................... 38
    2.6 Closed-loop Communication Logic ................................... 45
    2.7 Simulation ................................................................. 50

**CHAPTER 3: COHESIVE SWARMING UNDER CONSENSUS** ....... 54
    3.1 Overview ................................................................... 54
    3.2 Introduction ............................................................... 55
    3.3 Swarm Dynamic Model ................................................... 57
    3.4 Error Equations ......................................................... 60
    3.5 Uniform Ultimate Bound Analysis .................................... 64
### FIGURES

1.1 Interconnection of swarm and consensus filter ................................. 16
2.1 Markov chain for the threshold-based logic .................................... 46
2.2 Transition probability of Markov chain ........................................... 47
2.3 Average cost in different communication logics ............................... 51
2.4 Average cost versus different communication logic variables ............ 52
2.5 Variance of open-loop logic performance ....................................... 53
2.6 Simulated and theoretical performances ........................................... 53
3.1 Interconnection of swarm and consensus filter ................................ 57
3.2 Geometric analysis of interconnected system cohesiveness ................. 70
3.3 Convergence rate of interconnected system cohesiveness ................. 73
3.4 Convergence rate of consensus error .............................................. 74
3.5 High-energy J and low-energy J versus swarm error ....................... 77
3.6 Agent configurations associated with low-energy(left) and high-energy(right) ................................................................. 78
3.7 Final swarm/consensus error vectors .............................................. 79
3.8 Swarm/consensus time history ....................................................... 80
3.9 Comparison with analytical bounds ................................................. 81
3.10 Communication graph, N= 20 (left) 8-degree (right) connected graph 89
3.11 Out degree distribution of connected graph .................................... 90
3.12 Consensus error bound with integral action, N=20 ........................ 91
3.13 Consensus error equilibrium with /without integral action ............... 92
3.14 Consensus error bound with different max and min communication degree ................................................................. 92
4.1 Property of the eigenvalues of $\mathbf{A}$ ........................................ 106
ACKNOWLEDGMENTS

Mere words cannot express the gratitude I owe the people who have made it possible for me to write this dissertation. I would first like to acknowledge my advisor, Prof. Lemmon for his assistance and guidance. His unquenchable enthusiasm and infectious spirit of hard work make him a great mentor to me. I am grateful to my committee members: Prof. Bauer, Prof. Goodwin and Prof. Hu.
CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

A multi-agent system (MAS) is a collection of intelligent agents interacting with each other in pursuit of a common goal. Each individual agent is an autonomous entity, which has the ability to sense, compute and act. Those agents cooperate their action in a distributed manner, in terms of their internal as well as neighbor’s state, through sensing and communicating. The study of MAS is motivated by models from biology and economics (e.g., [96]). It is normal phenomena that flocks of birds and schools of fish are able to perform formation maneuvers without a center leader [52]. Inspired by this observation, researchers proposed a similar framework for studying a multiple-agent system. Current MAS applications cover an increasing variety of domains, ranging from manufacturing to process control [10, 51], traffic control [1, 14], information management [69, 86] and sensor network application [22, 71].

Control strategies for a MAS were originally studied in the centralized scheme, in which a specified agent is assumed to be able to observe the states of all of agents in the network and assign control commands for all agents. This control strategy is, however, fragile because if the specified agent stops functioning, the entire system will fail.
A distributed control strategy is more robust to the specified agent’s failure than centralized control strategy, since there is no need for a specified agent to direct the behaviors of the entire system. Each agent only depends on its neighbors for information update, and actions are determined locally. Thus, distribution dramatically improves flexibility and robustness to agent’s failure for MAS. However, one drawback of the distributed strategy is that individual agents only have partial information of the entire system state. To successfully achieve the global objective, agents must therefore coordinate their actions. Much as people do, the cooperation is supported by communicating across the group of agents.

Multi-agent systems (MAS) over ad-hoc communication networks have recently received much attention in the control area. The distributed character of MAS makes it a natural choice to adopt ad-hoc networks as the communication layer, since an ad hoc network is a mobile communication network which does not have fixed base stations to support information exchange [15, 68]. The combination facilitates the design and implementation of distributed control systems. Nevertheless, unlike the ideal communication layer assumed in traditional control systems, wireless communication networks can only provide limited communication range due to the bounded communication resource. Therefore, in a practical multi-agent system, each individual agent only has access to its neighbor’s information directly, rather than an entire networks. Henceforth, the behavior of an agent is regulated solely based on this restricted knowledge, which may significantly affect the performance of MAS. In another word, the combination brings up many interesting and challenging issues. For example, it is critical to efficiently implement the control strategies to optimize the MAS performance, given limited communication throughput. Furthermore, the effect of limited communication
energy on the behavior of MAS also needs to be studied.

In order to conquer these challenges, estimation strategies are proposed to estimate desirable control parameters so that helps to reduce the communication period and hence, decrease the communication cost. On the other hand, the estimation error will propagate the system control scheme and affect the system performance. Correspondingly, a serial of optimization problems are addressed according to different objectives to implement the estimation strategies efficiently plus optimizing MAS performance. This work is dedicated to presenting our recent research on distributed control schemes based on estimation for networked multi-agent systems, which can be roughly divided into three topics: communication logics for the distributed control of MAS, cohesive swarming under consensus, and convergence of consensus filtering under network throughput limitations.

1.2 Related Work

Distributed strategy for coordinated control of MAS has been studied for decades. Taking into account communication limitations, researchers have studied various types of techniques, which try to effectively deal with the trade-off between the system performance and communication costs. A short survey of recent related work is provided in the section, which obviously is bounded by author’s ignorance and biases.

1.2.1 MAS Formation Control

Robotic vehicle formation control is a typical illustration of the coordination problem in MAS, in which every robot acts as an autonomous agent with the ability to sense, compute and act. Formation control has been extensively investigated
in numerous applications such as coordination of multiple robots [5, 27, 90, 97], formation flight of unmanned aerial vehicles [70, 81, 92], operations of underwater vehicle [28, 85], and vehicle platoon control [23, 40]. The goal of formation control is for a group of agents to maintain their relative positions by cooperation with limited communication resources.

- Formation Control with Limited Communication Coverage
  Because the wireless signals fade over distance, each agent only has access to a partial view of the entire system’s states. The topology of the communication networks with limited coverage can be categorized as rooted or rootless. A central node such as a gateway is available in a rooted communication network, which is capable of collecting all agent’s information and distributing local information across the whole network. In such a rooted network, other communication nodes can work as relays. On the contrary, data communications in a scenario without a gateway is called rootless network, where individual nodes in the network have the same priorities to transmit and receive information from others. Correspondingly, the control schemes adopted in formation applications is classified as “leader-follower” [20, 21] and “leaderless” [7, 47].

In “leader-follower” scheme, a physical (or virtual) leader specifies a desired reference trajectory. All followers track this reference trajectory in a way that maintains the group’s formation. In this approach, if the leader is a physical robot, then the leading robot is responsible for computing each robot’s relative position and informing corresponding followers. Namely, the leader plays the role of a central controller. At the same time, individual followers must track the reference path well. Underlying communication
topology could be modeled as a spanning tree in which the leader acts as the root of the tree. Even though the communication architecture is simple, it requires a large amount of communications to maintain the positions of individual followers. If the communication graph is not completed, the leader’s decision would have to be relayed to the followers. Multihop transmission delay becomes a limiting factor in the system performance. If the leader is virtual (the geometry center of a formation), then all robots must have foreknowledge of the reference trajectory, and no center’s broadcast is required. This scheme however, is impractical in the situation of obstacle avoidance scenarios, since it is extremely difficult for distributed agents to automatically adjust the reference trajectory without a central decision-maker. Hence, one of the most challenging issues of the leader-follower scheme is the communication cost, especially when the system scale is not negligible.

When individual robots only use their neighbors’ information to decide their local behavior, we refer to the scheme as leaderless. Under this scheme, besides the internal state measurement by each robot, it also knows the position relative to its neighbors, like,

\[
\dot{x}_i = f(x_i, u_i(x_i, x_{j\in\mathcal{N}_i}), w_i)
\]  

(1.1)

where \(x_i\) denotes the states of robot \(i\), and \(u_i\) is the control effort at robot \(i\). The set \(\mathcal{N}_i\) consists of the neighbors of the \(i\)th robot. The exogenous disturbance is given by \(w_i\). The control effort \(u_i\) for the formation control is a linear combination of a set of robots state. The interaction topology is the key for designing formation control.
Formation Control with Limited Communication Capacity

Besides communication connectivity, communication capacity also has effects on the coordination system performance. Given limited communication capacity, the communication may be erroneous, or the information packet could be dropped during transmission, which will result in inaccurate control decisions. In addition, information packets could be delayed because of the limited bandwidth of the wireless network. Namely, communications are not reliable because the communication capacity is limited in the wireless network. Exogenous disturbance $w_i$ in equation (1.1) could represent the possible erroneous packets. Due to these practical considerations, estimation or prediction techniques have been employed for reducing the capacity impact on system performance. Individual agents use current communication data as well as the previously received information to predict or estimate the future behavior. The possibility of inaccurate control decision hence could be decreased in this way.

One of widely used control techniques is model predictive control (MPC), studied in [76, 77, 106] for formation control. MPC is a feedback control scheme in which a trajectory optimization is solved dynamically. The trajectory optimization problem is how to operate a robotic vehicle efficiently. To address the scalability issue of a MAS, the decentralized MPC (DMPC) is developed to reduce the computational burden at each agent. Each agent only solves a sub-problem for individual trajectory planning, and these sub-problems are solved only once per time step. Assuming a bounded disturbance, all sub-problems are guaranteed to be feasible [76].

To reduce the influence of communication delay, the state estimator frame-
work is proposed in [108]. The main idea is that each agent estimates the states of other agents, and then uses the estimates for local control effort. That estimation approach is hence able to reduce the effect of communication delay on system performance. Unlike the successive communication strategy used in [76, 77], this framework employs the threshold-based communication logic. When the estimates deviate from the true states by a given level, the true state is broadcast to the rest of the system. By constraining the estimation error, the communication logic adapts the performance of the entire system. The dynamics of estimators at the \( k \)-th robot for robot \( i \) is given by

\[
\begin{align*}
\dot{x}_i^{(k)}(t+1) &= A_i \hat{x}_i^{(k)}(t) + \sum_{j \in \mathcal{N}_i(t)} B_{ij} \hat{x}_j^{(k)}(t) \\
\hat{y}_i^{(k)}(t) &= C_i \hat{x}_i^{(k)}(t)
\end{align*}
\]

where \( \hat{x}_i^{(k)}(t) \) is the estimated state of robot \( k \) by robot \( i \) at time \( t \). The \( i \)th robot has the observations of \( y_i \) and \( \hat{y}_i^{(i)} \) denotes predicted positions for the robot that are later in the planning sequence. System matrices \( A, B \) and \( C \) are of appropriate dimensions. If \( \left| y_i - \hat{y}_i^{(i)} \right| \geq \epsilon_i \), then the \( i \)th robot will broadcast its actual state \( x_i(t) \). When the \( i \)th robot broadcasts at time \( t \), \( \hat{x}_i^{(k)}(t) \) is replaced by \( x_i(t) \). The expected communication frequency is shown in [108] as a function of the desirable system performance, which implies the threshold is decided by the tolerable performance degradation.

This approach is extended in [103], where a stochastic communication logic is proposed for the same framework. The probability of an agent broadcasting a message is a Poisson processes, of which the rate is dependent
on estimation error, i.e., $e_i = \hat{x}_i^{(i)} - x_i$. The system performance is characterized by the statistical moments of the estimation error, $E[e_i^T e_i]$. By applying different broadcast density functions, we can study the trade-off between packet exchange rate and system performance.

Aforementioned estimation techniques are to achieve a better system performance with unreliable information exchange. The unreliability arises in the limitation of communication resources. Specifically, the limited communication capacity results in the data packet delay, error and/or loss.

• Formation Control with Limited Communication Energy

A successful wireless transmission consumes considerable energy. Multiple agent systems are usually of high flexibility and mobility, so that each agent mainly relies on a battery. Due to limited power supply, excessive energy consumption will result in agents failure, while minimizing the amount of transmissions which can prolong the lifetime of MAS. One approach to conserve energy is to reduce the rate of the information exchange. Therefore, energy efficiency becomes a critical objective for communication logic design. The optimized closed-loop communication logic was investigated in [104] based on the same estimation framework in [103]. Closed-loop communication logics schedule broadcasts conditioned on the current system states. One closed-loop communication logic is the threshold-based communication protocol discussed in [108] and [103]. The closed-loop communication protocol in [104] is applied to jointly minimize the costs for both communication and estimation error. Dynamic programming technique is used to study the optimal closed-loop communication logic, where the broadcasting decision is a threshold test whether the measured estimation error exceeds a spec-
ified level. The authors focused on the discrete time model proposed in [104]. Let \( \{u_i[t]\} \) denote the sequence of broadcast decisions made by the \( i \)th agent at time \( t \). In particular, let \( u_i[t] = 1 \) if agent \( i \) broadcasts its state, and \( u_i[t] = 0 \) otherwise. The optimization problem tries to minimize the long term average cost [104]

\[
J = \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ \sum_{t=0}^{T-1} (e_i[t]^T Q e_i[t] (1 - u_i[t]) + u_i[t] \lambda) \right]
\]

where \( \lambda > 0 \) is the energy cost of broadcasting across the network, \( Q > 0 \) and \( Q \in \mathbb{R}^{N \times N} \) is a fixed weight matrix, and \( T \) is the horizon’s length.

Closed-loop communication logic relies on the quality of error estimation, as well as the threshold, to make the decision. An inaccurate error estimation, or mis-chosen threshold parameter may result in considerable performance degradation. In addition, the estimation error is not available for the agent locally in some situations. To overcome those problems, we studied optimal open-loop communication logic based on the same estimation framework, where the communication logic is called “open-loop” if the broadcast decision is not related to the current state of the system.

1.2.2 Multi-agent Swarming

In recent decades, researcher of MAS application such as formation control have pay more attention in swarming and collective motion patterns [3, 32–34, 50, 53, 60, 85, 89, 91] and [62]. Since, in nature, swarming and collective patterns have good features of the mobile agents capable of coordinated group behaviors in a distributed manner, and throw a light in design of self-organized MAS networks.

Reynold [75] developed a computer animation model for coordination motion
of groups of animals such as bird flock and fish school in the late of 1980’s. In the model, Reynold introduced three heuristic rules: flocking centering, collision avoidance and velocity matching. These rules were motivated by the observation of biological society foraging for food or avoiding predators, and provided the basic objectives to analyze a cohesion swarm framework. Similar theoretical models were found in physics and biology [37, 55, 82, 93, 95] and [56] most in the 1990’s. Among these models, two distinct methodologies are used to describe the swarm dynamic in common which are continuum (Eulerian) method and individual-based (Lagrangian) method. The Eulerian approach applies nonlinear partial differential equations to follow the evolving swarm density. Even though it has a well-studied mathematical standpoint, the limitation lies on that continuous description does not be allowed to numerous situations. In the Lagrangian method, the basic approach is based on Newtonian mechanics equations of motion on individuals, and therefore it is more efficient for modeling and analysis of complex social interactions and aggregations.

Understanding the operational principle from the biological swarming model, the control scientists have recently begun studying MAS for applications involving cooperative groups of unmanned autonomous vehicles (UAV’s). The cooperative behaviors includes moving in formation [42] [63] [57] [26], aggregating in swarms [32] [34] [53] [89], and exploring hazardous environments [60] [18]. They take the advantage of Lagrangian model to investigate agent coordination performance.

Broadly speaking, Lagrangian models can be divided further into two types; swarming and flocking. The term “swarming” is often reserved for kinematic models in which swarm members are treated as point masses. The standing assumption in this case is that viscous forces are large enough so that an agent’s acceleration
is only significant over a short period of time. The $i$th agent’s state, $x \in \mathbb{R}^n$, therefore, satisfies a first order differential equation $\dot{x}_i(t) = F_i(t)$. The function $F_i$ is the control signal. On the other hand, the term “flocking” pertains to a group of agents whose states satisfy a second order differential equation, $\ddot{x}_i(t) = F_i(t)$, in which individuals react to external forces by accelerating. This is clearly distinct from the “swarming” case in which inertial forces are neglected. In both cases, the control input can be written as

$$F_i(t) = \sum_{j \in \mathcal{N}_i} f(x_i, x_j) + u_i$$

(1.2)

where $\mathcal{N}_i$ is the set of the $i$th agent’s neighbors, $f_i : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ models the inter-agent forces and $u_i$ is an exogenous input.

The interaction function $f_i$ are usually described as the gradient of a potential field inspired by mathematical biology. This potential field can be automatically generated from proximity sensors detecting neighboring agents and obstacles. Potential fields associated with obstacles cause agents to move away from the obstacle, while potential fields generated by neighboring agents are based on long-range attraction and short-range repulsion between agents. This mechanism helps assure the cohesiveness of the swarm while minimizing the likelihood of agent collisions. The mixture of short-range repulsion and long-range attraction forces is in the form,

$$f(x_i, x_j) = \rho(\|x_i - x_j\|)(x_i - x_j) - \alpha(\|x_i - x_j\|)(x_i - x_j)$$

(1.3)

where $\rho : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ and $\alpha : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ represent repulsive and attractive forces between agents, respectively, and $\|x_i - x_j\|$ is the Euclidean distance between agent
\( i \) and \( j \). A serial of empirical models used to describe attraction and repulsion between agents in the group is presented [8, 29, 33, 56, 59]. Both piecewise linear, inverse-power and exponential distance dependence have been applied in the analysis the swarm stability. These distance-based dynamic agent interactions can guarantee collision avoidance, regardless of the structure of the interconnection graph. Stability analysis of a groups of agents with all-to-all interconnections were consideration in [32, 89]. Other works [42] [62] considered agents coordination in which \( \mathcal{N}_i \) only captured nearest neighbor interactions.

1.2.3 Coordinated Behaviors under Consensus

As an inherently distributed strategy to multi-agent coordination, information consensus has received significant attention in the control community recently. The consensus problem is that agents in the group must agree on certain quantities of interest eventually, under the challenge on lack of the global view of the entire networked information. Specially, in the consensus algorithm, agents update the value of their states only based on the states of their neighbors, but each agent’s state will be propagated across the MAS step by step during every iteration. Surprisingly, consensus algorithms provide a possible distribution solution for the group of agents converge to a common state finally. For example, one consensus issue in formation control application addressed in [42] is the heading agreement problem. That is, initially autonomous agents are moving at the same speed but in different headings, and ultimately they head to the same direction by regulating local heading according to the information from neighbors.

Considerable research, e.g., [6, 30, 57, 64, 72, 74, 88, 89, 102, 107], has been done on consensus algorithms and their convergence analysis. The most common
consensus algorithm is given by

\[ \dot{x}_i(t) = - \sum_{j \in \mathcal{N}_i} a_{ij}(t) (x_i(t) - x_j(t)) \]  (1.4)

where \( a_{ij}(t) \) denotes the weight between agent \( i \) and \( j \) associated with communication graph at time \( t \). Setting \( a_{ij} = 0 \) means the fact that no communication occurs between agent \( i \) and \( j \). Analogous to the distributed control law in equation (1.1), the control effort \( u_i \) is some simple linear average principle. The consensus objective is to achieve \( \lim_{t \to \infty} x_i(t) = x_{ss} \) for the entire group \( i = 1, \ldots, N \), where \( x_{ss} \) is a constant value, depending on the initial state \( x_i(0) \). If the stable value is the average of initial states, that is \( \lim_{t \to \infty} x_i(t) = \frac{1}{N} \sum_{i=1}^{N} (x_i(0)) \) then this consensus problem is called average-consensus [64, 65].

The distributed system coordination performance is strongly dependent on the communication topology. For instance, the average consensus can be obtained, either in undirected communication graphs [65], or in some particular directed communication graphs [64], which are named as balanced graphs. The topology of valid graphs implies that, for each agent, the number of agents to which it sends information should always be the same as that from which it receives information. More general consensus could be acquired in the undirected communication graphs with switching topologies. The sufficient condition for this type of consensus problem is that the union of interaction graphs for the teams are connected frequently enough as the system evolves [42]. Ren extended the results of [42] from undirected case to directed case in [74]. It has been shown that consensus can still be achieved, if the collection of interaction graphs during some time interval has a spanning tree frequently enough.

In prior works, different optimization tools have been used to analyze con-
sensus convergence under the constraints of communication limitation. In [64], it is shown that the convergence rate is proportional to the connectivity of the communication network. The attainable performance of the coordination system therefore is characterized by the the second-smallest eigenvalue of the Laplacian matrix associated with the communication graph. Luc in [57] applied a set-valued Lyapunov approach to study the consensus algorithm with directed switching communication graphs. This rate can also be changed by manipulating weights on the edges of the communication graph [102] [30]. Xiao in [102] proposed a distributed iterative algorithm to seek a fast convergent control weight adjustment on $a_{ij}(t)$ in equation (1.4) based on the fixed communication graphs.

Wireless network throughput limitations [38] have a major impact on the consensus convergence rate. A direct consequence of limited throughput capacity is longer communication delay or latency. Due to message collisions, it is impossible for a receiver to collect all its neighbors’ information instantaneously. There is always a finite probability that some of the neighboring data will be corrupted and require retransmission. Resending data will delay message delivery in a way that can adversely effect the relative stability of the consensus function, and in a way that decrease the convergence rate. Let $\tau_{ij}$ denote the time delay for information communication from agent $j$ to reach agent $i$. In this case, the equation (1.4) is modified as

$$\dot{x}_i(t) = -\sum_{j \in N_i} a_{ij}(t) (x_i(t - \tau_{ij}) - x_j(t - \tau_{ij}))$$

As for the uniform constant delay case, where $\tau_{ij} = \tau$ and communication topology is fixed, undirected, and connected, average consensus is achieved if and only if $0 < \tau < \frac{\pi}{2\lambda_{max}(L)}$, where $L$ is the Laplacian matrix of the communication graph.
The similar results has been extensively studied by taking the new framework of Partial difference Equations on graphs in [11]. A local closed-loop control scheme is proposed in [48] to eliminate the non-uniform communication delay impact on agreement convergence.

General information graph with non-uniform delay are considered as an asynchronous consensus framework [12, 24, 54], where agents are modeled as first-order discrete-time systems.

\[ x_i(t + 1) = \frac{1}{\sum_{j=1}^{n} a_{ij}(t)} \sum_{j=1}^{n} a_{ij}(t)x_j(t - \tau_{ij}(t)) \quad (1.5) \]

where \( \tau_{ij}(t) \leq \tau_{max} \) is the transmission time-delay of information from agent \( j \) to agent \( i \), and \( \tau_{ii}(t) \equiv 0 \). The maximal communication time-delay is denoted by \( \tau_{max} \). The weight factor \( a_{ij}(t) \) is allowed to dynamically change, representing possible communication topology switching, related to the reliability of information exchange links between agents. A classical result for the asynchronous consensus convergence property involves to the study of infinite products of stochastic matrices [73, 101], taking the advantage of applying the indecomposable and aperiodic (SIA) property of stochastic matrices [99]. Specially, \( A = \{a_{ij}\} \) is a stochastic matrix, then it is called SIA if there exists a column vector \( v \) such that \( \lim_{k \to \infty} A^k = 1v \). An theoretical conclusion is that the system equation (1.5) solves a consensus problem if the union of the communication graphs have a spanning tree, given bounded communication delays.

On the other hand, the time-switching communication graph takes into consideration of a random network [41, 80, 100]. The connectivity in probability of a random network may offer some uncertainty assumptions in wireless communication. For such a case, the general consensus linear system is investigated in [80].
Let $\mathbb{Q} = \{ A_1, A_2, \cdots \}$ be an infinite set of $n \times n$ SIA matrices, and denote $E A_k$ is the average weighted matrix at time $k$. Suppose that eigenvalues of average weighted matrix satisfying $0 \leq |\lambda_n(E A_k)| \leq \cdots \leq |\lambda_2(E A_k)| \leq |\lambda_1(E A_k)| = 1$. Then if $|\lambda_2(E A_k)| < 1$, then the discrete-time linear dynamical system reaches consensus asymptotically.

However, it is not easy to see how the techniques used in these works can be applied to analyze the convergence rate, rather than only giving the sufficient condition of asymptotically consensus.

Consensus filtering provides a way of computing such aggregates in a distributed manner. The consensus filter was originally introduced by Olfati-Saber and Shamma [66]. The algorithm computes aggregates of sensor measurements by passing local agent information between nearest neighbors. The computed aggregates are distributed across all agents and ”consensus” occurs when all agents agree on the same estimate for the aggregated statistic. The novelty in our work is to introduce and analyze the external inputs $u_i$ of swarm dynamics in equation
generated by consensus filter. In other words we study the interconnection of a swarm with a consensus filter as shown in figure 1.1. In this work, the consensus filter generates a collective estimate of the swarm’s center and agents use that estimate to guide their movements. The primary question addressed in this work concerns the cohesiveness of the swarm under consensus and the level of consensus achieved. Moreover, how network throughput limitation impact on the convergence of consensus filter is explored.

1.2.4 TestBed

Some laboratory platforms have been developed to conduct the interdisciplinary study on distributed computation, wireless communication and coordinated control. In what follows, we reviewed some typical testbeds for multiple robotic vehicle applications, in which formation control, mapping/tracking trajectory, and obstacle avoidance are of main interests.

The multi-vehicle wireless testbed (MVWT) [16, 19] was designed at Caltech to test control algorithms for single or multiple vehicle control problems. A widely considered matter of a multiple vehicle system is formation control wherein vehicles must maintain their positions to form some global “shape”. The goal of MVWT is to investigate the difficulty of the decentralized control schemes. The testbed consists of eight robotic vehicles which are capable of on-board sensing, communication and computation. Every vehicle has a particular “hat”, which are of lace pattern (staggered belt-shaped stripe). The on-board sensor is an ultrasonic range-finder, so that each vehicle can determine the relative distances to others through sonar measurements. Four cameras are mounted on the ceiling to observe vehicles’ locations and orientations, where each vehicle is identified by the
corresponding hat. Therefore, those cameras act as remote sensors. Based on data from remote and on-board sensors, vehicles can make control decisions by on-board computer. The computer outputs motor commands through a USB interface. The computer is also responsible for intra-vehicle communications through an IEEE 802.11b wireless LAN. The “leaderless” control strategy [26] and the consensus problem [64] have been experimented with this testbed.

Another multiple vehicle testbed [43, 44] was built at MIT, which is called the Rover/Blimp testbed. The rovers are ActiveMedia’s P3-AT robotic vehicles. A Sony VAIO laptop computer is mounted on the rover processes sensor data, and performs the low-level control. Meanwhile, the high-level planning is done off-board by a Dell laptop computer. A wireless LAN bridges the on-board and off-board laptops. On-board sensors with the Arcsecond Constellation 3Di indoor laser GPS device provide accurate positions for each rover in a global coordinate frame. Four 7ft diameter blimps carry other VAIOs floating in the air. The blimps are designed to inspect the movement of rovers.

The DMPC techniques [76, 77] are implemented in the Rover/Blimp testbed. The motor control and the prediction tasks are run on-board, and the trajectory planning is done off-board to guide vehicles to a destination by a specific time. The off-board planners monitor the uncertainty of environment through measurements of blimps, and dynamically output feasible way-points actions to vehicles.

The multi-robot cooperation testbed is being developed jointly by Georgia Institute of Technology and Carnegie Mellon University. Team behaviors studied by this testbed include individual motion control and path planning for obstacle avoidance through team cooperation. In [13, 79], a robot soccer game is presented to demonstrate the team behaviors. A cooperative team is implemented using
Sony Quadruped robot platforms, each of which is a four-legged robot equipped with an IR proximity sensor and a single color camera. Another camera mounted on the ceiling captures the global image of the team. An on-field computer provides motion commands for each robot platform via wireless radio links. Behavior-based architecture [5] is employed for determining low-level actions.

As part of this work we built our own robotic vehicle testbed to examine the robotic system performance with the communication resource limitations. First we studied communication logics and distributed cooperation for formation control. Then we implemented the consensus algorithm in the tested.

Those research facilities enable researches to develop, build and test distributed control applications in a real-world setting. However, building and maintaining these environments are expensive, in terms of time and money, even for our robotic vehicle testbed. Thus, many researchers are not able to work in them. Moreover, developing and debugging a real application implies huge work burdens for a theoretical-oriented research. Therefore, a multiple-robot simulator is a significantly useful tool that simplifies programming developments.

Player/Stage project, http://playerstage.sourceforge.net, is a popular public domain tool for multi-robot and distributed sensor systems. The Player/stage project started at the University of Southern California in the late nineties to address an internal need of interfacing and simulation for multi-robot systems, and moved to Sourceforge in 2001. The Player/Stage project is a free software for robotics research and education. Currently, there is a pool of active developers, consisting of people working at universities and research institutions all around the world [17, 35, 46, 98].

The project has two major components: Player is a distributed device reposi-
tory server for robots, sensors and actuators. Client control programs connect to Player over TCP sockets, reading date from sensors, writing commands to actuators, and configuring devices on the fly. Because Player’s external interface is simply a TCP socket, client programs can be written in any programming language that provides socket support, and almost every language does, currently available in C, C++, Tcl, Python, Java and Common LISP.

Stage and Gazebo, another component of the project, is a 2-D respectively 3-D simulator backend to the Player middleware. Stage/Gazebo simulates the multi-robot act of cooperative tasks, as equally real robots behavior in real world. The Player/Stage project experienced significant popularity, mainly due to the great benefits that come from the use Player which supports multiple robot platforms and Player/Stage project runs on many UNIX-like platform.

In the past years there have been continuous efforts to develop the multiple-robot simulators, in order to enable rapid development of controller that will drive real robots later, and also to enable robot experiments without access to the real hardwares. Webots [67, 84] is a commercial simulator to support several commercial robots, such as K-Team Kephera and Koala robots, Sony Aibo robots, and Fujitsu HOAP-2 humanoid robots. Robots inside Webots can be controlled by writing a webot controller. Delta 3D [83] is another open source simulation engine, which integrated mainly for military applications developed. Urban Search And Rescue Simulation (USARSim) [4, 94] was originally established aiming to Urban Search and Rescue simulation. Currently, the simulator grows to support general propose multi-robot applications, and is selected by RoboCup Virtual Robots competition.

In this work, we build a formation control simulation base on Player/Stage tool.
We developed a virtual communication device for Player/Stage free software, to simulate the message broadcast and relay in practical wireless networks.

1.3 Statement of Contribution

Contributions of this work are briefly introduced in the following paragraphs, which are sorted by chapter and provide an outline of the dissertation.

Chapter 2 In this chapter, we focus our interests on studying the optimal communication logic. In the case that there is insufficient or prohibitively expensive communication resources for frequent information exchange, the estimation techniques have been applied to cut down the required communications. That is, each agent has a dynamical model of its neighbors, so it has the ability to estimate the neighboring agents’ state, and to use these estimates to control its behavior. With the estimation technique used to reduce communications, an efficient communication logic is needed to guarantee system performance.

A communication logic is a protocol that each agent uses to decide when it should broadcast its state information to its neighbors. The communication logic is called “open-loop” when the broadcast decisions are only a function of the last broadcast time. Otherwise, the communication logic is “closed-loop” where the broadcast decisions are conditioned on the current estimation error. However, the close-loop communication logic is not feasible to many applications. One obvious example is multi-robotic formation control, where it is difficult for individual robot to obtain the immediate estimation error without GPS equipped. The lack of global coordinates constrains each robot to the local measurements of the positions and ve-
locities relative to local coordinates. Therefore, accurate estimation error is not available instantaneously, which motivates the study of an open-loop communication logic.

This chapter examines open-loop communication logic that are “optimal” in the sense that it minimizes the entire group’s aggregate state estimation error at a low broadcast rate. Specifically, we find that the optimal open-loop communication logic is to periodically transmit an agent’s state. Broadcast decision only depends on the time elapsed since the last broadcast. For comparison, we also studied a widely-used closed loop communication logic, which schedules the broadcast if the estimation error is above some threshold. Our contribution is to present a close-form theoretical analysis of the performance of this threshold-based logic. The optimal open-loop logic performs worse than the close-loop logic as expected but within a reasonable close range. However, the comparison is under the assumption that local estimation error is available which is the optimistic for close-loop logic. Therefore, we conclude that the open-loop communication is a more practical alternative in such applications that the estimation error can not be obtained directly or accurately.

Chapter 3 This chapter studies cohesive swarming under consensus filter. Due to the inherent throughput limitations in ad hoc networks, multiple agents in a large scale system hence work in a distributed manner, such that an agent in the network is able to take actions only based on the partial network information. Data aggregation problem is raised in order to achieve a common objective. Consensus filtering provides a way of computing such an aggregation in a distributed way, which allows individual agents to pass lo-
cal information between nearest neighbors and eventually agree on the same estimate for the aggregated statistic.

The interconnection structure of a swarm with a consensus filter is employed in this chapter. The structure is shown in figure 1.1. In this structure, the swarm dynamics apply the short-range repulsion and long-rang attraction inter-agent mutual forces. This mechanism helps assure the cohesiveness of swarm while minimizing the likelihood of agent collision. Specifically, the consensus filter generates a collective estimate of the swarm centers, and then computes the guidance direction from estimated center to a known target point. The consensus filter implicitly utilized the information from nearest neighbors and accomplished the data aggregation.

The fundamental questions answered in this chapter are the cohesiveness of the swarm under consensus and the level of consensus archived. The main contribution of this chapter is to derive the detailed stability analysis of multiple agent swarm under consensus, and further to establish the uniform ultimate bounds on the swarm size. Then it goes on indicate that adding integral action dramatically improves the level of the consensus. In particular, we found out that if communication graph is regular, then the swarm achieves perfect consensus with integral action. Moreover, the computer simulations of the interconnected system are used to verify the theoretical analysis.

**Chapter 4** An important challenge of multiagent consensus is possible time delays resulted from the limited resource communication medium. This information delay may lead to an unstable group coordinated behavior. A critical observation is that certain communication graphs would be feasible to achieve the
group agreement behavior rather than the fully-connected one, though it is well known that consensus rate increases with network connectivity under no-delay assumption.

Unfortunately, available analysis of convergence rate of consensus with time delay is relatively rare and only for some restrictive system models. Even so, most of them only provide the sufficient conditions on reaching consensus asymptotically.

This chapter examines the convergence behavior of consensus filter under such throughput limitations. In this chapter, we consider a time-slotted frequency division multiple access (FDMA) network assuming a regular network. Under these assumptions we proposed two types of consensus filter models, *synchronous* and *asynchronous* consensus filter with two formats of delay, correspondingly. The synchronous filters work in principle wherein individual agents only are allowed to update their states after receiving all neighbors information, while the asynchronous filters update their estimate state as long as any delayed information is received. The main contribution of this chapter is to characterize the convergence rate of the two types of consensus filters under different information delay models, and solved the optimal levels of communication connectivity which maximize the consensus convergence rate, respectively. We showed that the asynchronous consensus assures the system stability regardless of the network connectivity, while relative dense connectivity possibly causes the synchronous consensus system unstable. On the other hand, the synchronous scheme is able to achieve the same $\epsilon$-consensus in less iterations, compared to asynchronous scheme. The optimal connectivity is again investigated to minimize the energy required
to achieve $\epsilon$-consensus. The work here in this chapter provides a fundamental guidance for building appropriate communication network limited to the throughput.

**Chapter 5** This chapter introduces our robotic vehicle simulator and testbed we are using to study the coordination behavior of a multiple-agent system. The development of the simulator gives us the ability to put to test the coordinated control algorithm proposed in chapter 3 without accessing to the real hardware and environment, which dramatically cuts down the time and energy-consuming on complicated programming. In addition to the software simulation system, we also built a hardware testbed. The testbed is specific helpful in studying how swarms of autonomous robots can coordinate their behavior to ensure stability while moving towards a specified target.

The simulation software is developed based on the Player/Stage project. Though being comprehensive in supporting many different types of devices and sensors, there is no device currently supported that can do real wireless data exchange. We developed virtual device, which is capable of simulating the package exchange among the distributed nodes via a wireless network. Based on this device, we are able to implement and study the performance of the consensus control algorithm of a multi-robot system in a distributed manner. More importantly, in the simulation software, each simulated robot is treated as an independent entity and runs in an independent process, excluding a central scheduler, which is impossible for Matlab simulation. Correspondingly, the simulation system is conveniently implanted into the real testbed with requiring a little effort.

The hardware testbed uses two types of robotic systems: Mica-KoalaBot and
Pioneer robot swarm. In Mica-KoalaBot system, Berkeley Mica2 mote works as Koala robots’ on-board computation and communication unit to support communicating over an ad hoc radio network. While in Pioneer robot system, ActiveMedia robots are controlled by a on-board x586 embedded PC, communicating over a wireless 802.11b LAN. The Mica-KoalaBot system is suitable for the laboratory research because of its low cost. The Berkeley Mica2 mote is used as the on-board device rather than a PC. However, the relatively much less memories and power supply place the restriction on the complexity of control algorithms. The Pioneer robot system is powered by three 12 Volt Gel batteries, thereby providing enough power for the robot and the on-board computation unit. Embedded PC is equipped as the computation unit to greatly enhance the performance. The testbed hence, enables us to better understand the implementation issues of a distributed algorithms in multi-robot systems.

**Chapter 6** In the last chapter, we summarize the contribution in this work and point out the further extensions and possible research directions are provided.
CHAPTER 2

COMMUNICATION LOGICS FOR THE DISTRIBUTED CONTROL OF MAS

2.1 Overview

Communications among multiple agents are essential to support coordination control. With limited communication energy, an estimation approach is used by individual agents to reduce the amount of broadcasts. That is, every agent knows dynamical models of the neighbors, so that it is able to estimate a neighboring agent’s state between consecutive broadcasts from that agent. Therefore, there is a trade-off between the amount of information exchanged and the achievable performance. It is assumed that medium-access control (MAC) protocols guarantee collision-free broadcasts. The interesting issue here is how to minimize the communication cost while assuring required system performance. In this chapter, we examine communication logics, specifically open-loop communication logic, to optimize the MAS coordinated performance.

The organization of this chapter is as follows. We introduce communication logic concepts and review related work briefly in Section 2.2. The mathematical model is formally presented in Section 2.3. The optimal open-loop communication logic is investigated in Section 2.4, where the performance is evaluated in Section 2.5. We also proposed an approach to analyze close-loop communication logic in Section 2.6. Section 2.7 presented simulation results for communication logic.
2.2 Introduction

This chapter studies a system consisting of multiple discrete-time dynamical subsystems (also called agents) that must coordinate their local behaviors in pursuit of a global objective. Each agent measures its local state and broadcasts this state to all members of the group with a specified cost of $\lambda$. It is assumed that each agent has a dynamical model of its neighbors, so it can estimate a neighbor’s local state in between consecutive broadcasts from that neighbor. A communication logic is a protocol that each agent uses to decide when it should broadcast its state information to the group. We say the communication logic is open-loop if the broadcast decision is not related to the current state of the system. We say the communication logic is closed-loop if the broadcast decision is conditioned on the current state of the system. This chapter examines open-loop communication logics that are “optimal” in the sense that they minimize the average error in an agent’s estimate of its neighbor’s state discounted by a communication cost. The chapter’s main result proves that optimal open-loop logics require agents to periodically broadcast their state across the group. We then experimentally compare the performance of this optimal open-loop logic against a recently proposed optimal closed-loop logic [104].

In our framework, every agent uses an estimator to predict its neighbor’s state in between consecutive broadcasts from that neighbor. Our problem, therefore is similar to that considered in [108]. As to communication logics examined in [108], an individual agent decides to broadcast when the local estimation error exceeds a given threshold. This “threshold-based” communication logic is closed-loop because the broadcast decisions are made on the basis of the estimator’s performance. Yook [108] investigated the system performance achievable under
this threshold-based logic. A stochastic threshold-based communication logic is presented in [103]. In [103], the broadcast decision is a Poisson process whose rate depends on the estimation error. Both performance measures used in [103, 108], however, were not discounted by the communication cost. An optimized communication logic problem is presented in [104], which optimizes the mean square estimation error discounted by the communication cost. The optimal closed-loop decision executes under a deterministic threshold-based manner, in which the agent broadcasts when the measured estimation error exceeds a specified level.

The closed-loop logic studied in [104] requires that each agent be able to measure its local estimation error in real time. There are, however, many applications where this may not be possible. One obvious situation occurs in multi-robotic formation control. In this application, an individual robot only has local measurements of its position and velocity relative to a local coordinate frame. The robot’s knowledge of its error relative to a global coordinate frame must be obtained from remote sensors observing the robot’s movements relative to its neighbors. In this situation, it may be impossible for the individual agents to make broadcast decisions on the basis of their current estimation error, since they can’t observe that error locally and immediately. In these applications, it may make more sense to use an open-loop communication logic.

This chapter, therefore, studies “optimal” open-loop communication logics that minimize the weighted sum of the estimation error discounted by the broadcast cost. In particular we find that the optimal open-loop communication logic requires periodic transmission of an agent’s state. Unlike, the logic considered in [104], our communication logic does not broadcast on the basis of the current state estimation error. Broadcast decisions are solely based on the time since the
last broadcast. A simulation comparison shows that the difference between the periodic-based logic performance and the threshold-based logic of [104] can be relatively small.

2.3 Control-communication Model

Consider a set of \( N \) interconnected discrete-time feedback control systems in which the \( i \)th subsystem’s state, \( x_i \), satisfies the following difference equation,

\[
x_i[k + 1] = Ax_i[k] + B \sum_{j=0, j \neq i}^{N-1} \hat{x}_j^{(i)}[k] + w_i[k]
\]  

(2.1)

where \( x_i[k] \) is in \( \mathbb{R}^n \) and \( x_i[0] = 0 \). \( A \) and \( B \) are matrices of appropriate dimension. \( w_i[k] \) is a zero-mean white noise process with variance \( \sigma^2 \) and \( \mathbf{E}[w_i[k]w_i^T[k]] = \sigma_a^2 \mathbf{I} \) (\( \sigma_a = \frac{\sigma}{\sqrt{n}} \)). The \( i \)th agent’s estimate of the \( j \)th agent’s state is denoted as \( \hat{x}_j^{(i)} \).

Let \( u_j[k] \in \{0, 1\} \) denote agent \( j \)’s decision at time \( k \) to broadcast its state \( x_j[k] \) to all other agents in the system. In particular, we let \( u_j[k] = 1 \) if agent \( j \) broadcasts its state and let it be zero otherwise. Assuming that \( \hat{x}_j^{(i)}[0] = x_j[0] \), then the \( i \)th agent’s state estimate for neighbor \( j \) satisfies the following difference equation

\[
\hat{x}_j^{(i)}[k + 1] = A\hat{x}_j^{(i)}[k] + B \sum_{\ell=0, \ell \neq j}^{N-1} \hat{x}_\ell^{(i)}[k]
\]  

(2.2)

if the \( j \)th agent’s control decision is to stay quiet (\( u_j[k] = 0 \)). If the \( j \)th agent broadcasts its state (\( u_j[k] = 1 \)) then the \( i \)th agent’s estimate of agent \( j \)’s state
satisfies the difference equation

\[
\hat{x}_j^{(i)}[k + 1] = A\hat{x}_j^{(i)}[k] + B \sum_{\ell=0, \ell \neq j}^{N-1} \hat{x}_\ell^{(i)}[k]
\]  

(2.3)

The \(i\)th agent’s error in estimating the \(j\)th agent’s state is denoted as

\[
\tilde{x}_j^{(i)}[k] = x_j[k] - \hat{x}_j^{(i)}[k]
\]

Since the \(i\)th agent only has ability to observe its own state \(x_i\), the estimation error \(\tilde{x}_j^{(i)}\) is not available at the \(i\)th agent. In order to predict the estimation error at the \(i\)th agent, the \(j\)th agent imitates the estimation processes in equation (2.2) and (2.3) like,

\[
\hat{x}_j^{(j)}[k + 1] = A\hat{x}_j^{(j)}[k] + B \sum_{\ell=0, \ell \neq j}^{N-1} \hat{x}_\ell^{(j)}[k]
\]  

(2.4)

and,

\[
\hat{x}_j^{(j)}[k + 1] = Ax_j[k] + B \sum_{\ell=0, \ell \neq j}^{N-1} \hat{x}_\ell^{(j)}[k]
\]  

(2.5)

The \(j\)th agent uses the knowledge of the estimation error \(\tilde{x}_j^{(j)}[k] = x_j[k] - \hat{x}_j^{(j)}[k]\) to make the broadcast decision \(u_j[k]\).

Subtracting the estimator equation (eqn’s 2.4 and 2.5) from the true state equation (2.1) yields the following equation for the state estimation error, \(\tilde{x}_j^{(j)}\),

\[
\tilde{x}_j^{(j)}[k + 1] = \begin{cases} 
    A\tilde{x}_j^{(j)}[k] + w_j[k] & \text{if } u_j[k] = 0 \\
    w_j[k] & \text{if } u_j[k] = 1
\end{cases}
\]  

(2.6)
It is assumed that the $j$th agent imitates the estimator at the $i$th agent well without any error. So it is said that $\tilde{x}_j^{(j)} = \tilde{x}_j^{(i)}$.

Let $\{u_j[k]\}$ denote the sequence of broadcast decisions made by the $j$th agent and consider the finite-horizon cost functional,

$$J[\pi_j | T] = \mathbb{E} \left[ \sum_{k=0}^{T-1} \left( \tilde{x}_j^{(i)}[k]^T \tilde{x}_j^{(i)}[k](1 - u_j[k]) + u_j[k] \lambda \right) \right]$$

(2.7)

where $\lambda$ is the stage cost for broadcasting across the network, $T$ is the horizon’s length, $\pi_j = \{u_j[k]\}_{k=0}^{T-1}$. Our problem is to find the communication decisions $\pi_j$ that minimize the cost functional $J[\pi_j | T]$ for a given $T$.

The optimality problem posed here includes both communication and estimation performance penalties. Specifically, the per-instant cost consists of a quadratic term on the estimation error and a linear term on communication cost.

2.4 Open-loop Communication Logic

In the open loop communication logics, $u_j[k]$ is independent of the current value of estimation error $\tilde{x}_j^{(i)}[k]$. Therefore, we consider the finite-horizon cost functional,

$$J[\pi_j | T] = \sum_{k=0}^{T} \left( (1 - u_j[k]) \tilde{P}_j[k] + \lambda u_j[k] \right)$$

(2.8)

where,

$$\tilde{P}_j[k] = \mathbb{E} \left[ (\tilde{x}_j[k])^T (\tilde{x}_j[k]) \right]$$

(2.9)
is the variance of the estimation error at time $k$. We drop the $(i)$ superscript on the variance of estimation error $\hat{x}_j^{(i)}[k]$ because each agent has the identical estimator for the agent $j$.

Let the sequence $\{k_i\}_{i=1}^M$ denote the time instants when agent $j$ transmits its state, where $0 \leq k_i \leq T - 1$ for $i = 1, \ldots, M$. Denote the interval between $k_i$ and $k_{i+1}$ as the estimator’s $i$th stage and let the stage cost be defined as

$$C_j(m_i) = \lambda + \sum_{d=0}^{m_i-2} \hat{P}_j[k_i + d]$$

(2.10)

where $m_i = k_{i+1} - k_i$ is the interval between consecutive transmissions. The total cost over the horizon $[0, T - 1]$ may therefore be written as

$$J[\pi_j | T] = \sum_{i=1}^M C_j(m_i)$$

(2.11)

Our sequence of control decisions, $\{u_j[k]\}_{k=0}^{T-1}$ may therefore be characterized by the sequence $\{m_i\}_{i=1}^M$. The following lemma provides a useful expression for the stage cost.

**Lemma 2.4.1** The stage cost $C_j(m_i)$ in equation (2.10) is

$$C_j(m_i) = \lambda + \sigma_n^2 \sum_{r=0}^{m_i-2} (m_i - 1 - r)Q_r$$

(2.12)

where $Q_r = \text{trace } [(A^r)^T(A^r)]$.

**Proof:** The following proof drops the superscript, $(i)$, on the estimation error for notational convenience. For $0 \leq d \leq m_i - 2$, the estimation error variance may
be rewritten as

\[ \tilde{P}_j[k_i + d] = E\left[ (\tilde{x}_j[k_i + d])^T (\tilde{x}_j[k_i + d]) \right] \]

\[ = E\left[ \|A\tilde{x}_j[k_i + d - 1] + w_j[k_i + d - 1]\|^2 \right] \]

\[ = \sum_{r=0}^{d} E\left[ \|A^r w_j[k_i + d - r]\|^2 \right] \]

\[ = \sum_{r=0}^{d} \text{trace}\left( (A^r)^T (A^r)E[w_j w_j^T] \right) \]

\[ = \sigma_a^2 \sum_{r=0}^{d} Q_r \]

Substituting the above expression for \( \tilde{P}_j[k_i + d] \) into equation (2.10) yields,

\[ C_j(m_i) = \lambda + \sum_{d=0}^{m_i-2} \tilde{P}_j[k_i + d] \]

\[ = \lambda + \sigma_a^2 \sum_{d=0}^{m_i-2} \sum_{r=0}^{d} Q_r \]

\[ = \lambda + \sigma_a^2 \sum_{r=0}^{m_i-2} (m_i - 1 - r)Q_r \]

which completes the proof.

The following theorem shows there exists an optimal \( m^* \) that minimizes the average stage cost \( C_j(m_i)/m_i \).

**Theorem 2.4.1** If we let \( C_j(m_i) \) denote the average stage cost associated with a
given interval $m_i$, then there exists a unique interval $m^*$ such that,

$$\frac{C_j(m^*)}{m^*} \leq \frac{C_j(m_i)}{m_i}$$

for all $m_i \neq m^*$.

**Proof:** From lemma 2.4.1, we know that $\frac{C_j(m_i)}{m_i} \leq \frac{C_j(m_i+1)}{m_i+1}$ if and only if

$$(m_i + 1)\left(\lambda + \sigma_a^2 \sum_{r=0}^{m_i-2} (m_i - 1 - r)Q_r\right) \leq m_i \left(\lambda + \sigma_a^2 \sum_{r=0}^{m_i-1} (m_i - r)Q_r\right)$$

which can be rewritten as

$$\lambda \leq \sigma_a^2 \sum_{r=0}^{m_i-1} (r + 1)Q_r \quad (2.13)$$

So $\frac{C_j(m_i)}{m_i} \leq \frac{C_j(m_i+1)}{m_i+1}$ if and only if $m_i$ satisfies inequality (2.13).

In a similar way, lemma 2.4.1 shows that $\frac{C_j(m_i)}{m_i} \leq \frac{C_j(m_i-1)}{m_i-1}$ if and only if

$$(m_i - 1)\left(\lambda + \sigma_a^2 \sum_{r=0}^{m_i-2} (m_i - 1 - r)Q_r\right) \leq (m_i) \left(\lambda + \sigma_a^2 \sum_{r=0}^{m_i-3} (m_i - 2 - r)Q_r\right)$$

which can be rewritten as

$$\lambda \geq \sigma_a^2 \sum_{r=0}^{m_i-2} (r + 1)Q_r \quad (2.14)$$

So $\frac{C_j(m_i)}{m_i} \leq \frac{C_j(m_i-1)}{m_i-1}$ if and only if $m_i$ satisfies inequality (2.14).

Let $m^*$ denote any integer that satisfies both inequality (2.13) and (2.14). Does such an integer exist and if so, is it unique? To answer this question let $M_\lambda$ and
\( \bar{M}_\lambda \) denote the set of all \( m \) that satisfy equations (2.13) and (2.14), respectively. In other words,

\[
\bar{M}_\lambda = \left\{ m \left| \lambda \leq \sigma^2_a \sum_{r=0}^{m-1} (r+1)Q_r \right. \right\}
\]

\[
\overline{M}_\lambda = \left\{ m \left| \lambda \geq \sigma^2_a \sum_{r=0}^{m-2} (r+1)Q_r \right. \right\}
\]

We let \( \underline{m} = \max \bar{M}_\lambda \) and \( \overline{m} = \min \overline{M}_\lambda \). We shall prove that \( m^* = \underline{m} = \overline{m} \).

We can easily show that \( \underline{m} \geq \overline{m} \). Let’s suppose \( \underline{m} \neq \overline{m} \), so there exists \( c > 0 \) such that \( \underline{m} - \overline{m} = c \). There are then three possibilities for the average cost at these two values of \( m \). We have that \( \frac{C_j(\overline{m})}{\overline{m}} = \frac{C_j(\underline{m})}{\underline{m}} \) or \( \frac{C_j(\overline{m})}{\overline{m}} < \frac{C_j(\underline{m})}{\underline{m}} \). The last two cases cannot occur. The inequality in the third case, for example implies that inequality (2.14) is satisfied which means that \( \underline{m} \in \bar{M}_\lambda \). But \( \overline{m} = \max \bar{M}_\lambda \) and \( \overline{m} > \underline{m} \), which means there is an element of \( \overline{M}_\lambda \) which is greater than \( \underline{m} \). This contradicts the maximal nature of \( \underline{m} \) and so the third case can’t occur. A similar argument can be used to dispose of the second case.

If the first case is true then we know that \( \frac{C_j(\underline{m})}{\underline{m}} = \frac{C_j(\overline{m})}{\overline{m}} = \frac{C_j(m)}{m} \) for any \( \underline{m} \leq m \leq \overline{m} \). In particular, let’s consider \( m = \overline{m} - 1 \). In this case we see that

\[
(\underline{m} - 1) \left( \lambda + \sigma^2_a \sum_{r=0}^{\overline{m}-2} (m-1-r)Q_r \right) = \overline{m} \left( \lambda + \sigma^2_a \sum_{r=0}^{\overline{m}-3} (m-2-r)Q_r \right)
\]

This equation can be rewritten as

\[
\lambda = \sigma^2_a \left( \sum_{r=0}^{\overline{m}-3} mQ_r + (\overline{m} - 1)Q_{\overline{m}-2} - \sum_{r=0}^{\overline{m}-3} (m-1-r)Q_r \right)
\]
which can, in turn, be simplified to

\[
\lambda = \sigma_a^2 \left( \sum_{r=0}^{m-3} (1 + r)Q_r + (m - 1)Q_{m-2} \right)
\]

\[
= \sigma_a^2 \sum_{r=0}^{m-2} (1 + r)Q_r
\]

Note that this equality implies that \( m - 1 \) is an element of the set \( \mathcal{M}_\lambda \). Since \( m - 1 < m \), then clearly \( m \) cannot be the minimal element of \( \mathcal{M}_\lambda \). So we have a contradiction and we know that \( m = m = m^* \).

The following theorem states that a periodic communication logic minimizes the finite-horizon cost. In the following statement, we drop the \( j \) subscript on the communication logic \( \pi_j \) for notational simplicity.

**Theorem 2.4.2** The problem’s cost functional, \( J[\pi_j \mid T] \) is minimized by a communication logic that periodically transmits the agent state information. The optimal period, \( m^* \), satisfies inequalities (2.13) and (2.14).

**Proof:** Let \( \pi \) denote a sequence of transmission decisions consisting of \( M \) transmissions. Let \( m_i \) denote the \( i \)th transmission interval for \( \pi \). Let \( u^* \) denote a sequence of transmission decisions consisting of \( M' \) transmissions. Let \( m_i^* \) denote the \( i \)th transmission interval for \( u^* \). Further assume that \( m_i^* = m^* \) for all \( i \) where \( m^* \) satisfies inequalities (2.13) and (2.14).

The cost achieved under \( \pi \) may be written as

\[
J[\pi \mid T] = \sum_{i=1}^{M} \frac{C_j(m_i)}{m_i} m_i
\]

From theorem 2.4.1 we know that \( \frac{C_j(m_i)}{m_i} \geq \frac{C_j(m^*)}{m^*} \) so our preceding expression for
the cost may be written as

\[
J[\bar{u} | T] \geq \sum_{i=1}^{M} \frac{C_j(m^*)}{m^*} m_i
\]

\[
= \frac{C_j(m^*)}{m^*} \sum_{i=1}^{M} m_i
\]

\[
= T \frac{C_j(m^*)}{m^*}
\]

Note that the last expression is the cost achieved under \(u^*\). So we can conclude that \(J[\bar{u} | T] \geq J[u^* | T]\), which implies that the periodic logic is optimal.

2.5 Evaluation of Periodic Communication Logic

The results in Section 2.4 show that the optimal open-loop communication logics require periodic broadcasts given a performance metric that jointly minimizes the average estimation error and communication cost. In this section, we studied the performance metric for the optimal periodic broadcast, in terms of the variance of instantaneous performance.

The instantaneous average cost over finite-horizon converges to

\[
\lim_{T \to \infty} \frac{1}{T} \sum_{k=0}^{T-1} c(\bar{x}, u)
\]

where \(c(\bar{x}, u) = \bar{x}[k]^T \bar{x}[k](1 - u[k]) + u[k] \lambda\) is the cost per-instant. In Section 2.4, we considered the following expected finite-horizon cost function to show the optimality of open-loop periodic broadcasts.

\[
J_1 = \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ \sum_{k=0}^{T-1} c(\bar{x}, u) \right]
\]  

(2.15)
In this section, we will evaluate the system performance variance for periodic broadcast. Letting

$$J_2 = \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ \left( \sum_{k=0}^{T-1} c(\tilde{x}, u) \right)^2 \right]$$

the variance of average cost is hence $J_2 - J_1^2$.

In what follows, let the broadcast period length be $m$, and let the finite horizon $T$ be approximately $Lm$ where $L$ is the number of broadcasts.

**Lemma 2.5.1** The variance of the average cost is,

$$J_2 - J_1^2 = \frac{1}{m} \left\{ \mathbb{E}[\tilde{c}_i^2] - (\mathbb{E}[\tilde{c}_i])^2 \right\}$$

where $\tilde{c}_i = \sum_{k=i}^{(i+1)m-1} c(\tilde{x}, u)$ is the cost per-period.

**Proof:** If $Lm \leq T \leq (L+1)m$, then the extra part $\Delta T = T - Lm$ is finite and will go to zero after divided by $T$, which implies,

$$J_1 = \lim_{T \to \infty} \frac{1}{Lm + \Delta T} \mathbb{E} \left[ \sum_{k=0}^{Lm} c(\tilde{x}, u) + \sum_{Lm+1}^{T-1} c(\tilde{x}, u) \right]$$

$$= \lim_{T \to \infty} \frac{Lm}{Lm + \Delta T} \frac{1}{Lm} \mathbb{E} \left[ \sum_{k=0}^{Lm} c(\tilde{x}, u) \right] + o(1)$$

$$= \lim_{T \to \infty} \frac{1}{Lm} \mathbb{E} \left[ \sum_{k=0}^{Lm} c(\tilde{x}, u) \right]$$

Let $\tilde{c}_i = \sum_{k=i}^{(i+1)m-1} c(\tilde{x}, u)$ ($0 \leq i \leq L - 1$) denote the per-period-cost. Then, the two per-period-cost $\tilde{c}_i$ and $\tilde{c}_j$ ($i \neq j$) are independent and identically distributed, since the estimation error is reset after exchanging the information.
Therefore,

\[ J_2 - J_1^2 = \frac{L}{T} \{ \mathbb{E}[\tau_i^2] - (\mathbb{E}[\tau_i])^2 \} = \frac{1}{m} \{ \mathbb{E}[\tau_i^2] - (\mathbb{E}[\tau_i])^2 \} \]

which completes the proof.

This observation enables us to analyze the variance of the system performance in terms of the statistics of cost per period. It is assumed that an agent broadcasts information at the first time instant of a period, i.e.,

\[
c_d = \begin{cases} 
\lambda & d = 0 \\
\bar{x}[im + d]^T \bar{x}[im + d] & 1 \leq d \leq m - 1 
\end{cases}
\]

According to the dynamics of estimation error,

\[
\bar{x}[im + d + 1] = \begin{cases} 
A \bar{x}[im + d] + w[im + d] & \text{if } u[im + d] = 0, \text{ and } d > 0 \\
w[im] & \text{if } u[im] = 1 
\end{cases}
\] (2.17)

the per-instant-cost except for the broadcast instant is equal to,

\[
c_d = \bar{x}[im + d]^T \bar{x}[im + d] \\
= \left( \sum_{l=0}^{d-1} A^{d-1-l} w[l] \right)^T \left( \sum_{l=0}^{d-1} A^{d-1-l} w[l] \right) \\
= \sum_{l,l'=0}^{d-1} w[l]^T (A^{d-1-l})^T (A^{d-1-l'}) w[l']
\]
Therefore, the per-period-cost could be calculated in the way as $\tau_i = \lambda + \sum_{d=1}^{m-1} c_d$,

\[
\tau_i = \lambda + \sum_{d=1}^{m-1} \sum_{l,l'=0}^{d-1} w[l]^T \left( A^{d-1-l} \right)^T \left( A^{d-1-l'} \right) w[l']
\]

\[
= \lambda + \sum_{l,l'=0}^{m-2} w[l]^T G_{ll'} w[l']
\]

where $G_{ll'} = \sum_{d=\max\{l,l'\}+1}^{m-1} \left( A^{d-1-l} \right)^T \left( A^{d-1-l'} \right)$, and the entry$(i-j)$ of the matrix $G_{ll'}$ is denoted as $g_{ij}$. At the following, we can investigate the variance of average finite-horizon cost in terms of the result of per-period-cost $\tau_i$.

**Theorem 2.5.1** The variance of the average cost in periodic communication logic is,

\[
J_2 - J_1^2 = \frac{2}{m} \sum_{l,l'=0}^{m-2} \sigma_a^4 \text{trace} \left( G_{ll'} G_{ll'}^T \right) + \sum_{l=0}^{m-2} \sigma_a^4 \text{trace} \left( G_{ll'} G_{ll'}^T \right)
\]

where $G_{ll'} = \sum_{d=\max\{l,l'\}+1}^{m-1} \left( A^{d-1-l} \right)^T \left( A^{d-1-l'} \right)$.

**Proof:** In terms of the lemma 2.5.1, the variance of average cost is rewritten as,

\[
J_2 - J_1^2 = \frac{1}{m} \left\{ E[\tau_i^2] - (E[\tau_i])^2 \right\}
\]

\[
= \frac{1}{m} \text{Var} \left[ \sum_{l,l'=0}^{m-2} w[l]^T G_{ll'} w[l'] \right]
\]

\[
= \frac{1}{m} \text{Var} \left[ \sum_{l,l'=0,l\neq l'}^{m-2} w[l]^T G_{ll'} w[l'] \right] + \frac{1}{m} \text{Var} \left[ \sum_{l=0}^{m-2} w[l]^T G_{ll'} w[l] \right]
\]

The variance consists of two parts, with $l \neq l'$ or $l = l'$. We study these two cases in the following, respectively.
• \((l \neq l')\) For \(l \neq l'\), we have the following equation due to the independent property of the random noise,

\[
E \left[ w[l]^T G_{ll'} w[l'] \right] = 0
\]

Henceforth, we have,

\[
\frac{1}{m} \text{Var} \left[ \sum_{l,l'=0,l \neq l'}^{m-2} w[l]^T G_{ll'} w[l'] \right] = \frac{1}{m} \sum_{l,l'=0,l \neq l'}^{m-2} \text{Var} \left[ w[l]^T G_{ll'} w[l'] \right]
\]

Taking a look at one item each time,

\[
\text{Var} \left[ w[l]^T G_{ll'} w[l'] \right]
\]

\[
= E \left[ (w[l]^T G_{ll'} w[l'])^2 \right]
\]

\[
= E \left[ w[l]^T G_{ll'} w[l'] w[l']^T G_{ll'}^T w[l] \right]
\]

\[
= \text{trace} \left( E \left[ (G_{ll'} w[l'] w[l']^T G_{ll'}^T w[l]) \right] \right)
\]

\[
= \text{trace} \left( (G_{ll'} \sigma_a^2 I)(G_{ll'}^T \sigma_a^2 I) \right)
\]

\[
= \sigma_a^4 \text{trace}(G_{ll'} G_{ll'}^T)
\]

\[
= \sigma_a^4 \sum_{i,j=1}^{n} g_{ij}^2
\]

so, the result of the first part of the variance turns out,

\[
\frac{1}{m} \sum_{l,l'=0,l \neq l'}^{m-2} \text{Var} \left[ w[l]^T G_{ll'} w[l'] \right] = \frac{1}{m} \sum_{l,l'=0,l \neq l'}^{m-2} \sigma_a^4 \text{trace}(G_{ll'} G_{ll'}^T) \quad (2.18)
\]

• \((l = l')\) In the case of \(l = l'\), the matrix \(G_{ll}\) is a symmetric matrix, that
is,

\[
G_{ll} = \begin{bmatrix}
g_{11} & \cdots & g_{1n} \\
\vdots & & \vdots \\
g_{n1} & \cdots & g_{nn}
\end{bmatrix}
\quad \quad w[l] = \begin{bmatrix}
w_1[l] \\
\vdots \\
w_n[l]
\end{bmatrix}
\]

The first statistical moment of variable \(w[l]^T G_{ll} w[l]\) can be computed in the following way,

\[
E[w[l]^T G_{ll} w[l]] = \text{trace} \left( E(G_{ll} w[l]w[l]^T) \right)
\]
\[
= \sigma_a^2 \text{trace}(G_{ll})
\]
\[
= \sigma_a^2 \left( \sum_{i=1}^{n} g_{ii} \right)
\tag{2.19}
\]

and the second statistical moment is,

\[
E \left[ (w[l]^T G_{ll} w[l])^2 \right] = E[w[l]^T G_{ll} w[l]w[l]^T G_{ll} w[l]]
\tag{2.20}
\]

in which

\[
w[l]^T G_{ll} w[l] = \begin{bmatrix}
w_1[l] & \cdots & w_n[l]
\end{bmatrix}
\begin{bmatrix}
g_{11} & \cdots & g_{1n} \\
\vdots & & \vdots \\
g_{n1} & \cdots & g_{nn}
\end{bmatrix}
\begin{bmatrix}
w_1[l] \\
\vdots \\
w_n[l]
\end{bmatrix}
\]
\[
= \sum_{i,j=1}^{n} w_i[l]w_j[l]g_{ij}
\]
then,

\[
\mathbb{E} \left[ \left( \sum_{i,j=1}^{n} w_i[l]w_j[l]g_{ij} \right)^2 \right] = \mathbb{E} \left[ \left( \sum_{i,j=1}^{n} w_i[l]w_j[l]w_{i'}[l]w_{j'}[l]g_{ij} \right)^2 \right] = \sum_{i,j=1}^{n} g_{ij}^2 + \mathbb{E} \left[ \sum_{i=1}^{n} w_i[l]^4 g_{ii}^2 \right] + \mathbb{E} \left[ \sum_{i,i'=1,i\neq i'}^{n} w_i[l]^2 w_{i'}[l]^2 g_{ii'} \right]
\]

\[
= \sigma_n^4 \sum_{i,j=1,i\neq j}^{n} g_{ij}^2 + 3\sigma_n^4 \sum_{i=1}^{n} g_{ii}^2 + \sigma_n^4 \sum_{i,i'=1,i\neq i'}^{n} g_{ii}g_{i'i'}
\]

Combining the equations (2.19 and 2.20) together yields,

\[
\mathbb{E} \left[ \left( w[l]^T G_{ll} w[l] \right)^2 \right] - \mathbb{E} \left[ w[l]^T G_{ll'} w[l] \right]^2 = \sigma_n^4 \sum_{i=1}^{n} g_{ii}^2 + \sum_{i,j=1}^{n} \sigma_n^4 g_{ij}^2
\]

so, the result of the second part of the variance turns out,

\[
\frac{1}{m} \sum_{l=0}^{m-2} \text{Var} \left[ w[l]^T G_{ll} w[l] \right] = \frac{1}{m} \left( \sum_{l,l'=0,l\neq l'}^{m-2} \sigma_n^4 \text{trace} (G_{ll'} G_{ll'}^T) + \sum_{l=0}^{m-2} \sigma_n^4 \text{trace} (G_{ll} G_{ll}^T) \right)
\]

(2.21)

Therefore, the variance of the average cost is given by (2.18) plus (2.21), i.e.,

\[
J_2 - J_1^2 = \frac{2}{m} \sum_{l,l'=0,l\neq l'}^{m-2} \sigma_n^4 \text{trace} (G_{ll'} G_{ll'}^T) + \sum_{l=0}^{m-2} \sigma_n^4 \text{trace} (G_{ll} G_{ll}^T)
\]
which completes the proof.

2.6 Closed-loop Communication Logic

The communication logic is closed-loop when the broadcast decision \{u_j[k]\} is conditioned on the current estimation error \(\tilde{x}_j[k]\). For comparison, we studied a “threshold-based” closed-loop communication logic here, in which a broadcast is decided if instantaneous estimation error is above some pre-defined threshold. Specifically, agent \(j\) makes a broadcast decision when the norm of estimation error \(\tilde{x}_j\) is larger than specified threshold \(e_{th}\). That is,

\[
 u_j[k] = \begin{cases} 
 0 & \text{if } \|\tilde{x}_j[k]\| < e_{th} \\
 1 & \text{if } \|\tilde{x}_j[k]\| \geq e_{th} 
\end{cases}
\] (2.22)

which implies that the decision sequence \(\{u_j[k]\}\) is decided in a deterministic manner. In [104], the authors explored the optimal closed-loop communication logic in a similar framework discussed in Section 2.3. It has been proved that the deterministic threshold-based logic is the optimal choice among closed-loop logics. However, the optimal threshold is not explicitly given in [104]. Rather than theoretically computing the cost function, the authors only studied the logic performance with different thresholds numerically. In this section, we proposed a method to analyze the performance of the deterministic threshold-based logic. The major contribution of our work is that we presented a close-form expression of the threshold-based logic cost. Therefore, the optimal threshold can be found by numerically evaluating the cost function. Our analysis greatly facilitates the implementation of the threshold logic in practical systems.

It is straightforward that with the threshold-based logic, the broadcast decision
Figure 2.1. Markov chain for the threshold-based logic sequence \( \{u_j[k]\}_{k=0}^\infty \) is a Markov chain, as illustrated in the figure 2.1. In the figure, state \( s \) stands for the length of consecutive estimations, or the interval between broadcasts. For instance, \( s = 2 \) implies that broadcast sequence \( u_j[k] = 0, u_j[k+1] = 0, u_j[k+2] = 1 \).

To simplify the notation, let \( e_s := \tilde{x}_j[k+s] \), assuming that the the agent \( j \) broadcasts its state at time \( k \) (\( u_j[k] = 1 \)). The estimation error dynamics are rewritten as

\[
\begin{aligned}
\mathbf{e}_0 &= \mathbf{w}[0] \\
\mathbf{e}_s &= \mathbf{A}\mathbf{e}_{s-1} + \mathbf{w}[s]
\end{aligned}
\]

where \( \mathbf{e}_0 \) represents the estimation error \( \tilde{x}_j \) at broadcast time. The transition probability from state \( s \) (\( \mathbf{e}_s := \tilde{x}[k+s] \)) to state \( s + 1 \) (\( \mathbf{e}_{s+1} := \tilde{x}[k+s+1] \)) is given by,

\[
P_s = \text{Prob}\{\|\mathbf{e}_{s+1}\| < e_{th} \mid \|\mathbf{e}_s\| < e_{th}\} \tag{2.23}
\]
\[ P_s = \text{Prob}\{\|e_{s+1}\| < e_{th} \land \|e_s\| < e_{th}\} \]

\[ 1 - P_s = \text{Prob}\{\|e_{s+1}\| \geq e_{th} \land \|e_{s+1}\| < e_{th}\} \]

Figure 2.2. Transition probability of Markov chain

The transition process is demonstrated in the figure 2.2.

The corresponding transition matrix \( \mathcal{P} \) of the Markov chain is,

\[
\mathcal{P} = \begin{bmatrix}
1 - P_0 & P_0 \\
1 - P_1 & P_1 \\
\vdots & \vdots \\
1 - P_s & P_s \\
\vdots & \vdots \\
\end{bmatrix}_{T \times T}
\]

and the steady-state distribution is denoted as \( \mathbf{q} = [q_0, q_1, \cdots, q_s, \cdots] \). The steady-state distribution satisfies the following equations,

\[
\begin{cases}
\mathbf{q} \mathcal{P} = \mathbf{q} \\
\sum_{i=0}^{T-1} q_i = 1
\end{cases}
\quad (2.24)
\]

For comparison, we use the same finite-horizon cost function in (2.7) to analyze the performance of the threshold-based communication logic, given in the following
Theorem 2.6.1 The finite-horizon cost of the threshold-based communication logic is,

\[
J = q_0 \lambda + \sum_{s=1}^{T-1} q_s E_s
\]

(2.25)

where

\[
E_s = \frac{\int_{\|w_{s-1}\| < e_{th}} \exp\left\{ -\frac{w_{s-1}^T \sigma_{s-1}^2 w_{s-1}}{2} \right\} \left( w_{s-1}^T A^T A w_{s-1} \right) dw_{s-1}}{\int_{\|w_{s-1}\| < e_{th}} \exp\left\{ -\frac{w_{s-1}^T \sigma_{s-1}^2 w_{s-1}}{2} \right\} dw_{s-1}} + \sigma^2
\]

and \( q_s \) is the steady-state distribution of the Markov chain, and \( w_{s-1} = w[s-1] \).

Proof: It is easy to show that \( P_s q_s = q_{s+1} \) in terms of the equation 2.24, so,

\[
q_s = \sum_{j=0}^{s-1} P_j q_0 = \Theta_s q_0 \quad \text{where,}
\]

\[
\Theta_s = \sum_{j=0}^{s-1} P_j = \sum_{j=1}^{s-1} \text{Prob}\{\|e_{j+1}\| < e_{th} \mid \|e_j\| < e_{th}\} \cdot \text{Prob}\{\|e_1\| < e_{th}\} \]
\[
= \text{Prob}\{\|e_j\| < e_{th}, \quad 1 \leq j \leq s\}
\]

The result of individual distribution \( q_s \), hence, is,

\[
q_s = \frac{\Theta_s}{\sum_{i=0}^{T-1} \Theta_i}
\]
Therefore the cost function could be computed as
\[
J = \lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[ \sum_{s=0}^{T-1} x[s] x'[s] (1 - u[s]) + u[s] \lambda \right] \\
= q_0 \lambda + \sum_{s=0}^{T-1} q_s \mathbb{E} \left[ e_s^T e_s \mid \|e_{s-1}\| < e_{th} \right]
\]

For notational simplicity, let
\[
E_s = \mathbb{E} \left[ e_s^T e_s \mid \|e_{s-1}\| < e_{th} \right]
\]

After agents exchange the information, the estimation error is reset. At the reset point, the estimation error \( e_0 = w[0] \) satisfies the distribution \( \mathcal{N}(0, \sigma^2) \). Except for the reset point, the estimation error \( e_s \)’s distribution obeys with \( \mathcal{N}(0, \sigma_s^2) \) which is denoted as \( \mathcal{N}(0, \sigma^2) \). Hence,
\[
E_s = \int_{\|w_{s-1}\| < e_{th}} \frac{1}{(2\pi)^{n/2} \sigma_{s-1}} \exp\left\{ -\frac{w_{s-1}^T \sigma_{s-1}^{-2} w_{s-1}}{2} \right\} \cdot \\
\int_{\infty} \left( A w_{s-1} + w_s \right)^T \left( A w_{s-1} + w_s \right) \frac{1}{(2\pi)^{n/2} \sigma} \exp\left\{ -\frac{w_s^T \sigma^{-2} w_s}{2} \right\} dw_{s-1} dw_s
\]

\[
E_s = \int_{\|w_{s-1}\| < e_{th}} \frac{1}{(2\pi)^{n/2} \sigma_{s-1}} \exp\left\{ -\frac{w_{s-1}^T \sigma_{s-1}^{-2} w_{s-1}}{2} \right\} \cdot \\
\left( \int_{\|w_{s-1}\| < e_{th}} \frac{1}{(2\pi)^{n/2} \sigma_{s-1}} \exp\left\{ -\frac{w_{s-1}^T \sigma_{s-1}^{-2} w_{s-1}}{2} \right\} dw_{s-1} \right)^{-1}
\]

\[
= \int_{\|w_{s-1}\| < e_{th}} \frac{1}{(2\pi)^{n/2} \sigma_{s-1}} \exp\left\{ -\frac{w_{s-1}^T \sigma_{s-1}^{-2} w_{s-1}}{2} \right\} \left( w_{s-1}^T A^T A w_{s-1} + \sigma^2 \right) dw_{s-1}
\]

which completes the proof.

Therefore, the optimal threshold \( e_{th}^* \) for the threshold-based communication
logic is given by minimizing the cost function, i.e.,

\[ e^*_\text{th} = \arg \min_{e_{\text{th}}} J \tag{2.26} \]

where \( J \) is given by (2.25). Although the close-form expression of optimal threshold is not available due to the mathematical complexity, \( e^*_\text{th} \) can be numerically solved by evaluating (2.25) off-line easily.

2.7 Simulation

In this section, simulation results are presented to compare the performance for three different communication logics, including: (i) the periodic protocol studied in Section 2.4, (ii) a random logic in which the agent transmits with a probability \( p \) at each time instant, and (iii) the threshold logic in [104]. In what follows, the results are for a scalar system in which \( A = 0.95 \), with a process noise variance \( \sigma^2 = 6 \), and a communication cost \( \lambda = 100 \). Theorem 2.4.1 was used to compute the optimal period, which was found to be \( m^* = 7 \).

Figure 2.3 shows the average cost \( \frac{J}{T} \) versus the probability that an agent transmits. The probabilities were obtained by dividing the number of transmissions over the total time, which reflect the expected amount of transmissions. For example, for the optimal periodic logic using \( m^* = 7 \), this means the minimum cost should occur at a transmission probability of \( \frac{1}{m^*} = 0.142 \). The minimum cost is achieved by our proposed periodic logic and that this occurs at the predicted probability level, which is justified by Figure 2.3. The proposed periodic logic performed slightly better than the random logic and performed slightly worse than the threshold-based logic.

It is not unexpected that the optimal open-loop communication logic is worse
than the optimal closed-loop logic. There are many applications, however, where a closed-loop logic cannot be applied because individual agents cannot directly or accurately measure its estimation error. In these cases, the open-loop periodic logic is a practical alternative whose performance are only slightly worse than that of idealistic closed-loop logics, and much more robust to error estimation.

The transmission rate of the threshold-base communication logic is conditioned on the specified threshold value (refer to the left side of Figure 2.4). Adjusting the different thresholds results in the different information exchange rate. The left side figure shows that the average cost increases rapidly when the broadcast threshold deviates from the optimal point. The optimal threshold values swing within the range $[6, 6.5]$ from simulations. The swing stems from two reasons: 1) the broadcast decision depends on instantaneous exogenous disturbance which is unpredictable; 2) in limited simulation time, the optimal threshold has not converged due to insufficient statistical sample. Current simulation time is 50000 time steps.

Figure 2.3. Average cost in different communication logics
Figure 2.4. Average cost versus different communication logic variables steps. The right figure illustrates the periodic logic costs versus period lengths. It can be seen that the periodic logic is more robust to logic cost change than the threshold-based logic.

Variance of instantaneous performance for the periodic logic vs noise variance is plotted in Figure 2.5. The dot on each curve stands for the optimal information exchange rates, which is larger with higher noise variance. The result implies that optimal broadcast period could approach the desired system performance without dramatical fluctuation.

Figure 2.6 compares the simulated costs with the theoretical results for different communication logics. The results show that the theoretical results match the simulation results well, which justifies the theoretical analysis. Therefore, for threshold-based logic, the optimal threshold can be obtained off-line. The optimal solution also matches the simulation result shown in [104].
Figure 2.5. Variance of open-loop logic performance

Figure 2.6. Simulated and theoretical performances
CHAPTER 3

COHESIVE SWARMING UNDER CONSENSUS

3.1 Overview

In ad hoc networks, apparently, the information is multi-hopped from one node to another node using closer neighbors, so that there is a growing communication load for providing high system performance. In order to decrease the communication load, distributed estimation plays of course an important role in the networks associated with distributed monitoring, tracking, and control. We have witnessed, mainly due to the reasons mentioned in chapter 1, recently studied consensus filter is a practically useful tool to tackle distributed estimation problems for large scale size of MAS [2, 61, 66]. This chapter studies the cohesion of multi-agent swarms moving under the control of consensus filters. This chapter’s main result shows that swarming under consensus is cohesive. We establish specific bounds on the degree of cohesion and consensus as a function of the attraction/repulsion fields, swarm size, and connectivity in the communication network. We also prove that proves that if the swarm’s communication graph is regular, then the introduction of integral action into the consensus filter achieves perfect consensus regardless of swarm size.

The remainder of the chapter is organized as follows. The problem statement is stated in section 3.3. The concepts of swarm error and consensus error are
introduced in section 3.4. The swarm and consensus filter stability analysis are presented in section 3.5 and section 3.6. We study the distribution of swarm agents in section 3.7. We then study the behavior of the consensus filter under integral action in section 3.8.

3.2 Introduction

There has recently been considerable activity studying the swarming [37][93] of autonomous unmanned vehicles (AUV) or mobile agents. Most of this effort has used a Lagrangian framework [55] [56] which focuses on the relationship between individual agents. Nearly all of these papers assume the swarm consists of agents that have the same underlying vehicular dynamics. Agent movement is driven by a command that passes through either a single integrator [9][32][53] or double integrator dynamic [89]. The command input is usually the gradient of a potential field. This potential field can be automatically generated from proximity sensors detecting neighboring agents and obstacles. Potential fields associated with obstacles cause agents to move away from the obstacle. Potential fields generated by neighboring agents are based on long-range attraction and short-range repulsion between agents. This mechanism helps assure the cohesiveness of the swarm while minimizing the likelihood of agent collisions.

Potential fields, however, may also arise from virtual objects [60] that are not directly sensed by any individual agent. For example, a group of mobile robots attempting to find the source of a chemical plume, must use the aggregate of all local measurements of chemical concentration to determine the best direction for the swarm to move towards. The “source” of the chemical plume may be thought of as a virtual position that generates a potential field which draws all agents
in the swarm toward that location. An individual agent’s movements, therefore, are no longer determined locally by that agent’s sensors. Those movements are guided by a vector that is the result of aggregating sensor information from agents throughout the entire swarm.

Consensus filtering provides a way of computing such aggregates in a distributed manner. The consensus filter was originally introduced by Olfati-Saber and Shamma [66]. The algorithm computes aggregates of sensor measurements by passing local agent information between nearest neighbors. The computed aggregates are distributed across all agents and “consensus” occurs when all agents agree on the same estimate for the aggregated statistic. In recent years, there has been considerable interest in studying consensus algorithms as a powerful tool to govern the multiple-agent coordination [18][57][42][64] [66][102][74][80].

This chapter focuses on the use of consensus in swarm guidance and control. In particular, we study the interconnection of swarm dynamics with a consensus filter as shown in figure 3.1. The swarm dynamics used in this chapter employ short range repulsion and long range attraction functions, similar to [56] and [32], to prevent agent collisions. The individual velocity is generated by integrating the mutual forces related to the sensed distance between neighboring agents. The consensus filter is based on the system introduced in [66]. In this study, the consensus filter estimates the swarm center and then computes the guidance direction from estimated center to a known target point.

The chapter derives uniform ultimate bounds on the swarm size and level of consensus through Lyapunov-based methodologies, similar to that done in [32] for the swarm and [66] for the consensus filter. The bounds are expressed as a function of the attraction/repulsion strength, number of network agents, and com-
munication network connectivity. These results establish that the swarm is indeed cohesive under consensus filtering, though the level of consensus is a function of swarm size. We then go on to demonstrate that perfect consensus can be achieved through the introduction of integral action into the consensus filter.

3.3 Swarm Dynamic Model

Consider a swarm of $N$ dynamical agents that exchange information over a multi-hop communication network. Each agent is characterized by two types of states; its physical state representing the agent’s position in the real world and its consensus state representing the agent’s estimate of the swarm’s geometric center. The physical state of the $i$th agent at time $t$ is denoted as a vector $x_i(t)$ in Euclidean $n$-space, $\mathbb{R}^n$. The trajectory of the $i$th agent’s physical state is denoted by the function $x_i : \mathbb{R}^+ \rightarrow \mathbb{R}^n$ which satisfies the ordinary differential equation

$$\dot{x}_i(t) = u_i(t) + \sum_{j \sim i} g(||x_i(t) - x_j(t)||)(x_i(t) - x_j(t)) \quad (3.1)$$
for \( i = 1, \ldots, N \). The vector \( u_i(t) \in \mathbb{R}^n \) is an external input and \( g : \mathbb{R}^+ \to \mathbb{R} \) is a function from the positive real line, \( \mathbb{R}^+ \), into the real line, \( \mathbb{R} \). We will use the notation \( g_{ij} \) to denote \( g(\|x_i - x_j\|) \) and we let \( \sum_{j \sim i} x_j \) to denote \( \sum_{j=1,j\neq i}^N x_j \).

The summation in equation 3.1 represents long range physical interactions between agents. We assume that this interaction can be decomposed into a repulsive and attractive component. In particular, if we let \( \rho : \mathbb{R}^+ \to \mathbb{R}^+ \) and \( \alpha : \mathbb{R}^+ \to \mathbb{R}^+ \) denote the repulsion and attraction functions, then \( g \) may be written as

\[
g(r) = \rho(r) - \alpha(r)
\]

for any \( r \in \mathbb{R}^+ \). This chapter restricts its attention to attraction and repulsion functions of the form

\[
\rho(r) = \frac{\rho_0}{r}, \quad \alpha(r) = \frac{\alpha_0}{r}
\]

for any \( r \in \mathbb{R}^+ \) and where \( \rho_0 \) and \( \alpha_0 \) are positive constants.

The consensus state of the \( i \)th agent at time \( t \) is denoted as a vector \( \hat{x}_i(t) \in \mathbb{R}^n \). The trajectory of the \( i \)th agent’s consensus state is denoted by the function \( \hat{x}_i : \mathbb{R}^+ \to \mathbb{R}^n \) which satisfies the consensus filter equation

\[
\dot{\hat{x}}_i(t) = (x_0(t) - \hat{x}_i(t)) + \sum_{j \sim i} A_{ij}(\check{u}_j(t) - \hat{x}_i(t)) + \sum_{j=1}^N A_{ij}(\hat{x}_j(t) - \hat{x}_i(t))
\]

for \( i = 1, \ldots, N \). The vector \( x_0(t) \in \mathbb{R}^n \) is the state of the target at time \( t \). The coefficient \( A_{ij} \) is the \( ij \)-th components of the matrix \( \mathbf{I}_n + \text{Adj}(G) \) where \( \mathbf{I}_n \) is an \( n \times n \) identity matrix and \( \text{Adj}(G) \) is the adjacency matrix of the graph \( G \). The graph \( G \) models the communication connectivity within the swarm. Agent \( j \) is
able to transmit its consensus state \( \hat{x}_j \) and an input \( \hat{u}_j \) to agent \( i \) if and only if \( A_{ij} = 1 \).

Figure 3.1 shows that the entire swarm may be viewed as an interconnection of the swarm dynamics (equation 3.1) and the consensus filter (equation 3.3). The swarm dynamic’s input from the \( j \)th agent to the consensus filter’s \( i \)th agent is the position of the \( j \)th agent, in other words \( \hat{u}_j = x_j \). The consensus filter’s input from the \( j \)th agent to the swarm dynamic’s \( j \)th agent is the \( j \)’th agent’s estimate of the swarm center (consensus state) relative to the target, in other words \( u_j = x_0 - \hat{x}_j \). The consensus filter is trying to estimate the center of the swarm and the swarm is using those estimates to guide the swarm toward the target. The overall dynamics of this system may therefore be written as

\[
\dot{x}_i = (x_0 - \hat{x}_i) + \sum_{j \sim i} g_{ij}(\|x_i - x_j\|)(x_i - x_j) \tag{3.4}
\]

\[
\dot{\hat{x}}_i = (x_0 - \hat{x}_i) + \sum_{j \sim i} A_{ij}(\hat{x}_j - \hat{x}_i) + \sum_{j=1}^{N} A_{ij}(x_j - \hat{x}_i) \tag{3.5}
\]

Let \( \overline{x}(t) = \frac{1}{N} \sum_{j=1}^{N} x_j(t) \) denote the swarm center at time \( t \). The swarm error and consensus error of the \( i \)th agent at time \( t \) are defined as follows,

\[
e_i(t) = x_i(t) - \overline{x}(t) \in \mathbb{R}^n,
\]

\[
\hat{e}_i(t) = \hat{x}_i(t) - \overline{x}(t) \in \mathbb{R}^n,
\]

respectively. Furthermore let \( e(t) \in \mathbb{R}^{Nn} \) and \( \hat{e}(t) \in \mathbb{R}^{Nn} \) denote the swarm and
consensus error vectors,

\[ \mathbf{e}(t) = \begin{bmatrix} e_1^T(t) & e_2^T(t) & \cdots & e_N^T(t) \end{bmatrix}^T \]
\[ \dot{\mathbf{e}}(t) = \begin{bmatrix} \dot{e}_1^T(t) & \dot{e}_2^T(t) & \cdots & \dot{e}_N^T(t) \end{bmatrix}^T \]

We say that the swarm defined by equations 3.4 and 3.5 is **cohesive** if there exist positive real constants \( R \) and \( \bar{R} \) such that \( \limsup \| \mathbf{e}(t) \| = \bar{R} \) and \( \liminf \| \mathbf{e}(t) \| = R \). We say the swarm achieves **\( \epsilon \)-consensus** if there exists a positive real constant \( \epsilon \) such that \( \limsup \| \dot{\mathbf{e}}(t) \| = \epsilon \). The objective of this chapter is to establish whether the swarm defined in equations 3.4 and 3.5 is cohesive, achieves \( \epsilon \)-consensus, and to provide bounds on the constants \( \bar{R}, R, \) and \( \epsilon \) as a function of the swarm parameters.

### 3.4 Error Equations

Since our analysis is concerned with the asymptotic behavior of the error vectors \( \mathbf{e}(t) \) and \( \dot{\mathbf{e}}(t) \), it will be convenient to transform the original system equations into a set of coupled error equations.
The derivative of the $i$th agent’s swarm error is

\[
\dot{e}_i = \dot{x}_i - \frac{1}{N} \sum_{j=1}^{N} \dot{x}_j \\
= (x_0 - \dot{x}_i) + \sum_{j \sim i} g_{ij}(\|x_i - x_j\|)(x_i - x_j) \\
- \frac{1}{N} \sum_{j=1}^{N} \left((x_0 - \dot{x}_j) + \sum_{k \sim j} g_{jk}(\|x_j - x_k\|)(x_j - x_k)\right) \\
= -\dot{x}_i + \sum_{j \sim i} g_{ij}(\|x_i - x_j\|)(x_i - x_j) \\
- \frac{1}{N} \sum_{j=1}^{N} \left(-\dot{x}_j + \sum_{k \sim j} g_{jk}(\|x_j - x_k\|)(x_j - x_k)\right)
\]

Note that $\sum_{j=1}^{N} \sum_{k \sim j} g_{jk}(\|x_j - x_k\|)(x_j - x_k) = 0$, $x_i - x_j = e_i - e_j$, and $\dot{x}_i - \dot{x}_j = \dot{e}_i - \dot{e}_j$ so the swarm error equation becomes

\[
\dot{e}_i = \sum_{j \sim i} g_{ij}(\|e_i - e_j\|)(e_i - e_j) + \frac{1}{N} \sum_{j=1}^{N} (\dot{e}_j - \dot{e}_i) \quad (3.6)
\]

for $i = 1, \ldots, N$.

The derivative of the $i$th agent’s consensus error is

\[
\dot{\hat{e}}_i = \dot{\hat{x}}_i - \frac{1}{N} \sum_{j=1}^{N} \dot{\hat{x}}_j \\
= (x_0 - \dot{\hat{x}}_i) + \sum_{j \sim i} A_{ij}(\dot{\hat{x}}_j - \dot{\hat{x}}_i) + \sum_{j=1}^{N} A_{ij}(x_j - \dot{\hat{x}}_i) - \frac{1}{N} \sum_{j=1}^{N} (x_0 - \dot{\hat{x}}_j) \\
= \sum_{j=1}^{N} A_{ij}(\dot{\hat{x}}_j - \dot{\hat{x}}_i) + \sum_{j=1}^{N} A_{ij}(x_j - \dot{\hat{x}}_i) + \frac{1}{N} \sum_{j=1}^{N} (\dot{\hat{x}}_j - \dot{\hat{x}}_i) \\
= \sum_{j=1}^{N} A_{ij}(\dot{\hat{x}}_j - \dot{\hat{x}}_i) + \sum_{j=1}^{N} A_{ij}(x_j - \dot{\hat{x}}_i)
\]
where $\overline{A}_{ij} = A_{ij} + \frac{1}{N}$. Note that $\hat{x}_j - \hat{x}_i = \hat{e}_j - \hat{e}_i$ and $x_j - \hat{x}_i = e_j - \hat{e}_i$ so we can rewrite the consensus error state equation as

$$
\dot{\hat{e}}_i = \sum_{j=1}^{N} \overline{A}_{ij}(\hat{e}_j - \hat{e}_i) + \sum_{j=1}^{N} A_{ij}(e_j - \hat{e}_i) \tag{3.7}
$$

for $i = 1, \ldots, N$.

It will be convenient to express equation (3.7) in matrix-vector form. In particular, let $\Delta_i$ denote the out-degree of the $i$th agent in the swarm’s communication graph, $G$. Note that $\sum_{j=1}^{N} A_{ij} = 1 + \Delta_i$ and that $\sum_{j=1}^{N} \overline{A}_{ij} = \Delta_i + 2$. Further assume that there exist positive integers $\underline{\Delta}$ and $\overline{\Delta}$ such that $\underline{\Delta} \leq \Delta_i \leq \overline{\Delta}$ for $i = 1, \ldots, N$. With these notational conventions we may rewrite equation 3.7 as

$$
\dot{\hat{e}}_i = \sum_{j=1}^{N} \overline{A}_{ij}(\hat{e}_j - \hat{e}_i) + \sum_{j \sim i} A_{ij}(e_j - \hat{e}_i) + \sum_{j \sim i} \hat{e}_j e_j
$$

But note that

$$
e_i = \frac{1}{N} \sum_{j=1}^{N} (x_i - x_j) = \frac{1}{N} \sum_{j=1}^{N} (e_i - e_j) = e_i - \frac{1}{N} \sum_{j=1}^{N} e_j
$$

which implies that $\sum_{j=1}^{N} e_j = 0$. So we can rewrite our expression for $\dot{\hat{e}}_i$ as

$$
\dot{\hat{e}}_i = -\hat{e}_i - \sum_{j \sim i} A_{ij} \hat{e}_i + \sum_{j \sim i} \overline{A}_{ij}(\hat{e}_j - \hat{e}_i) - \sum_{j \sim i} e_j + \sum_{j \sim i} A_{ij} e_j
$$

$$
= \left( -\frac{2N - 1}{N} - 2\Delta_i \right) \hat{e}_i + \sum_{j \sim i} \overline{A}_{ij} \hat{e}_j + \sum_{j \sim i} (A_{ij} - 1)e_j
$$
Then the matrix-vector form of consensus error equation is,

\[
\dot{\mathbf{e}} = (\mathbf{A} \otimes \mathbf{I}_n) \dot{\mathbf{e}} + (\mathbf{B} \otimes \mathbf{I}_n) \mathbf{e}
\]  (3.8)

where \( \mathbf{M} \otimes \mathbf{N} \) is the Kronecker product of matrix \( \mathbf{M} \) and \( \mathbf{N} \), and

\[
\mathbf{A} = \begin{bmatrix}
K_1\mathbf{I} & \overline{A}_{12}\mathbf{I} & \cdots & \overline{A}_{1N}\mathbf{I} \\
\overline{A}_{21}\mathbf{I} & K_2\mathbf{I} & \cdots & \overline{A}_{2N}\mathbf{I} \\
\vdots & \vdots & \ddots & \vdots \\
\overline{A}_{N1}\mathbf{I} & \overline{A}_{N2}\mathbf{I} & \cdots & K_N\mathbf{I}
\end{bmatrix}
\]

\[
\mathbf{B} = \begin{bmatrix}
0 & (A_{12} - 1)\mathbf{I} & \cdots & (A_{1N} - 1)\mathbf{I} \\
(A_{21} - 1)\mathbf{I} & 0 & \cdots & (A_{2N} - 1)\mathbf{I} \\
\vdots & \vdots & \ddots & \vdots \\
(A_{N1} - 1)\mathbf{I} & (A_{N2} - 1)\mathbf{I} & \cdots & 0
\end{bmatrix}
\]

where \( K_i = \left(-\frac{2N-1}{N} - 2\Delta_i\right) \), \( \mathbf{0} \) is an \( n \times n \) matrix of zeros and \( \mathbf{I} \) is an \( n \times n \) identity matrix. The matrices \( \mathbf{A} \) and \( \mathbf{B} \) also could be written as,

\[
\mathbf{A} = \frac{1}{N} 11^T - 2\mathbf{I}_N - \text{Deg}(G) - \mathbf{L}(G)
\]  (3.9)

\[
\mathbf{B} = \text{Adj}(G) + \mathbf{I}_N - 11^T
\]

\[
\mathbf{L} = \text{Deg}(G) - \text{Adj}(G)
\]

where \( \text{Adj}(G) \) and \( \text{Deg}(G) \) are the adjacency and degree matrix of graph \( G \), respectively. The matrix \( \mathbf{L} \) is the Laplacian matrix of \( G \).
3.5 Uniform Ultimate Bound Analysis

The main result of this chapter establishes bounds on the level of consensus ($\epsilon$) and the swarm size ($R$ and $\bar{R}$) as a function of the swarm parameters $\rho_0$, $\alpha_0$, $N$, $\bar{\Delta}$ and $\Delta$. This is accomplished by studying the uniform ultimate boundedness (UUB) of the swarm dynamics and consensus filters.

The following lemma studies the directional derivative of a positive definite function, $V(e)$ of the swarm error, $e$, $V \in \mathbb{R}^{Nn} \rightarrow \mathbb{R}$ with $V(0) = 0$. The lemma provides sufficient conditions on the norm of $\|e\|$ for which we can show the directional derivative, $\dot{V}(e)$, of this function is negative. This lemma provides one part of the UUB analysis of the swarm under consensus.

**Lemma 3.5.1** Consider the system in equation 3.1 and let $V(e) = \frac{1}{2}e^T e$ for any $e \in \mathbb{R}^{Nn}$. If there exists a positive real constant $\beta$ such that

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\| \geq \beta \|e\| \tag{3.10}$$

and if

$$\|e\| \geq \frac{N(N-1)\rho_0}{\beta \alpha_0} \tag{3.11}$$

then $\dot{V}(e) \leq 0$. 
Proof: The directional derivative of $V(e)$ is

$$
\dot{V}(e) = \sum_{i=1}^{N} e_i^T \dot{e}_i \\
= \sum_{i=1}^{N} \left( e_i^T \left( \sum_{j \sim i} g_{ij}(e_i - e_j) + \frac{1}{N} \sum_{j=1}^{N} (\dot{e}_j - \dot{e}_i) \right) \right) \\
= \sum_{i=1}^{N} \left( \sum_{j \sim i} g_{ij}(\|e_i\|^2 - e_i^T e_j) + \frac{1}{N} \sum_{j=1}^{N} e_i^T (\dot{e}_j - \dot{e}_i) \right)
$$

The last term above is zero because $\sum_{i=1}^{N} e_i = 0$ so the above equation reduces to

$$
\dot{V}(e) = \sum_{i=1}^{N} \sum_{j \sim i} g_{ij}(\|e_i\|^2 - e_i^T e_j)
$$

Completing the square within the above summation yields

$$
\dot{V}(e) = \sum_{i=1}^{N} \frac{1}{2} \left( \sum_{j \sim i} g_{ij}(\|e_i\|^2 - \|e_j\|^2 + \|e_i - e_j\|^2) \right)
$$

Summing the first two terms over $i$ equals zero and recall that $e_i - e_j = x_i - x_j$ so the above equation reduces to

$$
\dot{V}(e) = \sum_{i=1}^{N} \frac{1}{2} \sum_{j \sim i} g_{ij} \|x_i - x_j\|^2
$$

Equations 3.2 allow us to reduce the above equation to

$$
\dot{V}(e) = \frac{N(N - 1)}{2} \rho_0 - \frac{\alpha_0}{2} \sum_{i=1}^{N} \sum_{j \sim i} \|x_i - x_j\| 
$$

(3.12)
By the assumption in equation 3.10 there exists $\beta$ such that

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\| \geq \beta \|e\|,$$

so if

$$N(N - 1)\rho_0 - \alpha_0 \beta \|e\| \leq 0.$$  \hspace{1cm} (3.13)

then we can use equation 3.12 to show that $\dot{V}(e) \leq 0$ and thereby obtain equation 3.11.

**Remark 3.5.1** Equation 3.10 of lemma 3.5.1 can be viewed as a lower bound on the average interagent distance.

The following lemma is an instability result that characterizes the set of $\|e\|$ for which $\dot{V}(e)$ is positive.

**Lemma 3.5.2** Consider the system in equation 3.6 and let $V(e) = \frac{1}{2}e^T e$ where $e \in \mathbb{R}^{Nn}$. If there exists $\beta > 0$ such that

$$\beta \|e\| \geq \sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\|$$  \hspace{1cm} (3.14)

and if

$$\|e\| \leq \frac{N(N - 1)\rho_0}{\beta \alpha_0}$$

then $\dot{V}(e) \geq 0$.

**Remark 3.5.2** Equation 3.14 of lemma 3.5.2 is an upper bound on the average interagent distance.
Proof: If there exists $\beta$ satisfying inequality 3.14 and if we require
\[
\frac{N(N-1)}{2} \rho_0 - \frac{\alpha_0 \beta}{2} \|e\| \geq 0
\] (3.15)
then equation 3.12 in the proof of lemma 3.5.1 implies that $\dot{V} \geq 0$.

The following lemma provides bounds on $\|e\|$ and $\|\hat{e}\|$ for which a positive definite function $\dot{V}(\hat{e})$ of the consensus state has a negative definite directional derivative. Since the consensus error system is a linear system, this lemma is a straightforward UUB analysis.

Lemma 3.5.3 Consider the system defined in equation 3.8 and let $V : \mathbb{R}^{Nn} \rightarrow \mathbb{R}$ be the function $V(\hat{e}) = \frac{1}{2} \hat{e}^T \hat{e}$. Let $\Delta$ denote the minimum out-degree of the swarm’s communication graph. If
\[
\|\hat{e}\| \geq \frac{N - 1 - \Delta}{1 + \Delta} \|e\|
\] (3.16)
then $\dot{V}(\hat{e}) \leq 0$.

Proof: Consider the consensus error dynamics in equation 3.8. Let $\lambda(A)$ and $\lambda(B)$ denote eigenvalues of system matrices $A$ and $B$, respectively. The eigenvalues of $A$ and $B$ are real since these matrices are symmetric. An application of Gershgorin’s theorem establishes that the eigenvalues of $A$ lie in the union of the sets
\[
\Omega_i(A) = \left\{ z \in \mathbb{C} : \left| z + \left( \frac{2N - 1}{N} + 2\Delta_i \right) \right| \leq \Delta_i + \frac{N - 1}{N} \right\}
\]
which means the eigenvalues of $A$ are bounded as

$$\frac{-3N - 2}{N} - 3\bar{\Delta} \leq \lambda(A) \leq -1 - \Delta,$$

where $\bar{\Delta}$ is the maximum out-degree of the swarm’s communication graph, and $\Delta$ is the minimum out-degree. A similar application of Gershgorin’s theorem establishes that the eigenvalues of $B$ lie in the union of sets

$$\Omega_i(B) = \left\{ z \in \mathbb{C} : |z| \leq \sum_{j \sim i} |A_{ij} - 1| \right\},$$

which means the eigenvalues of $B$ are bounded as

$$\Delta + 1 - N \leq \lambda(B) \leq N - 1 - \Delta.$$

Now consider the directional derivative

$$\dot{V}(\hat{e}) = \hat{e}^T \dot{\hat{e}} = \hat{e}^T A\hat{e} + \hat{e}^T B e.$$

We may use the aforementioned bounds on $\lambda(A)$ and $\lambda(B)$ to show that

$$\dot{V}(\hat{e}) \leq -(1 + \Delta)\|\hat{e}\|^2 + (N - 1 - \Delta)\|e\|\|\hat{e}\| \quad (3.17)$$

If the righthand side of equation 3.17 is negative definite then $\dot{V}(\hat{e}) \leq 0$. Inequality 3.17 can be rearranged to yield equation 3.16. □
3.6 Cohesion Analysis

Establishing the cohesion of the swarm under consensus is accomplished by examining the regions identified in lemmas 3.5.1 to 3.5.3. This examination allows us to identify a compact region which is an attracting invariant set of the system.

**Proposition 3.6.1** Consider the interconnected system given by equation 3.6 and 3.7. Assume there exist constants $\beta$ and $\bar{\beta}$ such that

$$\beta \| \epsilon \| \leq \sum_{i=1}^{N} \sum_{j=1}^{N} \| x_i - x_j \| \leq \bar{\beta} \| \epsilon \| \quad (3.18)$$

Let

$$\Omega^{-}_s = \left\{ (\epsilon, \hat{\epsilon}) \in \mathbb{R}^{2Nn} : \| \epsilon \| \geq \frac{N(N-1)\rho_0}{\beta\alpha_0} \right\}$$

$$\Omega^{+}_s = \left\{ (\epsilon, \hat{\epsilon}) \in \mathbb{R}^{2Nn} : \| \epsilon \| \leq \frac{N(N-1)\rho_0}{\beta\alpha_0} \right\}$$

$$\Omega^{-}_c = \left\{ (\epsilon, \hat{\epsilon}) \in \mathbb{R}^{2Nn} : \| \hat{\epsilon} \| \geq \frac{N - 1 - \Delta}{1 + \Delta} \| \epsilon \| \right\}$$

For any initial state $(\epsilon(0), \hat{\epsilon}(0)) \in \mathbb{R}^{2Rn}$, the set

$$\Omega = (\Omega^{+}_s)^c \cap (\Omega^{-}_s)^c \cap (\Omega^{-}_c)^c$$

is an attracting invariant set.

**Proof:** The region identified in lemmas 3.5.1, 3.5.2, and 3.5.3 are precisely the sets $\Omega^{-}_s$, $\Omega^{+}_s$ and $\Omega^{-}_c$, respectively. The set $\Omega$ is the intersection of the complements of these sets. From lemmas 3.5.1 and 3.5.2, we know the region $(\Omega^{+}_s)^c \cap (\Omega^{-}_s)^c$ must be an attracting invariant set. From lemma 3.5.3 we know that the region $(\Omega^{-}_c)^c$
is an attracting invariant set. Therefore the intersection of these two sets (the set \( \Omega \)) is also an attracting invariant set and the proposition’s proof is complete.

\[
\hat{e} \in \partial \Omega \quad V(e) \leq 0
\]

\[
\hat{e} \not\in \partial \Omega \quad \dot{V}(e) \leq 0
\]

Figure 3.2. Geometric analysis of interconnected system cohesiveness

**Remark 3.6.1** Figure 3.2 provides a graphic illustration of proposition 3.6.1’s proof. In this illustration, a specific swarm size \( N = 20 \), and the repulsion coefficient \( \rho_0 = 1 \) and the attraction coefficient \( \alpha_0 = 2 \). The communication graph has a minimum connectivity of \( \Delta = 10 \). The boundary of sets \( \Omega^- \), \( \Omega^+ \) and \( \Omega^- \) are shown in figure 3.2 for a system in which \( N = 20 \), \( \Delta = 10 \), \( \rho_0 = 1 \) and \( \alpha_0 = 2 \). The downward arrow shows the direction in which \( V(\hat{e}) \) is a monotone decreasing function of time. The right to left (left to right) arrow shows the direction in which \( V(e) \) is a monotone decreasing (increasing) function of time. The arrows point to the boundary that \( \hat{e} \) or \( e \) is converging. The attracting invariant set \( \Omega \) is the
shaded region in the figure.

The following corollary of proposition 3.6.1 simply states that the swarm is cohesive under consensus.

**Corollary 3.6.1** Consider the interconnected system given by equation 3.6 and 3.7. Assume there exist constants $\beta$ and $\bar{\beta}$ such that

$$\beta \|e\| \leq \sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\| \leq \bar{\beta} \|e\|$$

Then the swarm is cohesive and achieves $\epsilon$-consensus where

$$\begin{align*}
\overline{R} &= \frac{N(N-1)\rho_0}{\beta \alpha_0}, \\
R &= \frac{N(N-1)\rho_0}{\bar{\beta} \alpha_0} \\
\epsilon &= \frac{N-1-\Delta}{1+\Delta} \frac{N(N-1)\rho_0}{\beta \alpha_0}
\end{align*}$$

**Proof:** The variables $\overline{R}$ and $R$ are the bounds on $\|e\|$ obtained in lemmas 3.5.1 and 3.5.2, respectively. The variable $\epsilon$ is obtained by inserting $\overline{R}$ into the upper bound for $\|\hat{e}\|$ in lemma 3.5.3. This corresponds to the upper righthand corner of the set $\Omega$ in figure 3.2.

The following corollaries to the earlier stability analysis discuss the rate at which the swarm error $\|e\|$ and the consensus level $\|\hat{e}\|$ enter the attracting invariant set $\Omega$.

**Corollary 3.6.2** Consider the dynamical system given by equation 3.6 and assume the assumptions in lemma 3.5.2 hold. If $e(t)$ lies in the set $\Omega^-$ for $t \in [0, T]$, then $V(e(t)) \leq W(t)$ where $W(t)$ satisfies

$$A - B \sqrt{2W(t)} - A \log(-A + B \sqrt{2W(t)}) = A^2 t + W_0$$
and $A = \frac{N(N-1)}{2}\rho_0$, $B = \frac{\alpha_0 \beta}{2}$, and $W_0 = \frac{1}{2}e^T(0)e(0)$.

**Proof:** From lemma 3.5.1, we know that the swarm size is monotone decreasing within region $\Omega_s^-$. The Lyapunov function satisfies the following differential inequality in this region,

$$
\dot{V} \leq \frac{N(N-1)}{2}\rho_0 - \frac{\alpha_0}{2}\beta \sqrt{2V} \leq 0 \quad (3.19)
$$

The result in the theorem follows from a straightforward application of the Comparison lemma. ■

**Remark 3.6.2** A similar corollary can be used to bound the swarm error state trajectory when it lies within $\Omega_s^+$.

**Remark 3.6.3** Figure 3.3 provides a graph illustrating the function $W(t)$ in corollary 3.6.2. The illustration has same system parameters to figure 3.2. The boundaries of sets $\Omega_s^+$ and $\Omega_s^-$ are computed using same swarm system parameters in figure 3.2. The dashed lines are the function trajectories of $W(t)$. The solid line is the $V(t)$ trajectory of swarm size on simulation. The results show that the trajectory of the Lyapunov function is clearly bounded by $W(t)$.

**Corollary 3.6.3** Consider the system consensus error defined in equation 3.7 and assume the assumptions in lemma 3.5.3 are true and let $\theta$ be a real number between 1 and 0. If $\hat{e}(t)$ satisfies

$$
\|\hat{e}\| \geq \frac{N - 1 - \Delta}{(1 - \theta)(1 + \Delta)}\|e\|
$$
Figure 3.3. Convergence rate of interconnected system cohesiveness

for $t \in [0, T]$, then

$$V(t) = \frac{1}{2} \mathbf{e}^T(t) \mathbf{e}(t) \leq V_0 \exp (-2\theta(1 + \Delta)t) \quad (3.20)$$

for $t \in [0, T]$.

Proof: Within the specified region, we know that

$$\dot{V}(\mathbf{e}) \leq -2\theta(1 + \Delta)V(\mathbf{e})$$

Another application of the Comparison lemma completes the proof.
Figure 3.4. Convergence rate of consensus error

Remark 3.6.4 Figure 3.4 provides a graph illustrating how quickly $\hat{e}(t)$ converges. The illustration also has the same system parameters to Figure 3.2. For different values of $\theta$, there exist different convergence regions within $\Omega_c$. This figure plots the trajectory of $V(\hat{e})$ (solid line) and the bound (dashed line) in equation 3.20 for $\theta = 0.3$. The results clearly show the actual trajectory of the consensus error is bounded as indicated by the corollary 3.6.3, where in this case $\|\hat{e}\| = 3$.

3.7 Interagent Distance Analysis

As mentioned earlier, equation 3.18 in proposition 3.6.1 is an assumed upper and lower bound on the average inter-agent distance. This bound is expressed as a linear function of the vector 2-norm of the swarm error vector, which we can consider as a reasonable measure of the swarm’s size. This section justifies the bound in equation 3.18 and shows how we can go about computing the constants $\hat{\beta}$ and $\underline{\beta}$. 

74
We first claim that we can always bound the interagent distance as shown in equation 3.18. Let $\|x\|_1 = \sum_{i=1}^{n} |x_i|$ denote the 1-norm of vector $x \in \mathbb{R}^n$. Let $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ denote the 2-norm of vector $x \in \mathbb{R}^n$. It is already known from standard mathematical analysis that we can always find constants $c$ and $C$ such that

$$c\|x\|_2 \leq \|x\|_1 \leq C\|x\|_2$$

So now consider the swarm error vector $e \in \mathbb{R}^{Nn}$ and note that

$$\|e\|_1 = \sum_{i=1}^{N} \|e_i\|_1 \leq \sum_{i=1}^{N} \frac{1}{N} \sum_{j=1}^{N} \|x_i - x_j\|_1$$

$$\leq \sum_{i=1}^{N} C \sum_{j=1}^{N} \|x_i - x_j\|_2$$

(3.21)

which implies that there exists a constant $\beta$ such that

$$\beta\|e\|_2 \leq \sum_{i} \sum_{j} \|x_i - x_j\|_2.$$ 

Also note that

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\|_2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \|e_i - e_j\|_2$$

$$\leq \sum_{i=1}^{N} \sum_{j=1}^{N} (\|e_i\|_2 + \|e_j\|_2) \leq \frac{N}{c} (\sum_{i=1}^{N} \|e_i\|_1 + \sum_{j=1}^{N} \|e_j\|_1)$$

$$= \frac{2N}{c} \|e\|_1$$

(3.22)
which implies there exists a constant $\overline{\beta}$ such that

$$\overline{\beta}\|e\|_2 \geq \sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\|_2.$$ 

So we can conclude that we can always find constants $\beta$ and $\overline{\beta}$ such that inequality 3.18 is true.

The determination of constants $\overline{\beta}$ and $\beta$ may be accomplished by solving an associated optimization problem. In particular, consider the optimization problem

\begin{align*}
\text{minimize:} & \quad J = \sum_{i=1}^{N} \sum_{j=1}^{N} \|e_i - e_j\|_2 \\
\text{with respect to:} & \quad e_i \quad (i = 1, \ldots, N) \\
\text{subject to:} & \quad \sum_{i=1}^{N} \|e_i\|_2^2 = E^2 \\
& \quad \sum_{i=1}^{N} e_i = 0
\end{align*}

where $E$ is a parameter to be chosen. This parameter represents the total squared distance between swarm agents. Essentially this problem is finding the smallest average interagent distance $\sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\|_2$ such that the total squared distance, $\|e\|_2$, is equal to $E$. Recall that we showed earlier $\sum_{i=1}^{N} e_i$ must always equal zero, so the final constraint in the optimization problem ensures that this condition is satisfied.

The solution to the previous optimization problem may be denoted as $J(E)$ where $E$ is the supplied parameter. Since $E$ equals the swarm size, the solutions to this optimization problem is generating the curve $J(E)$ which we can then easily fit with a linear function of $E$, thereby identifying the constant $\overline{\beta}$ which enforces the lefthand side of inequality 3.18. A similar approach may be used to determine $\beta$. In this case, however, we seek to maximize $J$ subject to the same constraints.
The solutions to this set of maximization problems will generate solutions $\bar{J}(E)$ which we can again over bound with a linear function of $E$ to determine $\bar{\beta}$.

This optimization problem was solved for a specific swarm of size $N = 20$ using Matlab’s `fmincon` function. The asterisks in figure 3.5 plot $\bar{J}(\|e\|_2)$ and $\bar{J}(\|e\|_2)$ versus $\|e\|_2$. The dashed lines represent the best fit linear functions of $\|e\|_2$ for the data. For this particular swarm we determined that $\bar{\beta} = 114$ and $\bar{\beta} = 40$.

The distributions of swarm error vectors $e$ computed by solving this optimization problem are shown in figure 3.6 for $\|e\|_2 = E = 5$. The left-hand figure corresponds to the $\bar{J}(E)$ configuration and the right-hand figure corresponds to the $\bar{J}(E)$ configuration. Given total distance between swarm agents, $E$, the $\bar{J}(E)$ and $\bar{J}(E)$ configurations are related to the minimal and maximal interagent distances over the whole swarm, respectively. In configuration $\bar{J}$, the agents have all
grouped together into two distinct clusters. The configuration $\mathbf{J}$ shows a set of agents that are uniformly spaced. With the preceding bounds for $\overline{\beta}$ and $\overline{\beta}$ we can now verify the preceding section’s analysis results through simulations.

A Matlab script was written to simulate the system equations in equations 3.4 and 3.5. This simulation was performed with 20 agents in which the repulsion coefficient $\rho_0$ and the attraction coefficient, $\alpha_0$ were both equal to one. The communication graph was specified at time 0 and that graph was kept static over the length of the run. This simulation’s communication graph had a maximum connectivity of about $\Delta = 10$. The swarm was attempting to intercept a target that started at $(0, 150)$ and moved with a constant velocity of $(-10, -10)$. The swarm was initialized to be uniformly distributed over a rectangular region with

![Diagram showing agent configurations with low-energy (left) and high-energy (right).](image-url)
side length 30 centered at the (15, 15). The simulation was run for 100 time steps with a step size of $T = 0.02$.

Figure 3.7 shows the swarm and consensus errors ($x - \bar{x}$ and $\hat{x} - \bar{x}$) at the final simulation time. The righthand plot shows the final swarm error vectors and the lefthand plot shows the final consensus error vectors. In this case the final swarm size $\|e\| = 3.31$ and the final consensus error $\|\hat{e}\| = 0.48$. Note that the final swarm configuration is similar to the $I(E)$ configuration shown in figure 3.6. This suggests that the associated swarm size should be closer to the lower bound in equation 3.18 than the upper bound.

![Swarm Error and Consensus Error](image)

Figure 3.7. Final swarm/consensus error vectors

Figures 3.8 plot the the swarm and consensus errors convergence processes. In
this particular simulation, the swarm size \( N = 20, \rho_0 = 1, \alpha_0 = 1 \) and \( \Delta = 10 \). The figures show that the swarm error \( e \) and consensus error \( \hat{e} \) converge to the invariant set \( \Omega \) exponentially.

Figure 3.8. Swarm/consensus time history

Figure 3.9 is similar to the plot shown in figure 3.2. In this figure, however, not only do we plot \( \Omega \), but we show the final swarm and consensus errors achieved by the simulation. This simulation was run for 5000 steps with a step size \( T = 0.02 \). This final error vector is shown by the big dot. The region \( \Omega \) is marked by the shaded region. The four plots in figure 3.9 show these regions and simulation data assuming \( \alpha_0 = 2 \) and with \( \rho_0 \) ranging from 0.5 to 2.0. The plots demonstrate
that the simulated swarm size $\|e\|$ and consensus error $\|\hat{e}\|$ lies within the set $\Omega$. Thereby, the simulations confirm the theoretical analysis on swarm cohesion under consensus.

### 3.8 Perfect Consensus

As stated in the introduction, the consensus filter generates estimates of the swarm center which are then used by agents to guide the swarm to the target. The analytical bounds and simulation results presented above indicate that using the consensus filter in equation 3.3, the best we can hope for is $\epsilon$-consensus where the size of $\epsilon$ is given in corollary 3.6.1. Obviously what we’d like to do is identify conditions under which we might drive $\epsilon$ to zero and thereby achieve perfect
consensus.

One obvious way of achieving perfect consensus is through the introduction of *integral action* in the consensus filter equation. The state equations for the consensus filter with integral action are shown below,

\[
\dot{x}(t) = (x_0(t) - \hat{x}_i(t)) + \sum_{j \sim i} A_{ij}(x_j(t) - \hat{x}_i(t)) + \sum_{j \sim i} A_{ij}(\dot{x}_j(t) - \dot{x}_i(t)) + Kz_i
\]

\[
\dot{z}_i = \sum_{j=1}^{N} A_{ij}(\dot{x}_j(t) - \hat{x}_j(t))
\]  

(3.23)

for \(i = 1, \ldots, N\). The entire system is formed by combining the swarm dynamics in equation 3.4 with the modified consensus filter above in equation 3.23.

Let \(z \in \mathbb{R}^{Nn}\) be the vector of integrated errors. We can use equations 3.4 and 3.23 to obtain the following state equation for the consensus error \(\hat{e}\),

\[
\dot{\hat{e}} = (A \otimes I_n) \hat{e} + (B \otimes I_n) e + K I_{Nn} z
\]

\[
\dot{z} = -L \otimes I_n \hat{e}
\]  

(3.24)  

(3.25)

where \(A \otimes B\) is the Kronecker product of matrix \(A\) and \(B\). We can rewrite these equations in matrix vector form as

\[
\begin{bmatrix}
\dot{\hat{e}} \\
\dot{z}
\end{bmatrix} = \begin{bmatrix}
A & K I_n \\
-L & 0
\end{bmatrix} \otimes I_n \begin{bmatrix}
\hat{e} \\
z
\end{bmatrix} + \begin{bmatrix}
B \\
0
\end{bmatrix} \otimes I_n e
\]  

(3.26)

where all those matrices \(A\), \(B\) and \(L\) have the same meaning of equation 3.9.
3.8.1 Consensus Error Analysis

This section contains the chapter’s main result, which is a theorem establishing conditions under which the consensus filter achieves perfect consensus. The proof of this result requires the following two technical lemmas. The first lemma characterizes the eigenvalues of the system matrix

\[
\Phi = \begin{bmatrix}
A & KT \\
-L & 0
\end{bmatrix}.
\]

The second lemma characterizes the similarity transformation taking \( \Phi \) to its diagonal canonical form. The proofs for both lemmas will be found in the appendix (chapter 7).

**Lemma 3.8.1** Assume the communication graph, \( G \), is connected then \( \Phi \) has exactly one zero eigenvalue and all other eigenvalues have real parts strictly less than zero.

**Lemma 3.8.2** Let \( \Lambda \) be a diagonal complex-valued matrix whose diagonal elements are the eigenvalues of \( \Phi \otimes I_n \). Let \( U \) denote the similarity transformation such that \( \Phi \otimes I_n = U \Lambda U^{-1} \).

\[
U = \begin{bmatrix}
\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{2N}
\end{bmatrix} \otimes I_n,
\]

and

\[
U^{-1} = \begin{bmatrix}
\mathbf{v}_1^T & \mathbf{v}_1^T \\
\mathbf{v}_2^T & \mathbf{v}_2^T \\
\vdots & \vdots \\
\mathbf{v}_{2N}^T & \mathbf{v}_{2N}^T
\end{bmatrix} \otimes I_n
\]
where \( u_i, \ u_i, \ v_i \) and \( v_i \) \( \in \mathbb{R}^N \) for \( i = 1, \ldots, 2N \). The matrices \( U \) and \( U^{-1} \) have following properties,

1. \( u_{2N} = u \cdot 1^T \in \mathbb{R}^N \), where \( u \) is a constant.

2. \( v_{2N} = 0 \cdot 1^T \in \mathbb{R}^N \), and \( v_{2N} = v \cdot 1^T \in \mathbb{R}^N \), where \( v \) is a constant.

3. \( K v_i^T = \lambda_i v_i^T \quad i = 1, \ldots, 2N - 1 \)

4. \( \sum_{i=1}^{2N-1} u_i v_i^T = -u_{2N} v_{2N}^T \)

5. \( v_{2N}^T u_{2N} = 1 \)

6. \( A u_{2N} + K u_{2N} = 0 \)

From lemma 3.5.2, we know eventually \( \|e(t)\| \leq \overline{R} \). Assume that \( e(t) \) satisfies this inequality at time 0. Assuming that initial state satisfies \( \hat{e}(0) = z(0) = 0 \), we can use the consensus error equations 3.24-3.25 to see

\[
\begin{bmatrix}
\hat{e}(t) \\
z(t)
\end{bmatrix} = \int_0^t U e^{A(t-\tau)} U^{-1} \begin{bmatrix}
B \otimes I_n \\
0
\end{bmatrix} e(\tau) d\tau
\]

We define the vector \( \|\hat{e}_{ss}\| \)

\[
\|\hat{e}_{ss}\| = \lim_{\tau \to \infty} \sup_{t} \{\|\hat{e}(t)\| : t \geq \tau\}
\]

The following theorem provides an upper bound on \( \|\hat{e}_{ss}\| \).

**Theorem 3.8.1**

\[
\|\hat{e}_{ss}\| \leq \left\| \left( \frac{1}{c} 11^T \right) B \right\| \overline{R}
\]
where the scalar \( c \) is \( c = -1^T A 1 \).

**Proof:** For notational convenience we let \( \overline{A} = A \otimes I_n \) and \( \overline{B} = B \otimes I_n \). Then

\[
\| \hat{e}(t) \| = \left\| \int_0^t [I_{Nn} 0_{Nn}] U e^{A(t-\tau)} U^{-1} \begin{bmatrix} \overline{B} \\ 0 \end{bmatrix} e(\tau)d\tau \right\|
\]

where

\[
u = \begin{bmatrix} u_1 & u_2 & \cdots & u_{2N} \end{bmatrix} \otimes I_n
\]

\[
v = \begin{bmatrix} v_1 & v_2 & \cdots & v_{2N} \end{bmatrix} \otimes I_n
\]

Since \( v_{2N} = 0 \), the lowest \( n \times n \) sub-matrix in the matrix \( v^T \overline{B} \) is zero. So we can conclude

\[
\| \hat{e}_{ss} \| \leq \left\| u \begin{bmatrix} -\frac{1}{\lambda_1} & & \cdots & \frac{1}{\lambda_{2N-1}} \end{bmatrix} \otimes I_n v^T B \right\| \overline{R}
\]

\[
= \left\| \sum_{i=1}^{2N-1} -\frac{u_i v_i^T}{\lambda_i} + r u_{2N} v_{2N}^T \right\| \overline{B} \overline{R}
\]

\[
= \sum_{i=1}^{2N-1} -\frac{u_i v_i^T}{\lambda_i} \overline{B} \overline{R}
\]

\[
= \left( \frac{1}{K} u_{2N} v_{2N}^T \right) \overline{B} \overline{R}
\]

\[
= \left( \frac{1}{K} uv 11^T \right) \overline{B} \overline{R}
\]
where \( r \) could be any value, and the last couple equalities follow from lemma 3.2.

Multiplying \( \mathbf{v}_{2N}^T \) on the left-hand side and using the sixth item in lemma 3.8.2 yields,

\[
\mathbf{v}_{2N}^T \mathbf{A} \mathbf{u}_{2N} + K = 0
\]

The above equation is equivalent to,

\[
\mathbf{u} \mathbf{v} \cdot \mathbf{1}^T \mathbf{A} \mathbf{1} + K = 0
\]

Therefore,

\[
\| \hat{e}_{ss} \| \leq \left\| \left( \frac{1}{\mathbf{1}^T \mathbf{A} \mathbf{1}} \mathbf{1}^T \right) \mathbf{B} \right\| \sqrt{\mathbf{R}}
\]

and the proof is complete. \( \blacksquare \)

The following theorem represents the main result of this chapter. It states and proves a bound on \( \| \hat{e}_{ss} \| \) for swarming under consensus with integral action.

**Theorem 3.8.2** Let \( \overline{\Delta} \) and \( \underline{\Delta} \) denote the maximum and minimum out-degree of the communication graph, respectively. Then

\[
\| \hat{e}_{ss} \| \leq \frac{\overline{\Delta} - \underline{\Delta}}{N(1 + \overline{\Delta})} \sqrt{\mathbf{R}}
\]

(3.29)

**Remark 3.8.1** A regular graph is one in which \( \underline{\Delta} = \overline{\Delta} \), so that all nodes have the same out degree. Theorem 3.8.2 means that if the graph, \( \mathbf{G} \), is regular then the consensus error will be zero in swarms under consensus with integral action regardless of the swarm’s size.
Proof: Let

\[ M = \left( \frac{1}{c} 1 1^T \right) B, \]

where the scalar \( c \) is \(-1^T A 1\). The norm of \( M \) can be bounded as

\[
\|M\|^2 \leq \frac{1}{c^2} x^T B^T 1 1^T \cdot 1 1^T B x
\]
\[
= \frac{N}{c^2} x^T B^T 1 1^T B x
\]
\[
= \frac{N}{c^2} (1^T B x)^2
\]

where \( x \in \mathbb{R}^N \) is a nonzero vector such that \( \|x\| = 1 \) and \( \sum_{i=1}^{N} x_i = 0 \).

From the definition of matrix \( B \) we can show that

\[ 1^T B x = - \sum_{i=1}^{N} (N - 1 - \Delta_i) x_i \]

where \( x_i \) is the \( i^{th} \) element of vector \( x \).

By construction \( \sum_{i=1}^{N} x_i = 0 \) so we can partition \( x \) so that \( x_i \leq 0 \) \((i = 1, \cdots , \ell)\) and \( x_i \geq 0 \) \((i = \ell + 1, \cdots , N)\). Then,

\[
|1^T B x| \leq \Delta \sum_{i=1}^{\ell} x_i - \Delta \sum_{i=\ell+1}^{N} x_i = (\Delta - \Delta) \sum_{i=1}^{\ell} x_i
\]

Application of Cauchy’s Inequality \( N \left( \sum_{i=1}^{\ell} x_i \right)^2 \leq \sum_{i=1}^{\ell} x_i^2 \) yields,

\[
\|M\|^2 \leq \frac{N}{c^2} \left( (\Delta - \Delta) \sum_{i=1}^{\ell} x_i \right)^2 \leq \frac{1}{c^2} (\Delta - \Delta) \sum_{i=1}^{\ell} x_i^2 \leq \frac{1}{c^2} (\Delta - \Delta) \sum_{i=1}^{N} x_i^2
\]
\[
= \frac{1}{c^2} (\Delta - \Delta) \|x\| = \frac{1}{c^2} (\Delta - \Delta)
\]

87
From the definition of $A$ we obtain
\[ c = N + \sum_{i=1}^{N} \Delta_i \]
which we can combine in the above inequality to obtain the theorem’s result. ■

The following corollary characterizes the degree of consensus achieved with and without integral action when swarming under consensus.

**Corollary 3.8.1** The ratio of the minimum consensus errors $\hat{e}_{\text{int}}$ and $\hat{e}_{\text{no-int}}$ achieved with and without integral action, respectively is
\[
\frac{\|\hat{e}\|_{\text{int}}}{\|\hat{e}\|_{\text{no-int}}} \leq \frac{\bar{\Delta} - \bar{\Delta}}{N(N - 1 - \bar{\Delta})}
\] (3.30)

**Remark 3.8.2** This corollary bounds the decrease in the consensus error when we add integral action. The result shows that consensus error can be small in poorly connected graphs ($\bar{\Delta}$ is small) provided the swarm is large enough.

**Proof:** This follows directly from the lemma 3.5.3 and equation 3.29 in theorem 3.8.2. ■

3.8.2 Perfect Consensus Simulation

A matlab script was written to simulate swarming under consensus with integral action. In the following simulations, the integrator gain is $K = 20$ and the swarm size is $N = 20$. The repulsion/attraction strengths are $\rho_0 = 1$ and $\alpha_0 = 2$, respectively. Every simulation ran for 6000 time steps with a step size of $T = 0.02$.

We first simulated swarming under consensus with integral action on the two communication graphs shown in figure 3.10. The left-hand figure corresponds to
a regular graph with degree 8. The right-hand figure corresponds to a connected graph with $\Delta = 19$ and $\Delta = 8$. The degree distribution for this graph is shown in figure 3.11.

![8-degree Communication Graph](image1.png)

![Connected Communication Graph](image2.png)

Figure 3.10. Communication graph, $N = 20$ (left) 8-degree (right) connected graph

Even though the connected graph has an agent that is connected to all other agents, the entire swarm is unable to achieve perfect consensus. The regular graph, on the other hand, achieved perfect consensus as is shown in figure 3.12. This figure plots the log of the norm squared consensus error, $\|\hat{e}\|^2$ as a function of time. In this particular simulation the swarm size, $\|e\|$ was bounded above by 1.6662. The solid line in figure 3.12 is the consensus error for the regular graph and the dashed line is the consensus error for the other graph. In the regular graph, the consensus error reached a minimum level of $\|\hat{e}_{ss}\| = 5.9174e - 014$, which is essentially zero. The minimum consensus error achieved over the other
graph was several orders of magnitude larger.

As noted above, even if the graph is not regular, integral action can dramatically improve the level of consensus. Figure 3.13 shows the comparison of minimum consensus error with /without integral action on the same communication graph. In this particular graph the node out-degrees were bounded between $\Delta = 3$ and $\Delta = 7$. Without integral action the minimum consensus error was about 0.9851 (dashed line). With an integral gain of $K = 20$, the same system achieved a minimum consensus level of 0.0102. The figure verifies that integral action can decrease the consensus error significantly.

The following simulation results experimentally evaluate the tightness of the bounds presented in theorem 3.8.2 and corollary 3.8.1. Theorem 3.8.2’s proof used the following bound

$$|1^T B x| \leq (\Delta - \underline{\Delta}) \sum_{i=1}^\ell x_i \quad \text{and,}$$

$$(\Delta - \underline{\Delta}) \sum_{i=1}^\ell x_i^2 \leq (\Delta - \underline{\Delta}) \sum_{i=1}^N x_i^2$$

in which a unit vector $x$ satisfying $\sum_{i=1}^N x_i = 0$ was partitioned into its positive
and negative components \((x_i \leq 0 \ (i = 1, \cdots \ell) \text{ and } x_i \geq 0 \ (i = \ell + 1, \cdots, N))\).

The bound clearly gets tight when \(\Delta\) is close to \(\overline{\Delta}\).

Figure 3.14 illustrates the relationship between \((\overline{\Delta} - \underline{\Delta})\) and the bound on \(\|\hat{e}_{ss}\|\). This figure plots \(\|\hat{e}_{ss}\|\) for two different graphs. In the first graph (solid line) there is a large spread in the node out-degrees \((\overline{\Delta} = 13 \text{ and } \underline{\Delta} = 5)\). In this case the consensus error is predicted to be less than 0.3176 by theorem 3.8.2. The actual minimum consensus error, however, was only 0.0154. In the second graph (dashed line), there is a small spread in the node out-degrees \((\Delta = 10 \text{ and } \overline{\Delta} = 11)\). For this case, theorem 3.8.2 predicts a consensus error that is less than 0.0216 with the actual norm being 0.0134. These results show close agreement between the predictions made in theorem 3.8.2 and actual simulated results.
Figure 3.13. Consensus error equilibrium with /without integral action

Figure 3.14. Consensus error bound with different max and min communication degree
CHAPTER 4

CONVERGENCE OF CONSENSUS FILTERING UNDER NETWORK THROUGHPUT LIMITATIONS

4.1 Overview

Real-life communication networks have limited resources such as the number of channels and channel bandwidth. These resource constraints lead to a bounded network throughput, or delay message delivery in ways that can adversely affect the consensus filter stability. Consensus with transmission delay was first explored in [64]. This work, however, only established an upper bound on the delay for the system's asymptotic stability. Following researchers focused on providing sufficient conditions of asymptotic consensus with bounded delay, by using different communication topology models. This chapter presented a more fundamental understanding on the relationships between network topology, delay, and convergence of consensus. Namely, assume a time-slotted frequency division multiple access (FDMA) wireless network is used and the transmission collisions, hence, occur due to insufficient FDMA channels. This chapter presented a bound on the convergence rate of consensus filters with delay. We show that there is an optimal level of communication connectivity in terms of the filter’s convergence rate and communication energy efficiency.

The related works that optimize network throughput by identifying network topologies are more likely to be found in communication literatures. For example,
the optimal connectivity was explored in [45] to maximize the throughput subject to the packet collision in wireless networks. More recently a series of papers [36, 39, 58, 105] have studied connectivity in packet radio networks for different optimization objectives.

The organization of this chapter is as follows. Section 4.3 discusses the delay models in multi-hop wireless sensor networks. Section 4.4 characterizes the convergence rate of the consensus filter with such message delays. Section 4.5 analyzes the minimum energy required to achieve $\epsilon$-consensus. Simulation results in support of the chapter’s analysis will be presented in section 4.6.

4.2 Introduction

In many MAS applications, it is important for individual agents to have a global aggregate of the network’s local measurements. Consensus filtering [66] provides one way of computing such aggregates in a distributed manner. That is, consensus filters exchange their local measurements among the nearest neighbors, hence the local information is implicitly relayed across the network. Eventually consensus can be achieved when all agents within the network agree upon the same aggregated variable. In most previous studies is shown that the rate at which such filters achieve consensus is proportional to the number of neighbors each agent can communicate with. This suggests that as we increase the connectivity within the network’s communication graph to increase the rate at which the algorithm achieves consensus. This conclusion, however, is simplistic because it ignores the intrinsic throughput limitation of multi-hop communication networks.

Network throughput limitations [38] have a major impact on the consensus filter’s convergence rate. A direct consequence of limited network throughput is
longer communication delay or latency. Namely, due to message collisions, it is impossible for a receiver to collect all its neighbors’ information instantaneously. There is always a finite probability that some of the neighbors’ data will be corrupted and have to be resent. Resending data will delay message delivery such that decreases the filter’s convergence rate. We define *synchronous* consensus scheme such that individual agents are allowed to update their states only after all neighbors information are received; while in the *asynchronous* consensus scheme, each agent updates its state as it receives any messages from neighbors which may be delayed. Regardless of synchronicity of the scheme, there is a fundamental trade off between network connectivity and delay that leads to an “optimal” level of connectivity for achieving the fast consensus. Thus, an immediate observation is that, certain graph would be appropriate to achieve the group consensus behavior, rather than the fully-connected one, though it is well known that consensus rate increases with network connectivity under no-delay assumption. The purpose of this chapter, therefore, is to examine that optimal connectivity in greater detail.

Considerable works, e.g., [6, 30, 42, 57, 64, 74, 80, 88, 89, 100, 102], have been published on studying sufficient conditions for asymptotic consensus convergence, using different mathematical tools, e.g., Lyapunov methodology [64], convex optimization [57] and matrix analysis [74, 80, 100]. The convergence rate is characterized by the second-smallest eigenvalue of the Laplacian matrix associated with the communication graph [64]. This rate can also be adjusted by manipulating weights on the edges of the communication graph [30][102]. Consensus with delay was first addressed in [64], in which an upper bound on the delay to sustain the system stability for synchronous case is established. Asynchronous consensus framework is generated to represent more generally unreliable
communication scenarios in [24, 41, 54, 73, 74, 80]. Correspondingly, switching communication graphs are modeled as a set of finite SIA matrices. Only sufficient conditions are studied for an asymptotic agreement cooperation. This chapter provides more fundamental understanding on the impact of network topology and delay on convergence rate.

The main contribution of this chapter is to design a “good” connectivity for the wireless network in maximizing the consensus convergence rate. The chapter presents an upper bound on the convergence rate of consensus for both the synchronous and asynchronous consensus scheme, subject to communication throughput limitation. We compared the advantages of two consensus schemes based on the analysis. In addition, the optimal connectivity is investigated to minimize the energy required to achieve $\epsilon$-consensus for synchronous consensus, and discussed the consensus performance under the different wireless communication conditions.

4.3 Problem Statement

The consensus problem studied in this chapter has its origin in the distributed filter framework introduced by [66]. Consider a sensor network of size $N$. The consensus state of this network is a function $x_i(t): \mathbb{R} \rightarrow \mathbb{R}^n (i = 1, \ldots, N)$. that satisfies the consensus filter equation as follows,

$$\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} \alpha_{ij} (x_j(t) - x_i(t)) + \sum_{j \in (\mathcal{N}_i \cup \{i\})} \alpha_{ij} (u_j(t) - x_i(t))$$
where $u_i : \mathbb{R} \rightarrow \mathbb{R}^n$ is the filter’s $i$th input. This equation can be rewritten in matrix-vector form as

$$
\dot{x} = -(I_N + \Delta + L)x + (I_N + \text{Adj}(G))u
$$

$$
= Ax + (I_N + \text{Adj}(G))u	ag{4.1}
$$

where $x$ is the vector of consensus states, and $u$ is the vector of filter inputs. We define $A = -(I_N + \Delta + L)$, $I_N$ is an $N \times N$ identity matrix, $\text{Adj}(G)$ is the adjacency matrix of the undirected graph $G$, $L$ is the graph $G$’s Laplacian matrix and $\Delta$ is a diagonal matrix whose diagonal elements are the outdegrees of the graph’s nodes. The graph $G$ characterizes the communication connectivity between the nodes in such a way that node $j$ is able to communicate with node $i$ if and only if the $ij$th component of $\text{Adj}(G)$ is 1 (i.e., $a_{ij} = 1$), otherwise it is zero.

In this chapter, we consider the delay of consensus state broadcast arising in a time-slotted wireless frequency division multiple access (FDMA) network. Let $\tau_{ij}$ denote the delay of information transmitted from agent $j$ to agent $i$, caused by collision, and $\tau_i = \max \tau_{ij}$, for $j \in \mathcal{N}_i$. The consensus dynamics can be expressed for synchronous consensus scheme as,

$$
\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} \alpha_{ij} (x_j(t - \tau_i) - x_i(t - \tau_i)) + \sum_{j \in (\mathcal{N}_i \cup \{i\})} \alpha_{ij} (u_j(t - \tau_i) - x_i(t - \tau_i))\tag{4.2}
$$

or for asynchronous consensus scheme,

$$
\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} \alpha_{ij} (x_j(t - \tau_{ij}) - x_i(t)) + \sum_{j \in (\mathcal{N}_i \cup \{i\})} \alpha_{ij} (u_j(t - \tau_{ij}) - x_i(t))\tag{4.3}
$$

Note that in asynchronous consensus scheme, $\tau_{ii} \equiv 0$. That is, agent $i$ uses the
error between its current state $x_i(t)$ and the delayed agent $j$’s state $x_j(t - \tau_{ij})$ for state update. On the other hand, in synchronous consensus, agent $i$ updates its state solely after gathering all $\Delta$ neighbors’ message, where $\tau_i = \max \tau_{ij}$ for $j \in \mathcal{N}_i$. We emphasize here, regarding the two schemes, the distributions of $\tau_{ij}$ are totally different. We will discuss the difference at length at the following subsections.

Assuming a constant input, and taking the Laplace transform on both consensus dynamics, equation (4.2) and equation (4.3) yields a general expression,

$$X(s) = (sI - A(s))^{-1} \left( x(0) + \frac{r}{s} \right)$$  \hspace{1cm} (4.4)

However, the $ij^{th}$ element of the Laplace matrix $A(s)$, $a_{ij}(s)$, has the corresponding format with respect to two consensus schemes correspondingly as follows,

$$a_{ij}(s) = \begin{cases} 
  a_{ii} & i = j \\
  a_{ij}e^{-\tau_{ij}s} & i \neq j 
\end{cases}$$

synchronous consensus

asynchronous consensus

where $a_{ij}$ is the $ij^{th}$ element of the matrix $A$ in equation (4.1), and $r$ is the constant input vector. Denote $X(s)$ as the Laplace transform of $x(t)$, and $x(0)$ is the initial state. This chapter confines its attention to the consensus dynamics given by,

$$X(s) = (sI - E(A(s)))^{-1} \left( x(0) + \frac{r}{s} \right)$$  \hspace{1cm} (4.5)

where $E(\cdot)$ is the expectation of a random variable.

The reminder of the chapter is devoted to studying the filter convergence rates,
in terms of synchronous and asynchronous consensus schemes, respectively.

4.3.1 Synchronous Consensus

We consider a wireless communication network in which agents transmit in time slots. When an agent tries to broadcast its local state, it randomly picks one frequency out of a set of $Q$ frequencies with equal probability. Namely, it is a time-slotted network using frequency division multiple access (FDMA) to the wireless medium. Suppose that agent $i$ transmits over the $m^{th}$ sub-channel to agent $j$. The delay $\tau_{ij} > 0$ is the time slots needed for accomplishing a successful transmission. Hence, the delay in synchronous consensus for agent $i$ is the maximum of $\tau_{ij}$ for $j \in N_i$.

Regular communication graphs have been shown to be an efficient network topology for distributed consensus [66]. They have also been shown to arise naturally in the swarming under consensus framework [49]. We therefore assume that each agent has the same number, $\Delta$, of neighbors, and is regularly spaced. Due to the nature of wireless transmission, the state information of each agent is broadcast to all $\Delta$ neighbors. The other consequence of wireless transmission is that the wireless medium is shared among agents in the network so that messages may collide at the designated receiver. We assume that the messages will be discarded as collisions occur. In addition, agents cannot receive and transmit at simultaneously. To decrease the likelihood of collisions, each agent broadcasts with a probability $p$ in every time slot. Specifically, the probability that an agent successfully gathers all of its $\Delta$ neighbors’ messages in one particular slot is

$$p_0 = p \left( 1 - \frac{p}{Q} \right)^\Delta$$  \hspace{1cm} (4.6)
and the delay $\tau_i$ satisfies a geometric distribution, i.e.,

$$\text{Prob}\{\tau_i = k\gamma\} = p_0(1 - p_0)^k$$

where $k \in \mathbb{Z}^+$ and $\gamma$ is the time interval of each slot, in unit of seconds per slot. After receiving the $\Delta$ messages from all nearest neighbors, the agent updates its consensus state.

Given $\tau_i$ is independent and identical distribution, for $1 \leq i \leq N$, the matrix $E(A(s))$ in equation (4.5) is,

$$E(A(s)) = E[A\text{diag}(e^{-\tau_i s})] = AE[\text{diag}(e^{-\tau_i s})] = A\sum_{k=0}^{\infty} e^{-k\gamma s} p(\tau_i)$$

In practical, the time interval $\gamma$ is small enough so that

$$\sum_{k=0}^{\infty} e^{-k\gamma s} p(\tau_i) \simeq e^{-\gamma s}E[k] \text{ where } E[k] = \frac{1}{p\left(1 - \frac{p}{\Delta}\right)}$$

Let \( \bar{\tau} = \frac{\gamma}{p\left(1 - \frac{p}{\Delta}\right)} \) denote the average delay with which messages are transmitted through the network, then the consensus dynamics could be approximated by,

$$\dot{x}(t) = Ax(t - \bar{\tau}) + r \quad (4.7)$$

and the dynamics in Laplace transform is simplified to,

$$X(s) = \left(sI - A e^{-\bar{\tau}s}\right)^{-1} \left(x(0) + \frac{r}{s}\right) \quad (4.8)$$

The average time delay $\bar{\tau}$ is a function of the graph’s connectivity, $\Delta$. Obviously, if an agent knows how many neighbors it has, then it will select $p$ to minimize the
delay $\tau$. It is straightforward that the broadcast probability that minimizes the average delay is $p^* = \frac{Q}{1+\Delta}$. The corresponding minimal delay is

$$\tau^* = \frac{\gamma}{Q} \frac{(1 + \Delta)^{1+\Delta}}{\Delta^\Delta}$$

which is monotonically increasing with to the communication degree. This implies that as the network connectivity increases, there will be a consequent increase in message latency. The following section 4.4.1 shows how the delay in message may affect the time and energy required to achieve $\epsilon$-consensus.

4.3.2 Asynchronous Consensus

We consider the same time-slotted frequency division multiple access (FDMA) communication model in section 4.3.1. As indicated in the preceding sections, the collision of messages in the wireless network is inevitable and results in transmission delay. Individual agents, therefore, are only able to update their states based on the outdated messages from neighbors. Fortunately, asynchronous consensus scheme allows individual agents to regulate their local states as soon as they receive any neighbor’s message. It is unlike in synchronous fashion that each agent must collect all $\Delta$ messages for once adjustment. Under such assumptions, the probability of delay for asynchronous consensus scheme is,

$$q = \text{Prob\{agent } i \text{ receives agent } j \text{ message successfully in one slot\}}$$

$$= p \left(1 - \frac{p}{Q}\right)^{\Delta-1}$$

(4.10)
and the delay $\tau_{ij}$ is again geometrically distributed, such that

$$\text{Prob}\{\tau_{ij} = k\gamma\} = q(1 - q)^k \quad (4.11)$$

Regarding the asynchronous consensus, the $ij^{th}$ element of $E(A(s))$ is,

$$E(A(s))_{ij} = \begin{cases} a_{ii} & \text{if } i = j \\ a_{ij}E(e^{-\tau_{ij}s}) = a_{ij} \sum_{k=0}^{\infty} e^{-\gamma \tau s} p(\tau_{ij}) = \frac{a_{ij}q}{1 - e^{-\gamma \tau s}(1 - q)} & \text{if } i \neq j \end{cases}$$

From the definition of matrix $A$ in equation (4.1), the matrix $E(A(s))$ could be represented by,

$$E(A(s)) = A\frac{q}{1 - e^{-\gamma s}(1 - q)} - (1 + 2\Delta)(1 - e^{-\gamma s})(1 - q)\frac{1}{1 - e^{-\gamma s}(1 - q)} I_N \quad (4.12)$$

So that the asynchronous consensus dynamics could be rewritten as,

$$X(s) = \left(sI - A\frac{q}{1 - e^{-\gamma s}(1 - q)} + (1 + 2\Delta)(1 - q)\frac{1 - q}{1 - e^{-\gamma s}(1 - q)} I\right)^{-1}(X(0) + \frac{\text{F}}{s}) \quad (4.13)$$

Instead of providing the sufficient and necessary condition to achieve the asymptotic consensus, we analyzed the convergence rate of the asynchronous consensus in section 4.4.2.

4.4 Convergence Rate

This section considers the convergence rate of the consensus filter with delay. We analyze the convergence rate for both synchronous and asynchronous consensus.
4.4.1 Convergence Rate of Synchronous Consensus

Section 4.3.1 modeled throughput limitations in a time-slotted FDMA wireless network as an average message delay. Olfati-saber et al. [64] derived an upper bound on the maximum delay with which the consensus filter is asymptotically stable. That is, the delay for the stable consensus filter should be in the range as

\[ \tau \in \left(0, \frac{\pi}{2 \lambda_{max}(A)}\right) \] (4.14)

For a regular network, we can use Gershgorin’s theorem to show that the eigenvalues of \( A = -(I_N + \Delta + L) \) are bounded

\[-3\Delta - 1 \leq \lambda(A) \leq -\Delta - 1\] (4.15)

Applying equation (4.15) in equation (4.14) yields the following lemma, and the proof is given in appendix.

**Lemma 4.4.1** If \( \tau \in \left(0, \frac{\pi}{2(1+\Delta)}\right) \), then consensus filter in equation (4.8) is asymptotically stable.

Furthermore, for the optimal delay \( \tau^* \), the degree of the network is bounded the bounds on the network connectivity \( \Delta \), required for the consensus filter’s stability is given in the following lemma.

**Lemma 4.4.2** If \( Q - 1 \leq \Delta \leq \sqrt{\frac{\pi Q}{2e\gamma}} - 1 \), then the consensus filter in equation (4.8) is asymptotically stable.

The convergence rate of the consensus filter can be analyzed directly on the Laplace transform of equation (4.8). The eigenvalue decomposition of the symmetric matrix \( A \) yields \( A = U\Lambda U^T \), where \( \Lambda \) is the diagonal matrix with the
eigenvalues of $A$ along the diagonal, and $U$ is an orthogonal matrix. Therefore,

$$X(s) \approx U \text{diag} \left( \frac{1}{s - \lambda_i e^{-\tau s}} \right) U^T \left( x(0) + \frac{r}{s} \right)$$  \hspace{1cm} (4.16)

Applying the $P_{2,1}$ Padé approximation on the delay term $e^{-\tau s}$ gives

$$\frac{1}{s - \lambda_i e^{-\tau s}} = \frac{1}{s - \lambda_i \frac{6 - 6\tau \lambda_i + (\tau s)^2}{6 + 2\tau s}} = \frac{6 + 2\tau s}{s^2 + \frac{6 + 4\lambda_i \tau}{2\tau - \tau^2 \lambda_i} s - \frac{6\lambda_i}{2\tau - \tau^2 \lambda_i}}$$  \hspace{1cm} (4.17)

Let $L_i(s)$ denote the characteristic polynomial for the $i^{th}$ subsystem, which is given by

$$L_i(s) = s^2 + \frac{6 + 4\lambda_i \tau}{2\tau - \tau^2 \lambda_i} s - \frac{6\lambda_i}{2\tau - \tau^2 \lambda_i}$$  \hspace{1cm} (4.18)

The roots of $L_i(s)$ determine the convergence rate of the consensus filter in equation (4.8). The properties of the characteristic polynomial’s roots are studied in the following lemma.

**Lemma 4.4.3** Suppose $s_1$ and $s_2$ are the two roots of the characteristic polynomial, $L_i(s)$, in equation (4.18) and let $Re(s_1) \leq Re(s_2)$. For any fixed $\lambda_i$ the following statements are true.

- If $s_1$ and $s_2$ are real roots, then $s_2$ is a monotonically decreasing function of the communication degree $\Delta$.

- If $s_1, s_2$ are a pair of conjugate complex roots, then $Re(s_2)$ is a monotonically increasing function of the communication degree $\Delta$.

- The root $Re(s_2)$ achieve its minimum value for that value of $\Delta^*(\lambda_i)$ that renders the discriminant of the quadratic function, $L_i(s)$, equal to zero.
The proof of this and the next lemmas is given in the appendix.

The lemma tells us that for a given eigenvalue $\lambda_i$, there is a corresponding communication degree, $\Delta^*(\lambda_i)$, which minimizes $\text{Re}(s_2)$. Different eigenvalues have different optimal degrees. The system’s overall performance depends on all the eigenvalues of $A$. The following lemma examines the relationship between roots corresponding to all of the eigenvalues of $A$.

**Lemma 4.4.4** Assume that the eigenvalues of matrix $A$ are sorted in non-decreasing order so that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N < 0$.

- Let $s_\ell(\lambda_i) = \text{Re}(s_2|\lambda_i, \Delta_{\text{min}})$ for eigenvalue $\lambda_i$. When $\Delta_{\text{min}} = \min\{Q-1, 2\}$, then $s_\ell(\lambda_N) \leq s_\ell(\lambda_{N-1}) \leq \cdots s_\ell(\lambda_1) < 0$.

- Let $s_r(\lambda_i) = \text{Re}(s_2|\lambda_i, \Delta_{\text{max}})$ for eigenvalue $\lambda_i$. When $\Delta_{\text{max}} = N - 1$, then $s_r(\lambda_N) \geq s_r(\lambda_{N-1}) \geq \cdots s_r(\lambda_1)$.

- $\Delta^*(\lambda_N) \leq \Delta^*(\lambda_{N-1}) \leq \cdots \leq \Delta^*(\lambda_1)$.

Figure 4.1 illustrates the conclusions of lemma 4.4.3 for a particular system in which $Q = 3$ and $N = 20$. This figure shows the communication degree $\Delta$ vs. the real part of the root $s_2$. The dashed and dotted lines plot $\text{Re}(s_2)$ as a function of the eigenvalue $\lambda_N$ and $\lambda_1$ respectively. The maximum of $\text{Re}(s_2)$ over all system eigenvalues $\lambda_i$ for $i = 1, \ldots, N$ is also plotted in the solid line. Lemma 4.4.3 asserts that if the discriminant of the quadratic function $L_i(s)$ greater than zero, then the maximum $\text{Re}(s_2)$ is a decreasing function of $\Delta$, while if this discriminant is negative, then the maximum $\text{Re}(s_2)$ is an increasing function of $\Delta$. The degree at which the discriminant vanishes minimizes the maximum of $\text{Re}(s_2)$.

Figure 4.1 verifies the conclusions of lemma 4.4.4. Let $\text{Re}(s_2|\lambda_i, \Delta)$ denote the largest real part of the roots of the characteristic equation, when the system
Figure 4.1. Property of the eigenvalues of $A$

eigenvalue is $\lambda_i$. The figure shows that when the communication graph outdegree $\Delta = Q - 1 = 2$, $Re(s_2|\lambda_N, Q - 1)$ is smaller than $Re(s_2|\lambda_1, Q - 1)$, where as the outdegree $\Delta = N - 1 = 19$, then the order of these two quantities is reversed as asserted in the first two statements in lemma 4.4.4. The solid line in figure 4.1 draws the largest eigenvalue $\max Re(s_2)$ as a function of the node degree $\Delta$, based on all $\lambda_i$, $i = 1, \ldots, N$. The smallest $Re(s_2)$ is at the intersection of the two curves for $Re(s_2|\lambda_1)$ and $Re(s_2|\lambda_N)$, with the node outdegree is 8. The consensus filter should exhibit the fastest convergence rate with this optimal outdegree, which is characterized in the following theorem.

Consider the synchronous consensus filter given by equation (4.8) whose delay $\tau$ as a function of network connectivity $\Delta$ satisfies equation (4.9). Let $Re(s_2|\lambda_i, \Delta)$ denote the maximum real part of the roots of the system’s characteristic equation (4.18) where $\lambda_i$ is the $i$th eigenvalue of $A$. The convergence rate of the system is bounded by the maximum of $Re(s_2|\lambda_i, \Delta)$, and the corresponding optimal degree is given by the following theorem.
Theorem 4.4.1 The optimal network connectivity to maximize the convergence rate of the consensus system is given by,

\[ \Delta^* = \min_{(\lambda_i, \Delta)} \max \{ \text{Re}(s_2 | \lambda_i, \Delta) \} = \left\lceil \sqrt{\frac{0.3Q}{\gamma e} - 1} \right\rceil. \]

Proof: According to Lemma 4.4.4, the fastest convergence is achieved at the degree of which \( \text{Re}(s_2 | \lambda_1, \Delta) = \text{Re}(s_2 | \lambda_N, \Delta). \) That is,

\[ \frac{3 + 2\tau - \lambda_1\sqrt{R_1}}{2\tau - \tau^2 \lambda_1} = \frac{3 + 2\tau \lambda_N}{2\tau - \tau^2 \lambda_N}. \]

Solving the equation gives that \( \Delta^* = \sqrt{\frac{0.3Q}{\gamma e} - 1}. \) \[ \square \]

To explicitly show the effect of the connectivity on the convergence of the consensus system, we solve the consensus dynamics in the following theorem.

Theorem 4.4.2 Consider the synchronous consensus dynamics in equation (4.8)

\[ \dot{x} = Ax(t - \bar{\tau}) + r \]

where the eigenvalues of \( A \) satisfy the inequality (4.15). Assume the initial condition is \( x(0) \), and the steady state is \( x(\infty) \), then

\[ \|x(t) - x(\infty)\| \leq C(\Delta)e^{-J(\Delta)t}(\|x(0)\| + 2kr) \] (4.19)

where \( k = \gamma e \) and

\[ J = \begin{cases} \frac{3k(1+\Delta)^2 + 2(1+\Delta)}{2 + k(1+\Delta)^2} & \Delta \leq \sqrt{\frac{0.3}{k} - 1} \\ \frac{3 - 2k(1+\Delta)(3\Delta + 1)}{2k(1+\Delta)^2(3\Delta + 1)} & \text{otherwise} \end{cases} \]
Proof: Taking the inverse Laplace transform on equation (4.16) yields,

\[
x(t) = U \mathcal{L}^{-1} \left\{ \text{diag} \left( \frac{1}{s - \lambda_i e^{-\tau s}} \right)_{N \times N} \right\} U^T \cdot \left( x(0) + \text{diag} \left( \frac{2\tau - \tau^2 \lambda_i}{\lambda_i} \right)_{N \times N} r \right) + U \mathcal{L}^{-1} \left\{ \text{diag} \left( \frac{-2\tau - \tau^2 \lambda_i}{s \lambda_i} \right) \right\} U^T r
\]

The second term in the above equation represents the system steady state \( x(\infty) \), so the consensus error can be written as,

\[
\|x(t) - x(\infty)\| \leq \left\| \mathcal{L}^{-1} \left\{ \text{diag} \left( \frac{1}{s - \lambda_i e^{-\tau s}} \right) \right\} \right\| \cdot \left\| x(0) + \text{diag} \left( \frac{2\tau - \tau^2 \lambda_i}{\lambda_i} \right) r \right\| 
\]

\[
\leq Ce^{\max\{Re(s)\} t} \left\| x(0) + 2 \frac{\epsilon}{Q} r \right\|
\]

where \( \max\{Re(s)\} \) is the maximal value of the roots of all subsystem \( L_i(s) = s - \lambda_i e^{-\tau s} = 0 \) in equation (4.18), and \( C \) is a function of the communication degree. The expression of \( \max\{Re(s)\} \) and \( C \) is obtained easily from lemma 4.4.3 and theorem 4.4.1.

In this chapter, define that the convergence is achieved when the error between the consensus state and stable state is less than a given constant \( \epsilon \), or \( \epsilon \)-consensus. That is, \( \|x(t) - x(\infty)\| \leq \epsilon \left\| x(0) + 2 \frac{\epsilon}{Q} r \right\| \), or the convergence time \( T_c \) is

\[
T_c \leq \frac{\log \left( \frac{\epsilon}{\epsilon} \right)}{\max Re(s)} \tag{4.20}
\]

therefore, the degree \( \Delta^* \) given in Theorem 4.4.1 minimizes the time needed to achieve convergence.
4.4.2 Convergence Rate of Asynchronous Consensus

Analogous to the convergence rate analysis in section 4.4.1, we first take the eigen-decomposition of the symmetric matrix \( \mathbf{A} \), and apply Pade approximation to simplify the consensus dynamics of equation (4.13) as

\[
\mathbf{X}(s) = \text{Udiag} \left( s - \lambda_i \frac{q}{1 - e^{\gamma s}(1 - q)} + (1 + 2\Delta) \frac{(1 - q)(1 - e^{\gamma s})}{1 - e^{\gamma s}(1 - q)} \right)^{-1} \text{U}^T \left( \mathbf{x}(0) + \frac{r}{s} \right)
\]

where \( q = p \left( 1 - \frac{p}{q} \right)^{\Delta - 1} \), and \( \lambda_i \) is the eigenvalue of the matrix \( \mathbf{A} \), satisfying the inequality (4.15). Let \( L_i(s) \) denote the characteristic polynomial for the \( i \)th subsystem. For the asynchronous consensus system, \( L_i(s) \) is given by

\[
L_i(s) = s^2 + \left( \frac{q}{(1 - q)\gamma} + 2\Delta + 1 \right) s - \frac{\lambda_i q}{(1 - q)\gamma}
\]  \hspace{1cm} (4.21)

The roots of \( L_i(s) \) determine the convergence rate of the consensus filter in equation (4.13). According to the property of the roots, the following lemma proves that the asymptotic consensus is achieved in an asynchronous manner.

**Lemma 4.4.5** The asynchronous consensus filter given in equation (4.13) is stable.

**Proof:** Since \( \lambda_i < 0 \), it is straightforward that the roots of the quadratic equation (4.21) are on the left side of s-plane. Therefore, the asynchronous consensus system can always achieve convergence. \( \blacksquare \)
The following theorem gives the optimal degree that corresponds to the fastest convergence rate of the system.

**Theorem 4.4.3** Consider the asynchronous consensus filter given by equation (4.13) whose delay satisfying the geometric distribution in equation (4.10). The optimal network connectivity is approximately given by,

\[ \Delta^* \approx \log_\alpha [2c \log_\alpha (2c \log_\alpha (3c + 3c) + 3c] \]

and the convergence rate \( J \) is,

\[ J = \frac{1}{2} \left( \frac{q}{(1-q)\gamma} + 2\Delta + 1 - \sqrt{\left( \frac{q}{(1-q)\gamma} \right)^2 + (2\Delta + 1)^2 - 2 \frac{q}{(1-q)\gamma}} \right) \]

where \( \alpha = 1 - \frac{p}{Q} \), \( c = \frac{(1-\frac{p}{Q})\gamma}{p\alpha} \) and \( q = p(1 - \frac{p}{Q})^{\Delta-1} \).

**Proof:** The polynomial \( L_i(s) \) has either a pair of conjugate roots as \( \lambda_i < -(2\Delta + 1) \), or two negative real roots as \( \lambda_i > -(2\Delta + 1) \). The convergence rate of the system is determined by the maximal root of \( L_i(s) \) for \( i = 1, \cdots, N \). Based on the inequality (4.15) of \( \lambda_i \), it is clear that the maximal roots for all polynomials, \( i = 1, \cdots, N \), is that of \( \lambda_i = -1 - \Delta \). That is,

\[ J = -\frac{1}{2} \left( \frac{q}{(1-q)\gamma} + 2\Delta + 1 - \sqrt{\left( \frac{q}{(1-q)\gamma} \right)^2 + (2\Delta + 1)^2 - 2 \frac{q}{(1-q)\gamma}} \right) \]

Letting the first order derivative of \( J \) equal to zero gives

\[ q \log \alpha + 2\gamma - \frac{q^2 \log \alpha - q\gamma \log \alpha + (4\Delta + 2)\gamma^2}{\sqrt{q^2 - 2q\gamma + (2\Delta + 1)^2 \gamma^2}} = 0 \]
where $\alpha = 1 - \frac{p}{Q}$, or

\[
4q^2 + \left(\frac{4}{\log \alpha} + (2\Delta + 1)^2 \log \alpha - 8 - \log \alpha - (8\Delta + 4)\right) \gamma q \\
+ \left(4(2\Delta + 1)^2 + (8\Delta + 4) - \frac{8}{\log \alpha}\right) \gamma^2 = 0
\]

For a time slot $\gamma$ that is short enough, the equation above can be approximately solved as

\[
q = p(1 - \frac{p}{Q})^{\Delta - 1} \approx (2\Delta + 1) \gamma
\]

Therefore,

\[
\Delta^* \approx \log_\alpha \left[2c \log_\alpha (2c \log_\alpha 3c + 3c) + 3c\right]
\]

where

\[
c = \left(1 - \frac{p}{Q}\right)^\gamma.
\]

4.5 Energy Efficiency for Synchronous Consensus

The preceding section identified network connectivities that minimize the time for $\epsilon$-consensus. This may be the preferred problem in situations such as the swarming under consensus model [49] in chapter 3, where the consensus filter is used to guide the movement of a dynamical swarm. In other sensor network applications, however, it may be more interesting to in minimize the “energy” required to achieve $\epsilon$-consensus.

Assume a disk model for wireless radio network in which each disk has an equal number of regularly spaced neighbors. This may be simplistic for randomly deployed sensor networks. However for the swarming under consensus model [49,
it was shown that agents usually converge to swarms such that agents are regularly spaced with nearly constant distances between neighbors.

Therefore, the $N$ agents are assumed to be uniformly distributed with a density of $\rho$. If $r$ is the radio transmission radius, then the average number of agents in a given disk will be $1 + \Delta = \rho \pi r^2$. The average power at the receivers (located at the edge of the disk) will be $P_R = P_T r^{-\alpha}$ where $\alpha$ is the path loss exponent and $P_T$ is the transmit power. Suppose $P_R$ is the minimum received power required to assure a successful reception of the transmitted message, then the transmit power is

$$P_T = (\rho \pi)^{-\alpha} \bar{P}_R (1 + \Delta)^{\frac{\alpha}{2}}$$

(4.22)

Let $T_c$ be the time elapsed for the consensus filter to achieve $\epsilon$-consensus, from an initial state $\mathbf{x}(0)$ and with a constant input $\mathbf{r}$. According to equation (4.20), we have

$$T_c \geq -\frac{1}{J(\Delta)} \ln \frac{\epsilon}{C(\Delta)}$$

(4.23)

If $\epsilon \ll 1$, then we can treat the $\ln(\epsilon/C(\Delta))$ term as a constant that is independent of $\Delta$. The parameter dominating the convergence time is therefore $T_c \propto \frac{1}{J(\Delta)}$.

The optimal $\Delta^*$ that minimizes $T_c$ has been presented in the preceding section. Here we consider for an energy-efficient communication, i.e., to minimize the total energy such as

$$\text{Total Energy} = P_T \cdot T_c \propto \frac{(1 + \Delta)^{\alpha/2}}{J(\Delta)} \equiv E(\Delta)$$

The following theorem characterizes the energy-efficient $\Delta^*$. 

112
Theorem 4.5.1  The optimal degree to minimize the energy consumed for $\epsilon$-convergence is given by

$$\Delta_E^* = \arg\min_{\Delta} E(\Delta)$$

s.t.  \( Q - 1 \leq \Delta \leq \sqrt{\frac{\pi}{2k}} - 1 \)

where \( k = \frac{\gamma e}{Q} \) and \( E(\Delta) \) is defined in equation (4.24). The optimal degree \( \Delta^* \) is

$$\Delta_E^* = \begin{cases} \max\left\{ \left\lfloor \sqrt{\frac{0.3Q}{\gamma e}} - 1 \right\rfloor, Q - 1 \right\} ; & \alpha \leq \alpha_0 \\ \max\{2, Q - 1\} ; & \alpha > \alpha_0 \end{cases} \quad (4.24)$$

where

$$\alpha_0 = \begin{cases} 4 \left( \frac{\ln 1.26 \frac{3^{3/2} k^2}{Q^2}}{\ln \frac{3^{3/2} k^2}{Q^2}} + 0.5 \right) ; & Q \leq 3 \\ 4 \left( \frac{\ln 1.26 \frac{3^{3/2} k^2}{Q^2}}{\ln \frac{3^{3/2} k^2}{Q^2}} + 0.5 \right) ; & Q > 3 \end{cases} \quad (4.25)$$

Proof:  Denote \( x = 1 + \Delta \). According to the second item of lemma 4.4.3, we know the cost function \( E(\Delta) \) is monotonically increasing with \( \Delta \) when \( x > \sqrt{\frac{0.3Q}{\gamma e}} \).

Therefore, we only consider optimization problem in the region that \( Q \leq x \leq \sqrt{\frac{0.3Q}{\gamma e}} \). We use the approximation of characteristic polynomial in the proof the first item of lemma 4.4.3 to have,

$$J(x) = \frac{3 - 2kx^2 - \sqrt{9 - 24k^2x^2 - 2k^2x^4}}{2kx + k^2x^3}$$

$$\approx \frac{3 - 2kx^2 - (3 - 4k^2x^2 - 3k^2x^4)}{2kx + k^2x^3}$$

$$= 3x - \frac{4x}{2 + kx^2}$$
and hence the energy cost function can be written as

\[ E(\Delta) = \frac{x^2(2 + kx^2)}{2x + 3kx^3} = \left( \frac{t}{k} \right)^2 \frac{2 + t}{2 + 3t} \]

where \( t = kx^2 \), and \( \beta = \frac{\alpha}{4} - \frac{1}{2} \). Since \( 2 \leq \alpha \leq 4 \), then \( 0 \leq \beta \leq \frac{1}{2} \). Taking the second-order derivative of \( E \) related to \( t \) gives

\[
\frac{d^2E}{dt^2} = \frac{1}{k^3(2 + 3t)^3} \cdot \left[ 4(\beta^2 - \beta)t^{\beta-2} + (22\beta^2 - 26\beta)t^{\beta-1} + (6\beta^2 - 18\beta + 12)t^\beta \\
+ 9(\beta^2 - \beta)t^{\beta+1} \right] < 0
\]

for \( 0 \leq t \leq 0.3 \). Hence, \( E \) is a concave function of \( t \), which implies that \( \min_x E = \min\{ E(\max \{ Q, 3 \}) \}, E(\sqrt{\frac{0.3}{k}}) \} \).

Let \( E(\max \{ Q, 3 \}) = E(\sqrt{\frac{0.3}{k}}) \), we have

\[
\alpha_0 = \begin{cases} 
4 \left( \frac{\ln 1.26 + \frac{2 + 9k}{2 + 27k}}{\ln \frac{k^3}{3k}} \right) + 0.5 & Q - 1 \leq 2 \\
4 \left( \frac{\ln 1.26 + \frac{2 + kQ^2}{2 + 3kQ^2}}{\ln \frac{kQ^2}{k}} \right) + 0.5 & (Q - 1 > 2)
\end{cases}
\]

To summarize, the optimal degree \( \Delta^* = x^* - 1 \) is

\[
\Delta^*_E = \begin{cases} 
\max \left\{ \left\lfloor \sqrt{\frac{0.3}{k}} \right\rfloor - 1, Q - 1 \right\} & \alpha \leq \alpha_0 \\
\max\{2, Q - 1\} & \alpha > \alpha_0
\end{cases}
\]

Note that the energy-efficient \( \Delta^* \) has an interesting threshold behavior in that it is either \( \left\lfloor \sqrt{\frac{0.3Q}{\gamma^e}} - 1 \right\rfloor \) or \( Q - 1 \). In relatively lossy environments (\( \alpha > \alpha_0 \)), we optimize energy efficiency by adopting a sparsely connected network. In a relatively lossless environment (\( \alpha \leq \alpha_0 \)), we achieve better energy-efficient by
increasing network connectivity to that level that minimizes the time $T_c$ to $\epsilon$-consensus.

4.6 Simulation

A Matlab script was written to simulate the behavior of consensus filters on regular graphs with various node out-degrees $\Delta$. The following simulations assumed $N = 40$ nodes with $Q = 2$ sub-channels and a nominal time-slot length of $\gamma = 0.004$ seconds.

Figure 4.2 shows the synchronous consensus convergence rate, denoted as $-J(\Delta)$ with different network topologies $\Delta$. The system diverges when $-J(\Delta) > 0$, or $\Delta > 9$. Lemmas 4.4.3 and 4.4.4 assert that $-J(\Delta)$ is a decreasing function for $\Delta < 6$ and an increasing function for $\Delta > 6$. Hence the optimal out-degree that minimizes the converge time, $T_c$, to $\epsilon$-convergence is 6, which agrees with
Figure 4.3. The optimal connectivity $\Delta$ for synchronous consensus

Figure 4.3 shows the norm of the total consensus error with time for a synchronous consensus system. The set of curves corresponds to different node degrees. The fastest convergence is shown to be achieved with $\Delta = 6$. As $\Delta$ increases, the trajectory becomes highly oscillatory, and eventually becomes unstable when $\Delta > 9^1$. The results closely match the analytical predictions in Figure 4.2.

In comparison, Figure 4.4 plots the convergence rate of the asynchronous consensus with respect to $\Delta$. The asynchronous system parameters are the same as in the synchronous case, including the broadcast probability $p$. Note that the broad-

\footnote{Note that it can be difficult to construct regular graphs for arbitrary $N$. In the simulations shown in Figure 4.3, we chose graphs in which 90\% of the nodes had the same out degree.}
cast probability \( p = \frac{Q}{1+\Delta} \) is optimal for synchronous consensus filter, but not for the asynchronous consensus filter. That results in some performance degradation in the asynchronous system as shown in the comparison. The figure shows the system is stable since \(-J(\Delta)\) is always less than zero, as Lemma 4.4.5 predicts. The optimal out-degree to optimize the asynchronous convergence rate is 9, which is larger than the synchronous case. That is because the average delay in synchronous consensus is evaluated by \( \tau_i = \max \tau_{ij} \) for \( j \in \mathcal{N}_i \), which is longer than the delay \( \tau_{ij} \) in asynchronous update principle. The shorter message delay implies a tighter optimal connectivity for asynchronous cases.

Figure 4.5 shows the system responses for asynchronous consensus system by presenting a set of error norms corresponding to different node degrees. The fastest convergence is achieved with \( \Delta = 10 \), as displayed in figure 4.4. An interesting observation is that, comparing with the synchronous scheme, the asynchronous filter achieves the \( \epsilon \)-consensus slower. Figure 4.3 demonstrates that the synchronous filter, working in the optimal connected graph, accomplishes the \( \epsilon \)-consensus within
150 time slots, while asynchronous filter needs at least 300 time slots. The observation agrees with the fact that $-J(\Delta^*)$ value is smaller in synchronous case than asynchronous, referred to Figure 4.2 and 4.4. An intuitive explanation is that in the synchronous scheme, agent $i$ uses the difference of two delayed information $x_j(t - \tau_i) - x_i(t - \tau_i)$ to update its current state. On the contrary, agent $i$ regulates its state based on its current state and the neighbors’ delayed message $x_j(t - \tau_{ij}) - x_i(t)$ in asynchronous mode, where the estimated error is less accurate and has impact on the consensus performance, resulting in a slower convergence rate.

The asynchronous consensus, however, has specific advantages in real world applications. First, the consensus algorithm guarantees a stable system perfor-
mance. Secondly, the implementation or system complexity is lower, since individual agents operate on alternating between sending and receiving in solely distributed manner. Unlike the synchronous update, the asynchronous consensus is not responsible to cooperate with its neighbors. The communication cost, therefore, is dramatically reduced.

In the end, Figure 4.6 summarizes results from several simulations that were used to study energy-efficient consensus for networks with \( N = 40 \) nodes and \( Q = 2 \) sub-channels. Again we chose the time slot duration \( \gamma = 0.004 \) seconds and measured the total energy required to achieve \( \epsilon \)-consensus where \( \epsilon = 10^{-6} \). Figure 4.6 plots the total energy cost versus network degree, \( \Delta \) for path loss exponents of \( \alpha \) of 2, 3, and 4. These plots show that the energy cost is a concave function over the interval \( \Delta \in [2, \Delta^*] \), where \( \Delta^* \) is the optimal degree to minimize the convergence time given in Theorem 4.4.1. The energy is monotonically increasing for \( \Delta > \Delta^* \). As a result, the optimal communication degree, \( \Delta^* \) occurs at one of two boundary points of the concave region, either when \( \Delta = Q - 1 \) (2 in the example) or \( \Delta^* = 6 \). From Theorem 4.5.1, we expect a smaller \( \Delta^*_E \) when \( \alpha > \alpha_0 \). For this case, \( \alpha_0 \) is about 2.8, thereby suggesting that the optimal degree, \( \Delta^*_E \) equals to \( Q - 1 = 2 \) when \( \alpha = 4 \). This is indeed as in Figure 4.6. For \( \alpha < \alpha_0 \), \( \Delta^*_E \) is expected to be 6. Figure 4.6 shows that this \( \Delta^* = 6 \) for \( \alpha = 2 \), which is again consistent with the analytical predictions. As \( \alpha = 3 \) which is close to \( \alpha_0 = 2.8 \), the energy consumed at both boundary.
Figure 4.6. Energy efficient convergence
CHAPTER 5
MULTIPLE ROBOT TESTBED

5.1 Overview

Currently there is great interest in controlling decentralized and networked systems. During the last decade, technological advances in computation and communication have provided efficient and inexpensive means to build laboratory hardware platforms for the research of decentralized control. These platforms work as proof-of-concept implementations which can demonstrate the effectiveness of the coordinated control techniques. Though the facilities needed to build these platforms can be affordable for academics, deployment and development are still very time and energy consuming. Thus, the demand for a software simulation system has been strong. Even when a hardware platform is readily accessible, the simulation can serve as a fast prototyping tool. In this chapter, we describe the software simulation system we used to test the algorithms we developed for distributed cooperation in a Multi-Agent System (MAS). We selected the Player/Stage project as our platform because it is a very popular choice for this type of applications. It provides all the fundamental environment variables and facilities needed for the simulation of a MAS. Also, the migration from the software simulation system to real hardware can require very little effort, for certain target hardware platforms

1. In what follows in this chapter we introduce our software simulation system as

1Assuming that the software at the client/agent side is well written.
well as the hardware testbed.

5.2 Introduction

The goal of building the hardware testbed and software simulation system is to study the control of distributed systems with regard to limited communication resource. Specific accomplishments under this work include:

* A simulator based on Player/Stage project for the coordinated control of multi-agent robotic systems. We studied how swarms of autonomous robots can coordinate their behavior to ensure swarm stability while moving towards a specified target.

* A laboratory robotic vehicle testbed built to measure the distributed system cooperation performance subject to the limit communication resource. The testbed uses two types of robotic systems:

  - Mica-KoalaBot: Koala Robots controlled by the MICA2 processor, programmed using TinyOS, and communicating over an ad hoc radio network.
  - Pioneer Robot Swarm: ActivMedia Pioneer 3-DX robots, controlled by a x586 embedded PC running Linux, and communicating over a wireless 802.11 LAN.

The rest of this chapter is organized as follows. Section 5.3 introduces the multi-robot simulator software. Section 5.4 gives some details on the hardware testbed.
5.3 Multi-robot Simulator

5.3.1 Player/Stage Project

The Player/Stage free software project has been briefly reviewed in section 1.2.4 in the introduction chapter. In this chapter, we introduce some under-explored research opportunities opened up by the Player/Stage Open Source tools.

![Diagram](image)

Figure 5.1. Infrastructure of the Player/Stage Project

The project provides the *Player* robot device server and the *Stage* multiple robot simulator, plus supporting tools and libraries. The infrastructure of the project is shown in Figure 5.1. The infrastructure implements the network-centric client-server architecture: player servers support a variety of robotic device control, and client robot control programs communicate with the devices by exchang-
ing the message over TCP socket-based interface.

The encapsulation of devices accesses by proxies eliminated the need for the client to communicate with device drivers directly. Though some device-specific features may not be provided by the proxy, the Player/Stage project has supports for a variety of devices from many different manufactures. Features for certain type of device are well abstracted and encapsulated by the proxy. It is actually desirable for the client’s algorithm not to rely on any special feature of a device available only from some specific vendor.

In the simulation system, proxies communicate with the environment server through sockets. This gives great flexibility to building distributed coordinated control systems. The clients can be independent processes, maybe even running on different computers from where the server is. Building the client application this way will further reduce the changes needed when porting the program to the OS’es on real robots.

The simulation system provides ways of customization; we can select only the devices we need and only environment elements of our interests. In addition to robotic devices associated with sensors and actuators, the system allows some drivers to be added for realizing certain abstract functionality. For instances, the ‘fixedtones’ drive implements a Fast Fouries Transform on the measured raw data and reports the corresponding frequencies and amplitudes of the highest peaks in the frequency domain.

In this work, in order to simulate wireless data communication, we created a device called ‘sbsrelay’. We implemented the driver and proxy for it and used it in our clients as the communication device. More details about this driver can be found in the section 5.3.2.
As a result of the Player’s device model, client control algorithm can largely ignore the details of the low-level robotic device. The server provides a collection of generic devices, unrelated to the exact mobile robot bases. It makes it very convenient for researches to verify their ideas for controlling real robots.

5.3.2 Multi-robot Simulator

Player makes it possible for clients to read data from sensors and send commands to the actuators to generate movements. Stage provide an efficient and convenient way to evaluate the client control algorithms. The Player/Stage project supports the framework for studying the multi-robot system coordination control. In this work, we take the advantage of this feature to explore how swarms of autonomous robots can cooperate their behavior to ensure swarm stability while moving towards a specified target. In this section, we describe the structure of our multi-robot simulator program developed based on the Player/Stage project, and present our contributions and test results.

Sbsrelay Device and Proxy

Although not being very comprehensive and sophisticated, the Player/Stage project provides no device that implements real wireless message exchange. In order to study the multi-robot coordination performance over wireless network on Stage simulator, we, therefore, developed a virtual device, named sbsrelay, to simulate the messages exchange among the each individual robot in a communication network. Based on this sbsrelay device, we are allowed to build a simulator to measure the swarm cohesive performance under consensus filter guidance, as described in chapter 3.

Sbsrelay device driver is a module that controls the virtual communication
device, which one can transmit to and receive from these formatted message packets.

The basic interface of Sbsrelay is its device proxy, called relayproxy. Its API is listed as the following:

- **int GetRelayDataCount (void)**
  the function returns the size of data received by the device

- **int GetRelayData (int32_t dst, int32_t & src, player_opaque_data_t & outData)**
  the function is used to retrieve data from a sbsrelay device

- **int SendRelayData (int32_t src, int32_t dst, player_opaque_data_t & pData)**
  the function is used to send data to a sbsrelay device.

- **int SendCmd(int cmd = PLAYER_RELAY_CMD_CLEAR)**
  the function can send a command to a sbsrelay device.

Each client is assigned a unique identification number (id). Relay messages are associated with source and destination id’s. The relay proxies at client side all talk to server side driver(s). Each driver takes the messages and route them to the queues, stored within the driver, of corresponding clients. The clients need to send explicit requests to retrieve their received messages from the server. Every server side driver will treat all of its messages as one type and route them accordingly. For multiple message types, one can use multiple drivers, e.g., one for relaying position data and another for estimation data.
If a client fails to retrieve its message in the timely manner, old messages will be overwritten by new ones. Once a message is retrieved by the client, the driver clears it from the queue. There are mainly two reasons why we designed the device to work this way: first, for our application, keeping old position data makes no sense as the newest one will help the clients do a better calculation/estimation. Secondly, it is much simpler to implement this replacing rule than to queue up the data and manage situations like queue overflow and etc. The simple message replacing rule suffices.

The proxy provides the necessary interface to the device and shields off some details. But it is also very low level because you have to pack and unpack the data messages and handle user errors like inappropriate message types, invalid data lengths and etc\(^2\). So we designed higher level abstraction and encapsulation by introducing the Relay class. The API of the Relay class is more straight forward (two relays are used, one for pose and one for estimation); packet packing and unpacking is done automatically:

- **void sendPose (int32_t src, int32_t dst,
  player_pose_t & pose)**
- **void sendEstimate (int32_t src, int32_t dst,
  player_pose_t & pose)**
- **bool getPose (int32_t dst, int32_t src,
  player_pose_t & pose)**
- **bool getEstimate (int32_t dst, int32_t src,
  player_pose_t & pose)**

\(^2\)Note that the relay device does generic message relay and should not be limited to a particular message format, though it does have its own requirements on formatting. So the proxy should also be generic as not to limit the capability of the device.
**Client Control Program** Consider the distributed nature of the swarm coordination behavior, we tried best to implement the algorithm in a similar way; there is no central control. Each robot is controlled under a particular client program running in a separate process. Client programs run in parallel. In other words, there is not a central base to schedule the client execution order. Based on this design of the client control program, our multi-robot simulator is capable to comprehensively reproduce the real world robot performance. This reproducing task is impossible to be realized by Matlab program. More importantly, the verified client control program has closer correlation to what would be used on real robots and thus is easier to transform to the real robot vehicles. We introduce the distributed client control program routines at follows.

The main classes consist of the client program are: SwarmAgent, Position, Fiducial, Sonar and Relay class. Their principle functions and relationship is shown in the figure 5.2. The class SwarmAgent carries out most execution routine to manipulate a robot. Other four classes support the basic detection and low-level control service. Position class is used to retrieve the robot’s position and generate the motor movement commands, such as turn and move. The purpose of Fiducial class is to find out the nearest neighbors and their position. Robot’s communication messages are exchanged via Relay class. Obstacle avoid is realized through Sonar class. Note that the Sonar class detects both neighboring vehicles and other environmental objects. At the following, we are confine to describe the SwarmAgent class.

The SwarmAgent class control routine is shown in the figure 5.3, based on the research of swarm coordination behavior under consensus in the chapter
Class Fiducial:
Detect neighbor’s position and identify the nearest neighbors

Class Position:
Retrieve 2D position and control motor

Class Sonar:
Detect the obstacle and neighboring agent to avoid collision

Class Relay:
Simulate data communication

Figure 5.2. Main classes in client program and flows
Estimate the neighbor center: calcCenter ();
\[ \hat{x}_i = \frac{x_i + \sum_{j \in N_i} x_j}{|N_i| + 1} \] estimate center of client i

Transmit local position and estimate center to neighbors:
sendPose();   sendEstimate();

Get neighbors positions and estimates:
updateNbrEst();  getPose();

Calculate the mutual force between robots : calcForce();
\[ f_i = \sum_{j \in N_i} \left( \frac{\rho_0}{\|x_i - x_j\|^2} (x_i - x_j) - \frac{\alpha_0}{\|x_i - x_j\|} (x_i - x_j) \right) \]

Consensus the neighbors’ estimates and head to Target :
updateEstimate();  headHome();  (x0: target, T: comm. freq)
\[ \hat{x}_i [k+1] = \hat{x}_i [k] + T \cdot \left( (x_0 - \hat{x}_i) + \sum_{j \in N_i} (\hat{x}_j - \hat{x}_i) + \sum_{j \in N_i} (x_j - \hat{x}_i) \right) \]
\[ x_i [k+1] = x_i [k] + T \cdot \left( (x_0 - \hat{x}_i) + f_i \right) \]
sonar.getNearestObjs does the minor change on motor direction

Figure 5.3. SwarmAgent classes chart flow
3. During the initialization, client $i$ uses the neighbors and self position to estimate the local neighborhood center $\hat{x}_i[0]$, then transmit the estimate to its neighbors. Later, client $i$ adjusts its estimate to $\hat{x}_i[k + 1]$ in terms of its old estimate $\hat{x}_i[k]$ and the received its neighbors’, client $j$, information $\hat{x}_j[k]$, where $j \in N_i$ and $N_i$ denote the neighbor list of client $i$. The adjustment follows the distributed consensus algorithm described in the chapter 3. Meanwhile, client $i$ calculates the interaction $f_i$ in the neighborhood to guarantee the cohesive movement format while avoiding collision. At the end of each iteration, client $i$ decides its move speed and direction according to the knowledge of the difference between target $x_0$ and its current estimate $\hat{x}_i[k + 1]$, as well as the force $f_i$.

Individual client control routines run independently without a central administrator. They only change their states reciprocally through the virtual device proxy. Based on this client control program, the simulator has ability to make in imitation of real robot cooperation performance over wireless network.

**Multi-robot Simulator** Stage is specifically designed to support multi-robot research through simulating a population of mobile robots, sensors and environment objects in a two-dimensional bit-mapped environment. A great deal of time and energy has been saved for the theoretical-oriented study of efficient development of a feasible control algorithm.

In this work, we leverage the features of the stage simulator to experiment our swarm cohesive under consensus study in chapter 3. The set-up is done through two configuration files: .world and .cfg. The .world file describes the environment elements and gives properties of each individual sensor. If
a sensor does not have any options specified, it may not show up in this file. The .cfg file sets up the bonding between the Stage and Player by assigning the sockets’ port numbers where the driver will be providing simulation services. The bonding relationship between Stage and Player is shown in the figure 5.1. Client control algorithm is then implemented onto each modeled robot through the Player interface. In other words, client program cannot even tell the difference between the simulated robots from a real one. Figure 5.4 plots the screen shot of the stage simulator. In this figure, each little square represents a simulated robot, and the trajectories are the trails of robots’ movement. At the beginning, five robots are randomly spaces near to the point (-4, -4), and the target is set at (7,7). Being governed by the the client control algorithm, these robots cooperate with each other under communication message exchange and head to the target. They stop when
their estimate center is closed to the target. The simulator update frequency is 0.1 second. The simulator results once again verify our algorithm for controlling swarm cohesive behavior under consensus.

5.4 Multi-robot testbed

5.4.1 MICA-KoalaBot

This testbed uses the MICA2 sensor node processor module to control the Koala autonomous robot. The testbed is used for testing algorithms for coordinating the behavior of multiple robots over ad hoc wireless communication networks. In this testbed, the estimation guidance movement method is adopted from the client control program described in section 5.3. We categorize the introduction on the testbed into three parts: Hardware, control algorithm and the graphic user interface (GUI).

MICA-KoalaBot Hardware

Koala Robotic Vehicle The robotic vehicle is the Koala robot from K-Team Inc, as shown in figure 5.5. The Koala is a mid-size robot designed for real-world application [78]. It rides on six wheels for mild all-terrain operation (it can not climb deep slopes but can handle small steps and non-paved surface, like lawn). Wheel positions are measured by encoders on the wheels. Commands are issued to the robot over an RS-232 serial port as ASCII strings. After receiving the command, the robot responds with an ASCII string. The basic command strings are listed as follows,

- \textbf{D},x,y-set speed of right and left wheels to \( x \) and \( y \), respectively.
- \textbf{E}-get wheel speed
• G,x,y-set wheel encoder counters (right/left)

• H-get wheel encoder counter values.

• N-get proximity sensor measurements

The Koala robot is equipped with sixteen infra-red proximity sensors that are placed around the robot, measuring the normal ambient light and the light reflected by obstacles. Measurement results are converted to a 10-bit output which can be accessed by the computation unite connected to the robot. For instance, the response to the 'N' command, getting proximity sensor measurements, would be a string such as

'n,21,13,14,15,16,20,255,233,250,120,34,23,24,25,45,50'

which is the echoed lower case command letter followed by the 16 sensor measurements using commas as token separators.

**Berkeley Mica2 Mote** A mote is a tiny computer with CPU and memories and powers itself, normally with batteries. The Mica2 processor is a third generation mote module that is manufactured by Crossbow Inc, as shown in
The Mica2 mote platform is built above the Atmel Atmega 128L processor which operates at 4 MHz. The Mica2 mote has 512 Kbytes of flash memory, a 4 Mbit serial flash, 4Kbytes of SRAM and a 4Kbyte EEPROM. The mote is able to communicate with the wireless network in either 433MHz or 868/916 MHz bands, with a data rate of 38.4 kbps and a maximum outdoor range of 100 feet. It is powered by 2 AA batteries, independently, so that the energy available to the mote is severely limited for transmitting messages. For example, a transmission (50\text{mW}) costs almost double power than receiving (29\text{mW}), and the power consumption is 0.6\text{\mu W} in the idle status.

A pre-compiled control programs is downloaded from PC to Mica2 mote through a programming serial I/F board. Figure 5.7 shows MIB500 parallel port programming board used in this work. In MICA-KoalaBot testbed, the MICA2 processor is connected to the rear serial port on the Koala Robot using the MIB500 programming board, as shown in figure 5.8. The MIB500 board is fastened to the Koala robot. Power for the MICA2 is obtained from a 5 volt output connector on the Koala robot.
Figure 5.7. Parallel port programming serial I/F board

Figure 5.8. MICA-KoalaBot
MICA-KoalaBot Control Algorithm and Challenge

The MICA-KoalaBot runs TinyOS. TinyOS is an object-oriented operating system developed at U.C. Berkeley to support the development of embedded sensor networks. NesC [31] is a system programming language for TinyOS, which is an extension to the C programming language. NesC-based TinyOS programs consist of software objects, called components, that communicate with each other through well-defined bidirectional interfaces. Hence, all computation within the system is in response to some received signal that may be generated from the external environment, such as a sensor signal, from the processor hardware, i.e. a clock or radio, or from other TinyOS components. The MICA-KoalaBot software can, therefore, be viewed as a set of interconnected TinyOS components that encapsulate various tasks ranging from low-level control of the UART connection between the MICA2 and Koala to high-level updating of the vehicle’s physical state.

Our MICA-KoalaBot control algorithm components are arranged in three layers. The robot control layer is the highest layer to generate the robot cooperate functions, the second translation layer translates the upper-level functions into ASCII strings that are sent to the UART. The lowest low-level layer consists of the basic TinyOS components that control the MICA2 hardware. They are discussed in detailed below.

The Planner component is in charge of coordination control of multiple robot, which involves issuing commands to trajectory planning and navigation in a manner that realizes a global goal. This component is responsible for determining which BasicBehavior component commands should be issued to the Robot. This component also provides a command interface that allows users to monitor and
Figure 5.9. MICA-KoalaBot control algorithm chart flow
control the robot remotely. The BasicBehavior component implements a state machine (see figure 5.10) through a set of motion primitives. These primitives are 1) moving the robot a specified distance and 2) turning the robot through a specified angle. This component also coordinates the program’s access to the Koala robot’s IR sensors and wheel encoders. It maintains the “global” position state of the robot, as measured in the robot’s local Cartesian coordinate.

The translation layer translates the high level commands from the BasicBehavior component into ASCII commands that a robot can understand. The translations are done based on Koala robot translation protocol. To be more specific, first the KoalaCmd component parses the high-level instruction commands for robot’s movement to the corresponding coordinate system, then the KoalaBase component generates one or several low-level ASCII command strings to complete a parsed high level command. In the UART component, KoalaPacket function packs the encoder command into a byte-based communication sequence prepared for UART transmission. The last function HPLUART is the UART driver of a Mica2 mote.

Figure 5.10. BasicBehavior state machine
One of remaining components is called Sync component. It is used to accomplish the synchronization routine. Coordinated multi-robot control often requires that robots agree to act at specific times. One specific reason for this is to coordinate access to the radio channel. In MICA-KoalaBot testbed, we need clock synchronization to coordinate when the robots checked their IR proximity sensors. Since the IR proximity sensor is an active sensor we have interference between different robot’s IR sensors if they attempt to take a proximity measurement at the same time.

Time synchronization is, therefore, a critical piece of coordination for any multi-robot system. Many different methods of distributed time synchronization are used nowadays. In this testbed we use a time synchronization protocol that first lines up the clock ticks of all Mica2 motes, and then labels these clock ticks with the same network time. Eventually, all motes in the wireless network will agree upon the time of the first initialized one, as shown in the figure 5.11 and 5.12. The explanation of the two figures is provided as the following. To line up clock ticks we use a modified Mica2 radio stack provided by UCLA, which allows us to timestamp the packages just before they are transmitted and just after they are received. This fine-grained clock technique can achieve highly precise synchronization level. We realized a network-wide time synchronization with an error of \(3 - 5 \mu \text{sec}\). It is worth mentioning that we modified the TinyOS components HPLClock and TimerM service. We use the second TCNT3 counter, rather than TCNT0, to avoid conflicts with timers used by the UART stack and radio stack.

A Mica2 mote clock is an electronic device that counts oscillations in a 7.37 MHz quartz crystal. The timer counts the oscillations of the crystal, which is associated with a counter register and a holding register. For each oscillation in
Figure 5.11. Before time synchronization illustrations

Figure 5.12. After time synchronization illustrations
the crystal, the counter is decremented by one. When the counter becomes zero, an interrupt is generated and the counter is reloaded from the holding register. The interrupt is referred to as a clock tick. The first action of synchronization is to line up these clock ticks across the network. After line-up, all ticks should fire simultaneously.

The slightly different frequencies of quartz crystals cause the tick firings to gradually diverge from each other. So the network needs to synchronize time periodically. The second action of a synchronization protocol is to label the clock ticks on each mote with the same global time. This is done by having all motes periodically broadcast their network times. If the received time in an information message is earlier than the receiving mote’s time, the mote labels its clock tick with the received time. This sets up a monotone increasing sequence of update time that are guaranteed to converge to a single network time that in common to all motes. Finally all mote clocks agreed upon the earliest network time and are synchronized. Re-synchronization is mandated by periodically broadcasting the local time of each mote.

Figures 5.11 and 5.12 illustrates the synchronization routine in our testbed, in which the x axis stands for time. The clock tick firing is drawn as a black arrow. In the figure 5.11, the clock ticks (black arrows) are not lined up with the dashed lines, therefore the ticks do not fire at the same time. Furthermore, the network time labeled on the three different motes are different. After the synchronization algorithm has converged, we see the illustration, in the figure 5.12, where all clock ticks (black arrows) are lined up with the dashed lines and the network time for error at the same dashed line are equal.

Graphic User Interface - The Monitoring System
The MICA-KoalaBot testbed monitoring system tracks robots’ positions and states through using video cameras mounted in the lab ceiling. A Java GUI program takes the feeds of these cameras. The program also monitors and processes the radio messages transmitted between robots. This Java GUI can also issue commands to the Planner component thereby allowing the user to directly control the robots. But we did not use it to control robots in our algorithm and its primary goal is to monitor how the system behaves.

Camera System is consisted of four Panasonic CCD cameras mounted in the ceiling of our lab and connected back to a PC. Circular target patterns are put on the robots, so the PC can easily recognize the robot’s position and orientation in a global coordinate. This system allows us to track the robots’ movements. Figure 5.13 shows the image frames captured by one camera. The two white circles in different size are the visual cues to indicate the head and tail of individual robots. Then robots’ positions and orientations are computed based on the specific visual cues, and transformed from each camera’s pixel coordinate to the global reference.
coordinate.

The Java graphical user interface running on the monitoring PC is called *Console*. This GUI is shown in the figure 5.14. The GUI displays the calibration and identification packets that communicate across the wireless network. The data contained in these packets allow the GUI to identify each robot’s position and orientation. As shown in this figure, the dark square represents the robot’s position measured by the cameras, and the light square represents the position picked from its radio signal. The line represents the front of the robot, and there-
fore shown its orientation. The data in the communication packet also contain the proximity light sensor’s measurements, which are shown as small dots around the robot. The further a dot is from the robot, the further an obstacle will be detected by a sensor. When the dots are close to the center of a light square, the robot is close to an obstacle. The robot estimated position is laid on top of the camera’s measurement. The GUI also provides a set of commands that can be used to remotely command the vehicle to move, turn, move to its home position, and reset its local position state. This is convenient for debugging purposes.

5.4.2 Pioneer Robot Swarm

A unique feature of MICA-KoalaBot testbed is that its on-board computation unit is a Berkeley Mica2 mote rather than a laptop computer. Unlike a laptop computer, Mica2 has relatively much less memories and power supply, and therefore limited computation ability. Based on the facts, complicated coordination control algorithm is not suited-well in this testbed. In order to support comparably sophisticated cooperation among the robots, the Pioneer Robot testbed is developed.

This testbed uses the ActivMedia Pioneer 3 robot, as shown in the figure 5.15, to study methods for controlling swarms of robotic vehicles. The Pioneer 3 robot uses acoustic proximity (sonar) sensors to detect obstacles around it. It uses gyro-corrected wheel encoders to determine its local position and orientation. It is controlled through an on-board embedded Linux PC (TS-5600 EPC) that communicates to the Internet over an 802.11 wireless LAN card. Low-level robot motion control is programmed using a set of C++ classes developed by ActivMedia. The use of these classes greatly simplifies the job of developing higher-level supervisory
control programs for robot swarms. The vehicles are currently controlled over the Internet using sockets. The vehicles are treated as servers and a program running on it as a client. Note that since the communication between the client and the server is implemented using sockets, the client program can run either on the robot or a remote computer. A TCL/TK client allows a user to issue movement commands to the robot while automatically displaying the vehicle’s current position, orientation, and proximity sensor data in a graphical window.

One of the reasons we have adopted the Player/Stage project is because it supports these ActiveMedia robots. The Player client program can control any of ActiveMedia robots via the same standard interfaces, with little changes required to move from our multi-robot simulator to this testbed. The only change is to replace our virtual communication device with the real wireless card, and modifies the corresponding drivers.
CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Summary of Main Contributions

In this work, we are interested in distributed control of cooperative multiple-agent system over ad hoc wireless networks. The motivations beneath the research come from the desire to understand the impact of limited communication resources on the performance of multi-agent collaboration. Estimation techniques are naturally employed for cutting down the demand on the communication resources. Unfortunately, the techniques may degrade the system performance thanks to the estimation error. Furthermore, the efficient application of estimation techniques is closely related to how to design the network topology. Given these inspirations, we studied the coordinated systems based-on estimation over ad hoc wireless network from several perspectives, and provide fundamental and insightful understanding. Specifically, this work addressed three important and related issues: communication logic, swarm cohesion under consensus, and convergence rate of consensus filtering under network throughput limitations. The in-depth results for these issues serve as important and comprehensive guidelines for designing a practical distributed control system.

We dealt with the optimal communication logic problem in Chapter 2. The communication logic is the fashion how each agent broadcasts its state information to the neighbors. To reduce the demanding communications, an agent can
also estimate the neighbors’ states between broadcasts by using its neighbor’s dynamical model. However, the estimation errors are introduced into the system and degrade the system performance. Therefore, an optimal communication logic is necessary to balance between the communication cost and estimation error.

Both open-loop and close-loop communication logic are examined in this chapter. The broadcast decision of open-loop logic solely depends on the last broadcast time, regardless of the current estimate error. Consequently, we discovered that the optimal open-loop logic is to periodically transmit the agent’s state to its neighbors. The logic can be easily implemented, and is appropriate in the situations where the estimation error is not available instantaneously or accurately. On the other hand, the broadcast decision of close-loop logic relies on the current error, in the way that a broadcast is triggered once the estimation error is above a certain threshold. Our original contribution is to present the performance analysis of closed-loop logic in close-form. Not surprisingly, close-loop logic performs better than the open-loop logic thanks to adaptive broadcast decision based on instant estimation errors at a higher implementation price.

Next, the cohesive swarming under consensus filter is considered in Chapter 3. The interconnection structure of a swarm with a consensus filter is applied to study the swarm stability and the level of the consensus. In this structure, the consensus filter is used to estimate the local neighborhood center and govern the agent movement towards a known target. The swarm dynamics employs the short-range repulsion and long-range attraction interaction to guarantee the cohesiveness of swarm while to minimize the likelihood of agent collision.

The primary question addressed in this chapter concerns the cohesiveness of swarm under consensus and the level of consensus achieved. We derived the uni-
form ultimate bounds of the swarm size, and exploited the special consensus level through Lyapunov-based methodologies. The bounds are expressed as a function of the attraction/repulsion strength, number of network agents, and communication network connectivity. One promising approach of attaining the perfect consensus is to introduce the integral action into the consensus filter. We reveal that if the communication graph is regular, the swarm reaches perfect consensus with integral action.

The study of convergence rate for distributed consensus filters subject to the communication throughput limitation is investigated in Chapter 4. Concerning the situation of the swarm under consensus model, consensus filtering provides a useful way of distributed estimation. However, the limited communication throughput constraint leads to the message exchange delay in ways that adversely effect the consensus filter stability.

In this chapter, we concentrated on exploring the relation between network topology, message delay and convergence rate theoretically. We presented two consensus filter schemes, synchronous and asynchronous consensus, associated to different communication protocols. As the name suggest, synchronous consensus obeys the principle wherein individual agents regulate their states only after receiving all neighbors’ message. While, in asynchronous consensus manner, each agent updates its state if and only if receives any delayed message from any neighbors. We employ a time-slotted wireless communication model in frequency division multiple access to consider the convergence rate of two types of consensus filters, respectively. The work provides the fundamental guidance for building appropriate communication connectivity for achieving the fast consensus convergence limited to the throughput. We showed that the asynchronous consensus assures
the system stability regardless of the network connectivity, while relative dense connectivity possibly causes the synchronous consensus system unstable. Conversely, the synchronous scheme is able to achieve the same $\epsilon$-consensus in less iterations, compared to asynchronous scheme.

In order to test the proposed coordinated control algorithm in real multi-robot system, we developed a software multi-robot simulator and hardware testbed of real robots. The simulation software is built based on the Player/Stage project, which enable us to experiment our studied control algorithm without accessing to the real hardware and environment. More importantly, in the simulator, each simulated robot is treated as an independent entity and runs in an independent process. There is no central based to schedule the robots’ cooperation order, which is unrealizable with Matlab simulation. Therefore, the simulator is capable to simulate the distributed coordinated network system more accurately. The simulator hence, is a very useful tool for rapid development of controllers. At the same time, we also introduced our laboratory robotic vehicle testbed built to measure the distributed system cooperation performance over ad hoc wireless network. On this testbed, we programmed the robots and tested our algorithms and verified the theories we have developed for applications with similar environmental assumptions.

6.2 Future Work

This work is only a start towards unveiling the complicated problems of distributed control under estimation in networked multiple agent systems. We made several assumptions in this work to make the problems tractable. Removing some of these assumptions will be very interesting and challenging for future research.
We have assumed an average delay model in this work to study the convergence rate of the consensus filters. In general, the information flow in a wireless network is considerably more complicated. Various delay models of the information flow will lead to many open research topics.

We have, so far, assumed that each agent knows some common objective in the beginning. For most applications of a multi-agent system, this may not be always practical. An example is the pollution consistency detection, in which the source of pollution is not known \textit{a priori} and has to be determined by measurements of each agent. Therefore, the agents in the group have to reach global agreements towards the sources of the pollution with the highest consistency, and to use it to guide the individual agents’ directions.

Also, porting the simulation to the newly acquired ActiveMedia robots will give us immediate confirmation on the simulation results. Algorithms can be further modified and tuned based on this testbed. All these will further solidify the foundations for research with deeper technical depth and greater application potential.
Proof: Lemma 3.8.1: Any eigenvalue $\lambda$ of $\Phi$ must satisfy the characteristic equation $\chi(\Phi) = 0$ so that

$$0 = \chi(\Phi) = \det \begin{bmatrix} \lambda I - A & -KI \\ L & \lambda I \end{bmatrix}$$

$$= \det (\lambda I - A) \det (\lambda I + KL(\lambda I - A)^{-1})$$

$$= \det (\lambda(\lambda I - A) + KL) \quad (7.1)$$

The rank of Laplacian matrix $L(G)$ is $N - 1$ when the graph $G$ is connected, so that $\det(KL) = 0$. This implies that $\Phi$ has at least one zero eigenvalue.

Considering the relationship between matrices $A$ and $L$ in (3.9), we define a new matrix

$$\tilde{A} = \frac{1}{N} 11^T - 2I_N - L(G) \quad (7.2)$$

and $A < \tilde{A}$. Substituting $A$ in $\Phi$ by $\tilde{A}$ gives us $\tilde{\Phi}$,

$$\tilde{\Phi} = \begin{bmatrix} \tilde{A} & KI \\ -L & 0 \end{bmatrix}.$$
and \( \Phi \leq \tilde{\Phi} \).

Since \( L \) is a symmetric matrix, it can be eigendecomposed as \( L = PSP^H \), where \( P \) is a unitary eigenvector matrix, and \( S \) is the diagonal eigenvalue matrix with diagonal elements \( s_i \). Without loss of generality, we suppose \( s_N = 0 \). Here and in what follows, we denote \((\cdot)^H\) as the conjugate transpose of a matrix, and \((\cdot)^*\) as the conjugate of a number.

In terms of equation (7.2), we have

\[
P^H \tilde{A} P = -S - 2I + \frac{1}{N} v v^H
\]

where \( v = P^H 1 = [0, 0, \cdots, 0, -\sqrt{N}]^T \), corresponding to \( s_N = 0 \), and the only non-zero element of the matrix \( v v^T \) is \( \frac{1}{N} [v v^T]_{N,N} = 1 \).

Define a unitary matrix \( Q \) as

\[
Q = \begin{bmatrix} P \\ P \end{bmatrix}
\]

then we obtain,

\[
\Phi_1 = Q^H \Phi Q = \begin{bmatrix} -T & KI \\ -S & 0 \end{bmatrix}
\]

which has the same eigenvalues as \( \Phi \), and

\[
\tilde{\Phi}_1 = Q^H \tilde{\Phi} Q = \begin{bmatrix} -\tilde{T} & KI \\ -S & 0 \end{bmatrix}
\]
where

$$\text{diag}(\tilde{T}) = t_i = \begin{cases} 
s_i + 2 & 1 \leq i \leq N - 1 \\
1 & i = N
\end{cases}$$

and $t_i > 0$ for $1 \leq i \leq N$.

Let $u$ be a normalized eigenvector of $\Phi_1$ associated with a non-zero eigenvalue $\lambda$, i.e., $\Phi_1 u = \lambda u$. Suppose $u = [x_1 \cdots x_N \; y_1 \cdots y_N]^T$, some algebra gives

$$-s_i x_i = \lambda y_i$$

(7.3)

Since $\Phi \preceq \tilde{\Phi}$, the inequality $u^H(\Phi_1 - \tilde{\Phi}_1)u < 0$ is hold regarding the structures of matrices $\Phi$ and $\tilde{\Phi}$, or $\lambda - u^H \tilde{\Phi}_1 u < 0$. Moreover,

$$u^H \tilde{\Phi}_1 u = -\sum_{i=1}^N t_i |x_i|^2 - \sum_{i=1}^N s_i x_i y_i^* + K \sum_{i=1}^N x_i^* y_i$$

$$= -\sum_{i=1}^N t_i |x_i|^2 + \lambda \sum_{i=1}^N |y_i|^2 - K \lambda^* \sum_{i=1}^N \frac{1}{s_i} |y_i|^2$$

$$= -C + c_1 \lambda - c_2 \lambda^*$$

where $C > 0$, $0 < c_1 < \|u\| = 1$, and $c_2 > 0$ based on the equation (7.3). Therefore,

$$\lambda - u^H \tilde{\Phi}_1 u$$

$$= (1 - c_1)\lambda + c_2 \lambda^* + C$$

$$= (1 - c_1 + c_2)\text{Re}(\lambda) + (1 - c_1 - c_2)\text{Im}(\lambda) + C < 0$$

which implies $1 - c_1 - c_2 = 0$. Therefore, $1 - c_1 + c_2 = 2c_2 > 0$. We get
Re(\(\lambda\)) < \(-\frac{C}{2c_2}\) < 0. In other word, the non-zero eigenvalues of \(\Phi\) have a negative real part.

Finally, we show that \(\Phi\) has at most one zero eigenvalue. We consider matrix \(\Phi^T\),

\[
\Phi^T = \begin{bmatrix}
A & -L \\
KI & 0
\end{bmatrix}
\]

and let \(\lambda_j\) be the zero eigenvalue of \(\Phi^T\) with associated eigenvector \(u = \begin{bmatrix} u_1^T & u_2^T \end{bmatrix}^T \in \mathbb{R}^{2N}\) in which \(u_1, u_2 \in \mathbb{R}^N\). then

\[
0 = Au_1 - Lu_2 \tag{7.4}
\]
\[
0 = -Ku_1 \tag{7.5}
\]

Equation 7.5 implies that \(u_1 = 0\) and \(u_2\) is the eigenvector resulting in \(Lu_2 = 0\). \(u_2\) will be any vector belonging to the null-space of matrix \(L\). Because \(L\) is the Laplacian of graph \(G\), the dimension of \(L\)'s null space is exactly one, thereby completing the proof.

**Proof:** Lemma 3.8.2: It is straightforward that

\[
U^{-1} \begin{bmatrix}
A & KI \\
-L & 0
\end{bmatrix} \otimes I_n = \Lambda U^{-1} = \begin{bmatrix} x \\
\vdots \\
x \\
0
\end{bmatrix} \in \mathbb{R}^{2Nn \times 2Nn}
\]

where \(x \in \mathbb{R}^{2Nn \times n}\) is any complex matrix satisfying dimension requirement, and
\(0 \in \mathbb{R}^{2N_n \times n}\). Moreover,

\[
\begin{bmatrix}
A & KI \\
-L & 0
\end{bmatrix} \otimes I_n U = UA = \begin{bmatrix} x & \cdots & x & 0 \end{bmatrix} \in \mathbb{R}^{2N_n \times 2N_n}
\]

where matrices \(x\) and \(0 \in \mathbb{R}^{n \times 2N_n}\).

1. For the eigenvalue of \(\lambda_{2N} = 0\), we have,

\[
\begin{bmatrix}
A & KI \\
-L & 0
\end{bmatrix} \begin{bmatrix} u_{2N} \\
u_{2N} \end{bmatrix} = \begin{bmatrix} 0 \\
0 \end{bmatrix} \in \mathbb{R}^{2N}
\]

Hence, \(u_{2N}\) is the null space vector of Laplacian matrix \(L\), which completes the proof of the first item.

2. The second item’s proof is similar to the first one. For the eigenvalue of \(\lambda_{2N} = 0\), we have,

\[
\begin{bmatrix}
V_{2N}^T & V_{2N}^T \\
-L & 0
\end{bmatrix} \begin{bmatrix} A & KI \\
-L & 0
\end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{2N}
\]

So that, \(v_{2N} = 0\) and \(\bar{v}_{2N} = u \cdot 1\).

3. For the eigenvalue of \(\lambda_i \neq 0\), we have,

\[
\begin{bmatrix}
v_i^T & \bar{v}_{i}^T \\
-L & 0
\end{bmatrix} \begin{bmatrix} A & KI \\
-L & 0
\end{bmatrix} = \lambda_i \begin{bmatrix} v_i^T & \bar{v}_{i}^T \end{bmatrix}
\]
It is easy to show,

\[ K \mathbf{v}_i^T = \lambda_i \mathbf{v}_i^T \quad i = 1, \cdots, 2N - 1 \]

4. Since \( \mathbf{U} \mathbf{U}^{-1} = \mathbf{I} \), it means the matrix block

\[
\begin{bmatrix}
\mathbf{u}_1 & \cdots & \mathbf{u}_{2N}
\end{bmatrix}
\begin{bmatrix}
\mathbf{v}_1^T \\
\vdots \\
\mathbf{v}_{2N}^T
\end{bmatrix} = 0 \in \mathbb{R}^{N \times N}
\]

It turns out,

\[
\sum_{i=1}^{2N-1} \mathbf{u}_i \mathbf{v}_i^T = -\mathbf{u}_{2N} \mathbf{v}_{2N}^T
\]

5. Because of \( \mathbf{U}^{-1} \mathbf{U} = \mathbf{I} \), we have the last element

\[
\mathbf{v}_{2N}^T \mathbf{u}_{2N} + \mathbf{v}_{2N}^T \mathbf{u}_{2N} = 1
\]

in terms of the property of \( \mathbf{v}_{2N} = \mathbf{0}^T \), the above equation is equivalent to,

\[ \mathbf{v}_{2N}^T \mathbf{u}_{2N} = 1. \]

6. It is easy to shown based on the equation,

\[
\begin{bmatrix}
\mathbf{A} & \mathbf{K} \mathbf{I} \\
-\mathbf{L} & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{u}_{2N} \\
\mathbf{u}_{2N}
\end{bmatrix} =
\begin{bmatrix}
0 \\
\vdots \\
0
\end{bmatrix} \in \mathbb{R}^{2N \times 1}
\]
Proof: **Lemma 4.4.1:** Let \( s = j\omega \) to find out the upper bound of delay \( \bar{\tau} \) such that \( s - \lambda_i(A)e^{-\bar{\tau}s} \) has a zero on the imaginary axis,

\[
\begin{align*}
\quad j\omega - e^{-j\omega\bar{\tau}}\lambda_i &= 0 \\
- j\omega - e^{j\omega\bar{\tau}}\lambda_i &= 0
\end{align*}
\]

Multiplying both sides of the above equation yields,

\[
\omega^2 - \lambda_i^2 - 2j\omega\lambda_i\cos(\omega\bar{\tau}) = 0
\]

then we obtain \( \omega = \pm \lambda_i \), and \( \cos(\omega\bar{\tau}) = 0 \). Or, \( \cos(\lambda_i\bar{\tau}) = 0 \), which implies \( \lambda_i\bar{\tau} = -(k + \frac{1}{2})\pi, k \in \mathbb{Z}^+ \) for a \( \lambda_i \) given \( \bar{\tau} \). Therefore, for the smallest delay \( \bar{\tau} \), we have

\[
\bar{\tau} = -\frac{\pi}{2 \max\{\text{eig}(A)\}} \leq \frac{\pi}{2(1 + \Delta)} \quad (7.6)
\]

The second inequality is in terms of the inequality 4.15. Hence, if \( 0 < \bar{\tau} \leq \frac{\pi}{2(1 + \Delta)} \), the roots of \( s - \lambda_i(A)e^{-\bar{\tau}s} \) locate on open LHP, and therefore the system is stable.

**Mathematical Preliminaries:** The discriminant of \( L_i(s) \) in equation (4.18) is

\[
R_i = -2(\overline{\tau}\lambda_i)^2 + 24\lambda_i\overline{\tau} + 9 \quad (7.7)
\]

and the roots \( s_1, s_2 \) of \( L_i(s) \) are,

\[
\begin{align*}
\quad s_1 &= -\frac{3 + 2\lambda_i\overline{\tau} + \sqrt{R}}{2\overline{\tau} - \overline{\tau}^2\lambda_i} \\
\quad s_2 &= -\frac{3 + 2\lambda_i\overline{\tau} - \sqrt{R}}{2\overline{\tau} - \overline{\tau}^2\lambda_i}
\end{align*}
\]
When $R_i \geq 0$, $s_1$ and $s_2$ are both real and $s_1 \leq s_2$. When $R_i < 0$, then $s_1, s_2$ are a pair of conjugate complex roots in which $\text{Re}(s_2) = -\frac{3+2\lambda_i}{2\tau-2\lambda_i}$.

**Proof:** Lemma 4.4.3

- if $R_i \geq 0$, we want to show $s_2$ monotonically decreases with $\Delta$ increasing.

First, we simplify $\sqrt{R_i}$ as,

$$\sqrt{R_i} = \sqrt{-2(\tau\lambda_i)^2 + 24\tau\lambda_i + 9} \approx 3 + 4\tau\lambda_i$$

then we have,

$$y = -\text{Re}(s_2) \approx \frac{3(\tau\lambda_i)^2 - 2\tau\lambda_i}{2\tau - 2\lambda_i}$$

$$= \frac{3\tau\lambda_i^2 - 2\lambda_i}{2 - \tau\lambda_i} = \frac{8/\tau}{2 - \tau\lambda_i} + \frac{4}{\tau} - 3\lambda_i$$

which is monotonically increasing with $\lambda_i$. Hence, we consider $\lambda_i = -(1+\Delta)$, and obtain,

$$y = \frac{3\tau(1+\Delta)^2 + 2(1+\Delta)}{2 + \tau(1+\Delta)}$$

$$= \frac{3\gamma(1+\Delta)(1+\Delta)}{\Delta^2} + \frac{2(1+\Delta)}{\Delta^2}$$

$$= \frac{3\tau(1+\Delta)^3 + 2(1+\Delta)}{2 + \frac{\gamma(1+\Delta)^3}{\Delta^3}} = \frac{u(\Delta)}{v(\Delta)}$$

where $x = \frac{\tau}{\Delta}$. To show $-\text{Re}(s_2)$ is monotonically increasing is equivalent to
showing \( u'v - v'u > 0 \). We first get

\[
\begin{align*}
    u' &= 3x \frac{(1+\Delta)^{3+\Delta}}{\Delta} (\ln \frac{1+\Delta}{\Delta} + \frac{2}{1+\Delta}) + 2 \\
    &= 3x A ((1+\Delta)^2 B + (1+\Delta)) + 2 \\
    v' &= x \frac{(1+\Delta)^{2+\Delta}}{\Delta} (\ln \frac{1+\Delta}{\Delta} + \frac{1}{1+\Delta}) \\
    &= x A (B(1+\Delta) + 1)
\end{align*}
\]

and then

\[
\begin{align*}
    u'v - v'u &= 2A(1+\Delta)(2(1+\Delta)B + 3)x + 4 > 0
\end{align*}
\]

where we let \( A = \frac{(1+\Delta)(1+\Delta)}{\Delta^2} > 0 \) and \( B = \ln \frac{1+\Delta}{\Delta} > 0 \), which completes the proof.

- For \( R_i < 0 \), the real part of the conjugate roots is

\[
Re(s_2) = -\frac{3 + 2\mu \tau}{2\tau - \tau^2 \lambda_i} = -\frac{7}{2\tau - \tau^2 \lambda_i} + \frac{2}{\tau}
\]

which decreases with \( \lambda_i \) increasing for a given \( \tau \). Hence, we consider the bound by letting \( \lambda_i = -1 - 3\Delta \). Therefore, let

\[
\begin{align*}
    y &= -Re(s_2) = \frac{u(\Delta)}{v(\Delta)} \\
    &= \frac{3 - 2\mu^2 \frac{(1+\Delta)(1+\Delta)(3\Delta+1)}{\Delta^2}}{2\mu \frac{(1+\Delta)(1+\Delta)}{\Delta^2} + (2\mu) \frac{(1+\Delta)(2+2\Delta)}{\Delta^2} (3\Delta + 1)} \\
    &= \frac{3 - 2 \mu \frac{(1+\Delta)(1+\Delta)}{\Delta^2} (3\Delta + 1)}{2 \mu \frac{(1+\Delta)(1+\Delta)}{\Delta^2} + 2 \mu \frac{(1+\Delta)(2+2\Delta)}{\Delta^2} (3\Delta + 1)}
\end{align*}
\]

160
where \( x = \frac{\gamma}{Q} \) is not dependent on \( \Delta \). Let \( A = \frac{(1+\Delta)(1+\Delta)}{\Delta^2} \) and \( B = \ln \frac{1+\Delta}{\Delta} \), then

\[
\begin{align*}
  u(\Delta) &= 3 - 2xA(1 + 3\Delta) \\
v(\Delta) &= 2xA + x^2A^2(1 + 3\Delta)
\end{align*}
\]

Taking derivative of \( y \) related to \( \Delta \) gives \( \frac{dy}{d\Delta} = \frac{u'v - v'u}{v^2} \) where

\[
\begin{align*}
  u' &= -2xA(B(1 + 3\Delta) + 3) \\
v' &= 2xAB + x^2A^2(B(1 + 3\Delta) + 3)
\end{align*}
\]

Therefore, we obtain,

\[
u'v - v'u = 3A^2(3 - B(1 + 3\Delta))x^2 - 6ABx\]

For \( \Delta > 1, 3 - B(1 + 3\Delta) < 0 \). Thus \( \frac{dy}{d\Delta} < 0 \), since \( x > 0 \), or \( y \) decreases with \( \Delta \) increasing. Therefore, the real part of the conjugate roots is monotonically increasing.

\[\blacksquare\]

**Proof: Lemma 4.4.4**

- Similar to the proof in the first part of lemma 4.4.3, we can show \( s_2 \) increases with increasing \( \lambda_i \). Hence, it is straightforward to prove \( s_\ell(\lambda_N) \leq \cdots \leq s_\ell(\lambda_1) \). Moreover, the numerator of the \( Re(s_2) \) is

\[
\begin{align*}
  -3 - 2\lambda_i\tau + \sqrt{-2(\tau\lambda_i)^2 + 24\lambda_i\tau} + 9 \\
  = -3 - 2\lambda_i\tau + \sqrt{(3 + 2\lambda_i\tau)^2 + 6\tau\lambda_i(2 - 2\lambda_i)} < 0
\end{align*}
\]
since $\lambda_i < 0$ and $0 < \overline{\tau} \leq \frac{\pi}{2\lambda_N}$.

- The proof is similar to the first item’s proof.

- In terms of the lemma 4.4.3, the optimal degree $\Delta^*$ associated with a given $\lambda_i$ is obtained by solving that the discriminant of $L_i(s)$ equals to zero, such that $R_i = 0$. This leads to

$$\lambda_i \overline{\tau}(\Delta) = -0.364$$  \hspace{1cm} (7.8)

for all $\lambda_i$ associated with different communication degrees. $\overline{\tau}$ is determined by the degree $\Delta$, which increases with increasing $\Delta$. Therefore, in equation (7.8), an increase in $\Delta$ leads to an increase in $\overline{\tau}$ and a decrease in $\lambda_i$. 


45. L. Kleinrock and J. Silverster. Optimum transmission radii for packet radio networks or why six is a magic number. In IEEE National Telecommunication Conference, pages 4.3.1–4.3.6.


