Adaptive Communication for Mobile Multi-Robot Systems

by

Ryan J. Marcotte

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2019

Doctoral Committee:

Associate Professor Edwin Olson, Chair
Professor Ella Atkins
Assistant Professor Geoffrey Hollinger, Oregon State University
Associate Professor Odest Chadwicke Jenkins

Ryan J. Marcotte

ryanjmar@umich.edu

ORCID iD: 0000-0002-8413-107X

Acknowledgments

Thank you to my advisor Ed Olson for giving me many opportunities and for guiding me along the way. Thank you to my other committee members (Chad Jenkins, Ella Atkins, and Geoff Hollinger) for their valuable feedback on this material.

Thank you to all those who have prepared me for this work, in particular all of the amazing teachers I have had throughout my life. Thank you to those who have lent their time, energy, and expertise to mentor me, especially Walter Voit and Nicholas Gans who gave me the earliest opportunities of my research career. Thank you to all of my labmates and collaborators who have contributed immensely to this work and to my development as a researcher. In particular, thank you to Xipeng Wang and Dhanvin Mehta, who contributed to part of this dissertation's material.

Most of all, thank you to my family, without whose love and support none of this would have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

ACK acknowledgment

AEC Adaptive Erasure Coding

HBAEC High Bandwidth Adaptive Erasure Coding

ARQ automatic repeat request

BEC binary erasure channel

CDF cumulative distribution function

ConTaCT Communication for Time-Critical Collaborative Tasks

CRC cyclic redundancy check

CSAR combinatorial successive accept-reject

Dec-MDP decentralized Markov decision process

Dec-POMDP decentralized partially observable Markov decision process

Dec-MCTS decentralized Monte Carlo tree search

DIF dynamic information flow

EKF extended Kalman filter

EWMA exponentially weighted moving average

FEC forward error correction

HTTP Hypertext Transfer Protocol

IID independent and identically distributed

KDE kernel density estimation

LIDAR Light Detection and Ranging

MANET mobile ad-hoc network

MDS maximum distance separable

MTU maximum transmission unit

OCBC Optimizing Communication under Bandwidth Constraints

PDF probability density function

PEC packet erasure channel

PHY physical layer

PMF probability mass function

PRR packet reception ratio

QoS Quality of Service

QUIC Quick UDP Internet Connections

RSSI received signal strength indicator

SLAM simultaneous localization and mapping

SNR signal-to-noise ratio

SMA simple moving average

TCP Transmission Control Protocol

ToM Theory of Mind

UDP User Datagram Protocol

UDT UDP-based Data Transfer Protocol

ABSTRACT

Mobile multi-robot systems can be immensely powerful, serving as force multipliers for human operators in search-and-rescue operations, urban reconnaissance missions, and more. Key to fulfilling this potential is robust communication, which allows robots to share sensor data or inform others of their intentions. However, wireless communication is often unreliable for mobile multi-robot systems, exhibiting losses, delays, and outages as robots move through their environment. Furthermore, the wireless communication spectrum is a shared resource, and multi-robot systems must determine how to use its limited bandwidth in accomplishing their missions.

This dissertation addresses the challenges of inter-robot communication in two thrusts. In the first thrust, we improve the reliability of such communication through the application of a technique we call Adaptive Erasure Coding (AEC). Erasure codes enable recovery from packet loss through the use of redundancy. Conditions in a mobile robotic network are continually changing, so AEC varies the amount of redundancy applied to achieve a probabilistic delivery guarantee.

In the second thrust, we describe a mechanism by which robots can make communication decisions by considering the expected effect of a proposed communication action on team performance. We call this algorithm Optimizing Communication under Bandwidth Constraints (OCBC). Given a finite amount of available bandwidth, OCBC optimizes the contents of a message to respect the bandwidth constraint.

CHAPTER 1

Introduction

Communication is fundamental to multi-robot systems. To achieve a shared goal, robots coordinate their actions by communicating their future intentions. Even in less tightly coupled tasks, a robot can share acquired information with its teammates, helping them in their respective missions. When multi-robot systems serve as the "eyes and ears" of human operators, communication transports the data from the field back to a control center.

Communication in multi-robot systems is also highly challenging. To be useful in applications like search-and-rescue or urban reconnaissance, such systems must be able to operate in harsh environments that lack fixed network infrastructure (e.g. underground tunnels, collapsed buildings, etc.). The robots themselves form a mobile ad-hoc network (MANET), transmitting messages wirelessly and routing them to teammates. When robots move, the topology of the network changes and its links vary in quality. Robots may move in and out of communication range, leading to significant packet loss and fluctuations in throughput. Furthermore, when multiple robots share the same communication medium (e.g. a WiFi channel), the limited bandwidth constrains the amount of data each robot can transmit.

To better understand the challenges involved in mobile robot communication, consider Figure 1.1, which illustrates instantaneous packet loss rates along a mobile robot's trajectory through an office building. The mobile robot communicates with a stationary node represented by the gold star, and darker portions of the trajectory indicate periods of higher packet loss. As the robot moves through the building, it experiences significant and varied rates of packet loss. This packet loss in turn results in a reduced amount of bandwidth available for communication.

We argue that to overcome these challenges, a mobile multi-robot system must use adaptive communication mechanisms. Specifically, the robots' network protocols must adapt to changing network conditions as well as the Quality of Service (QoS) needs of dif-

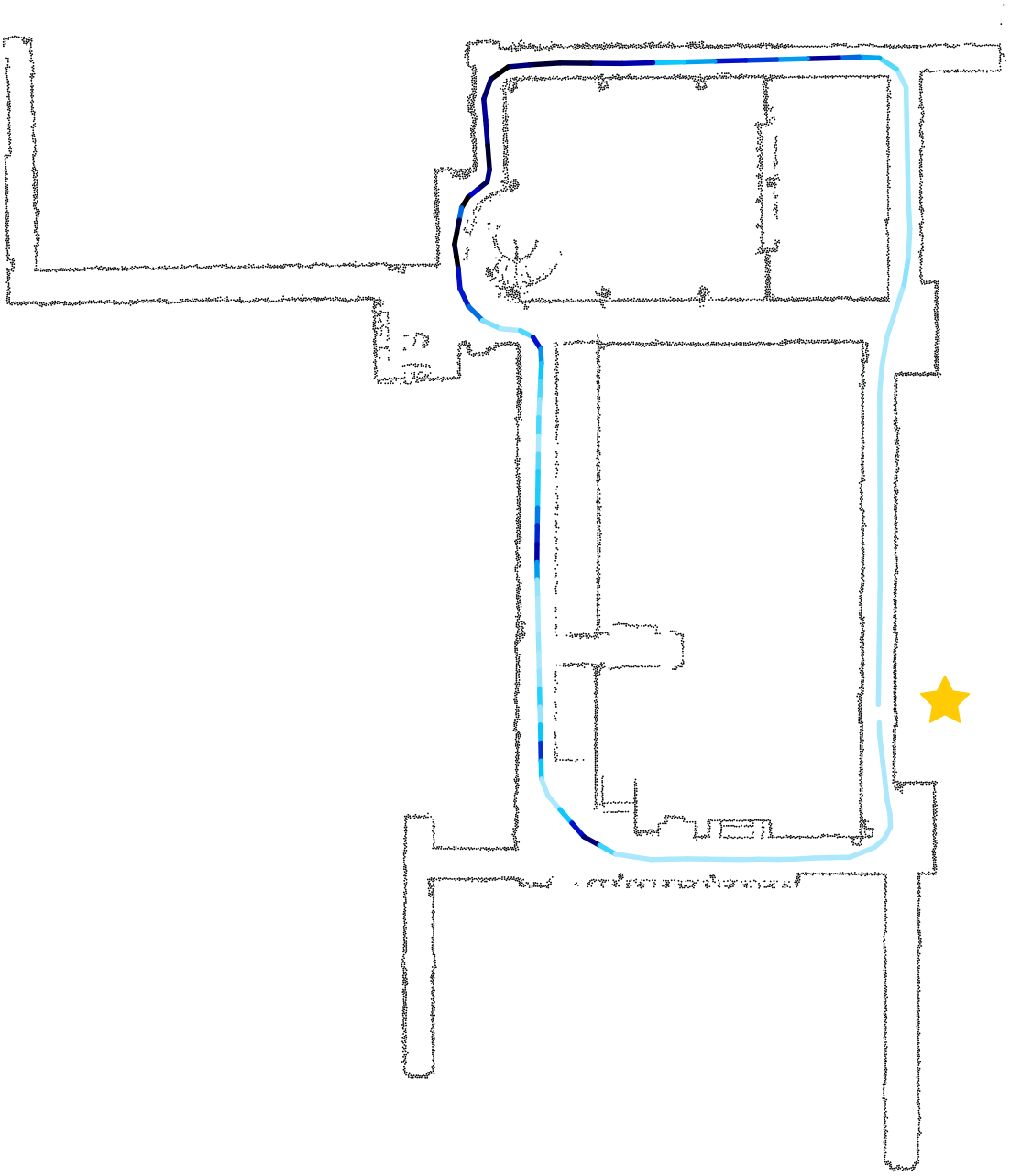

Figure 1.1: Visualization of instantaneous packet loss rates measured along a mobile robot's trajectory through an office building, with communication occurring between the robot and a stationary node (gold star). Darker colors indicate higher packet loss. We observe that the mobile robot experiences significant packet loss events as it travels through its environment and that the instantaneous packet loss rate can change rapidly.

ferent traffic, and the robots' planning mechanisms must adapt to the available bandwidth by reasoning about what information to transmit.

This dissertation aims to provide these types of adaptive communication mechanisms for mobile multi-robot systems. In Chapter 3, we identify and exploit a possible characteristic of network traffic in multi-robot systems to improve packet delivery performance. Namely, we observe that when network traffic is tolerant to latency, we can apply a forward error correction (FEC) scheme that we call Adaptive Erasure Coding (AEC) to reduce the effective packet loss rate with minimal overhead. The key idea of this method is that by applying erasure coding across groups of individual packets, we can use packets successfully received from the group to recover packets that have been lost. Previous erasure coding methods applied to traditional networks (e.g. Google's QUIC [74]) use a constant encoding strength, but this is unsuitable for the signficant and highly variable packet loss rates experienced in mobile robotic networks. AEC adapts its encoding strength, transmitting the minimum amount of redundancy needed to satisfy an upper bound on the probability of unrecoverable packet loss. To enable this calculation, we empirically estimate the instantaneous packet loss rate distribution by filtering historical packet loss measurements. We demonstrate that the AEC system can achieve a target packet loss rate across a range of link qualities and can successfully exchange latency tolerance for improvements in packet delivery performance. We also show that the adaptive system provides more robust performance with lower overhead than a traditional static system.

In Chapter 4, we extend the ideas of Chapter 3 and incorporate them into a system capable of transporting bandwidth-intensive traffic (e.g. images, raw sensor data, etc.) across a mobile robotic network. Efficiently encoding and decoding large quantities of data requires the use of a different class of erasure codes known as fountain codes. Fountain codes have probabilistic decoding characteristics, so we generalize the encoding strength calculation of Chapter 3 to apply to this type of erasure code. We also provide a technique to estimate instantaneous packet loss rate with kernel density estimation and show that this method yields higher-quality estimates than the filtering technique of Chapter 3. We describe some of the challenges of realistic and repeatable performance evaluation in these types of systems and provide a mechanism that facilitates high-quality evaluation. Finally, we demonstrate that the proposed system, High Bandwidth Adaptive Erasure Coding (HBAEC), outperforms TCP at continually delivering images across a mobile robotic network.

In Chapter 5, we consider the problem of deciding what to communicate given a constraint on available bandwidth. Making such communication decisions is prohibitively

complex for general multi-agent decision-making models such as a decentralized partially observable Markov decision process (Dec-POMDP) or decentralized Markov decision process (Dec-MDP). We therefore study the problem using a specialized variant of these models suitable for exploration-style robotics tasks. This allows us to evaluate potential communication actions directly in terms of their expected effect on reward, unlike many previous approaches that require a more heuristic approach. Given a set of possible observations to communicate, we select which ones to share using a bandit-based combinatorial optimization technique. We show that the resulting method, which we call Optimizing Communication under Bandwidth Constraints (OCBC), can achieve better task performance on an example task with up to an order of magnitude less communication compared to a previous state-of-the-art method.

The contributions of this dissertation are as follows:

- In Chapter 3, we describe a mechanism by which robot systems can exchange tolerance to latency for improvements in the probability of successful packet delivery. We introduce an algorithm that determines the minimum amount of redundancy to transmit to keep the probability of unrecoverable packet loss sufficiently small, an algorithm that depends on a technique to estimate the distribution of possible instantaneous packet loss rates from historical packet loss measurements.

- In Chapter 4, we generalize the optimization technique of Chapter 3 for use with erasure codes with probabilistic decoding characteristics, which are themselves suited to encoding large quantities of data. We present a technique that uses kernel density estimation to estimate the distribution of instantaneous packet loss based on historical packet loss measurements. We describe a repeatable evaluation that uses packet traces collected from a real-world robotic network to drive simulations.

- In Chapter 5, we provide a mechanism for making communication decisions under bandwidth constraints in collaborative, multi-robot settings. This mechanism uses forward simulations to evaluate potential communication actions directly in terms of their expected effect on reward. We describe a bandit-based method that optimizes message content based on the results of these forward simulations.

CHAPTER 2

Background

In this chapter, we provide the background needed to understand the proposed methods and their relation to existing work. Section 2.1 highlights relevant work relating to communication reliability in multi-robot systems, giving perspective on how other researchers have thought about the problems we tackle in Chapters 3 and 4. Section 2.2 discusses the communication decision-making literature, providing context for the method we propose in Chapter 5. Finally, Section 2.3 serves as a brief primer on erasure codes, the fundamental technology underlying our proposed methods in Chapters 3 and 4. Erasure codes lie outside the standard technological repertoire of roboticists, so we focus on their key properties and provide intuition for how they function.

2.1 Communication Reliability in Multi-Robot Systems

Given the importance of robust communication to multi-robot systems, previous researchers have considered many different approaches to improving communication reliability. In this section, we briefly review the main areas of this research.

Spatial Estimation of Communication Reliability

Multi-robot systems often rely on their communication mechanisms to operate. As such, it may be desirable to make spatial predictions of communication performance. Many approaches exist to generate maps of communication quality.

Gonzalez-Ruiz et al. [31] take an entirely analytical approach, building up a model from existing literature on wireless communication. They model communication between two wireless nodes as a multi-scale dynamical system with three major dynamics: multi-path,

shadowing, and path loss. Gonzalez-Ruiz et al. validate the model by taking an series of channel measurements and comparing the empirical results to the analytical model. Mostofi et al. [60] then provide a method that uses the same model to make spatial predictions of communication quality. Their method involves estimating the underlying parameters of the channel, which they then apply to estimate link quality at any location. Yan and Mostofi [99] applied a similar method to the problem of predicting communication performance along a robot's future trajectory. Given that most such models are developed for IEEE 802.11 channels, Fink et al. [24] and Fink [23] evaluated the applicability of such models to low-power IEEE 802.15.4 and found promising results. All of these analytical approaches yield a generative model, but that model is highly complex and sensitive to error.

As an alternative to the analytical approach, researchers have proposed data-driven (machine learning) methods. Such methods measure factors such as received signal strength indicator (RSSI), local traffic load, or packet reception ratio (PRR) and use them as features in a classifier that predicts link quality at a given location. Flushing et al. [25] learn such a spatial map offline, then incorporate the predictions into a path planner. Given a maximum allowable travel distance for a robot to take to a destination, their method optimizes the trajectory for expected communication quality. Lowrance et al. [52] build a classifier based on fuzzy logic, which allows a human to incorporate domain knowledge. Kudelski et al. [50] propose a two-stage approach: first, optimize robot trajectories to collect a maximally informative set of samples, then learn a classifier from those samples and re-deploy the system to complete its task. Strom and Olson [85] incorporate other forms of sensor data (e.g. LIDAR) in combination with network statistics to inform a Gaussian Process regression technique.

Guaranteeing Connectivity through Coordination

A significant amount of effort has been put toward algorithms that maintain network connectivity while accomplishing some task. The most interesting – and challenging – formulation of this problem is to maintain connectivity in a decentralized or distributed fashion. For example, Zavlanos and Pappas [100] present a distributed controller that uses algebraic graph theory to control individual robot motions based on estimates of their local topology. Michael et al. [58] validate this method and demonstrate its successful deployment to a real-world distributed multi-robot system. Rooker and Birk [73] propose a connectivity-preserving exploration algorithm that weighs the benefits of exploring unknown territory

with the goal of keeping communication intact. Tardioli et al. [89] present a cooperative motion control algorithm as well as a network-aware task allocation method, all aimed at enforcing network connectivity. Finally, Hsieh et al. [42] incorporate a learned spatial map of estimated signal strength into a reactive, decentralized controller to maintain communication links.

A related problem is the deployment of a team of robots, a fixed subset of which serves as communication relays or routers for the rest of the team. In such a system, the relay robots' only task is ensuring network connectivity, so they do not participate in the overall task of the rest of the team. Much of this work builds from ideas of static relay placement [21], generalizing to the case of mobile relays. Tekdas et al. [90] consider the problem of positioning a set of robotic routers so that a single mobile client node maintains connectivity to a base station. They present optimal motion planning algorithms for two cases, corresponding to a known client motion model and an adversarial one. Gil [28] and Gil et al. [29] propose an adaptive method for positioning mobile routing nodes according to network conditions. Specifically, a routing node measures network connectivity along each spatial direction, giving it insight into how to move to best provide connectivity to client nodes.

Delay and Disruption Tolerant Networking

Aside from teams of mobile robots, other systems face similarly austere network conditions. For example, communicating with space vehicles on interplanetary missions requires tolerating lengthy delays as signals propagate across the immense distances between nodes. Communicating with a probe on Mars involves delays of about 5 to 20 minutes, orders of magnitude longer than delays of the general Internet, rendering standard Internet protocols like TCP incapable of operating [10]. Furthermore, communication between nodes might be only periodically available, such as when a node is positioned on the far side of a planet from an Earth-based transmitter.

Problems like this have led to much research in the area of delay and disruption tolerant networking. One focus has been on designing congestion control mechanisms capable of operating under significant and variable delays [82]. Other work has developed routing protocols that account for limited, periodic connectivity [84].

A particularly relevant example is the erasure coding-based routing protocol of Wang et al. [93]. They argue that when contacts (periods of connectivity) occur infrequently

and unpredictably, traditional automatic repeat request (ARQ) mechanisms will fail due to long delays. Instead, they apply erasure coding to increase the likelihood of successful transmission during the contact. They show that the erasure coding approach outperforms simple data replication. Unlike the methods we propose in this thesis, however, their system applies only a fixed amount of redundancy to each transmission. Another paper worth highlighting here is that of Grøtli and Johansen [33], which outlines a joint motion- and communication-planning method for unmanned aerial vehicles, whose networks may exhibit delays and disruptions. They consider delay to be a free parameter controlled by the user and show in their evaluation that tolerating some delay improves task performance, an insight similar to our own in Chapter 3. In their case, increasing the specified latency tolerance leads to more efficient motion planning as it enables the robots to store and carry messages for later forwarding. Without this capability, the robots would need to make more frequent contacts to exchange information instantaneously. Rather than exploiting latency tolerance in the context of making path planning more efficient, in Chapter 3 we will show how to exchange latency tolerance for improvements in delivery reliability.

Relevant Work from Traditional Networking

The methods we devise in this dissertation draw inspiration from work in the traditional networking community.

Winstein et al. [96] propose Sprout, an adaptive transport system designed to improve the performance of time-varying cellular networks. Sprout makes stochastic forecasts of link quality and adjusts its transmission strategies based on these estimates. In particular, Sprout varies the rate at which it transmits packets to reduce self-induced network congestion. The authors demonstrate that Sprout can reliably support network-intensive tasks such as video conferencing for mobile internet users.

Gu and Grossman [34] propose UDP-based Data Transfer Protocol (UDT), a high-performance data transfer protocol built on top of UDP and designed for use over high-speed wide area networks. To make UDP more reliable for the transport of large quantities of data, UDT adds its own congestion control and selective acknowledgments (ACKs). UDT sends ACKs at a fixed interval, so they consume little overhead during high-speed data transfer. However, for low-bandwidth traffic, UDT acknowledges each packet individually, leading to deteriorating performance over lossy links. In Chapter 4, we will compare the performance of our proposed system, High Bandwidth Adaptive Erasure Coding (HBAEC),

to that of UDT.

Google has introduced Quick UDP Internet Connections (QUIC) [74] as a reliable UDP-based replacement for TCP. QUIC will serve as the underlying transport protocol in HTTP/3, the latest version of Hypertext Transfer Protocol (HTTP) [9]. QUIC includes numerous enhancements such as decreased latency in establishing connections, better congestion control, and forward error correction (FEC). QUIC's FEC module does not estimate link quality or attempt to vary redundancy levels, instead sending a small fixed ratio of redundant data. In Chapter 3, we will compare the performance of such an approach to that of our adaptive method, Adaptive Erasure Coding (AEC).

Much research has been devoted to improving the performance of TCP under the moderate amounts of packet loss that occur in traditional wireless networks. One of the best-known examples is TCP Westwood [16], a variant of TCP intended to improve throughput over wireless links. TCP Westwood accomplishes this by using information from the ACK stream to adapt congestion control window parameters. Grieco and Mascolo [32] show that TCP Westwood can improve utilization of wireless links affected by losses not due to congestion. In Chapter 4, we will compare the performance of HBAEC to that of TCP Westwood.

2.2 Communication Decision-Making

Respecting bandwidth constraints in multi-robot systems requires mechanisms for making communication decisions. In this section, we review work related to communication decision-making across fields such as artificial intelligence, robotics, and control theory.

Communicating Based on Decision Theory

A decision-theoretic method evaluates an action based on its expected effect on reward. Computing this value directly for communication decisions in general multi-agent models like decentralized Markov decision processs (Dec-MDPs) or decentralized partially observable Markov decision processs (Dec-POMDPs) is prohibitively complex [30, 65]. To avoid direct computation and its associated costs, Williamson et al. [94, 95] measure the information content in candidate messages and use this as a proxy for a message's value. More concretely, their method computes the KL divergence between an agent's current belief and its belief at the time of its last communication. If this divergence is sufficiently

large compared to some threshold parameter, the agent decides to communicate.

Carlin and Zilberstein [14, 15] and Becker et al. [4] examine the effect of common myopic assumptions made by existing methods. Specifically, they analyze the assumptions that future communication will not occur and that other agents will not initiate communication themselves. They argue that these assumptions lead to over-communication and present an algorithm free from these assumptions. However, the complexity of the resulting algorithm limits it to small problems.

Unhelkar and Shah [92] propose Communication for Time-Critical Collaborative Tasks (ConTaCT), which deliberately deemphasizes uncertainty to make communication decisions tractable in larger domains. ConTaCT assumes that the transition function is deterministic and that the ego-agent knows its own state. Such assumptions are reasonable in systems that have reliable actuation and localization capabilities, and they greatly simplify the task of reasoning about communication decisions. In ConTaCT, the transition function itself is only partially known, as might be the case in an exploration-style robotics task. We draw inspiration from this problem formulation in defining our own in Chapter 5. In ConTaCT, the ego-agent estimates the value of a communication decision by evaluating all agents' stated policies in the ego-agent's best estimate of the transition function. The ego-agent compares the expected reward resulting from those previously stated policies versus updated policies that would be executed if the ego-agent were to communicate. In Chapter 5, we will compare the performance of ConTaCT to that of our proposed method, Optimizing Communication under Bandwidth Constraints (OCBC).

Selecting Communications

Because of the complexities involved in deciding *when* to communicate, few papers have addressed the harder question of *what* to communicate. Roth et al. [77] and Roth [75] point out this deficit and provide one of the few methods for optimizing the content of communication messages. Specifically, they use greedy hill-climbing optimization to select observations to include in a communication. They build this algorithm on top of their previous coordination method in which all agents act on common knowledge [76]. They then try to select the set of observations to add to this common knowledge to maximize team reward. Because they use a general multi-agent model, their approach does not scale beyond small problem domains.

Giamou et al. [27] take a task-oriented approach to selecting what to communicate, proposing a communication planning framework for cooperative simultaneous localization and mapping (SLAM). Specifically, they find a solution for exchanging the minimal amount of raw sensory data without missing out on potential loop closures. However, this method does not generalize beyond cooperative SLAM.

Communicating to Maintain Coordination

In many multi-agent problems, agents must tightly coordinate their actions. Such coordination typically requires significant amounts of inter-agent communication. Therefore, some methods reason about when communication is necessary to maintain coordination.

In Roth [75], Roth et al. [76], agents generate an offline centralized policy that maps possible joint beliefs to joint actions. During execution, agents use this policy to select actions based on the current joint belief. When agents make observations, they consider how communicating those observations would change the joint belief and how the updated joint belief would affect team performance.

Wu et al. [97] also maintain coordination by having agents act only on common knowledge. Agents communicate whenever they detect an inconsistency in their shared belief. This guarantees that agents will remain coordinated even when they forgo communication for some time.

Best et al. [6] present a communication planning algorithm suitable for the decentralized Monte Carlo tree search (Dec-MCTS) coordination framework [5]. In Dec-MCTS, agents maintain belief distributions over the possible future action sequences of their teammates. Best et al. [6] reason about when an agent should request information from a teammate to update this distribution. More specifically, when a belief becomes sufficiently uncertain, the agent requests an updated distribution from the teammate.

The field of consensus and cooperative control considers how coordinating agents should share information so as to converge on a consistent view of inforamtion that is critical to the coordination task. In the most common consensus algorithm, the ego-agent generates its updated belief state based on the accumulated differences between its neighbors' current beliefs and its own current beliefs [69]. Given such an algorithm, a primary area of study is the conditions under which the algorithm will converge, leading to a consistent world view for all agents and thus enabling coordination [13].

Dynamic Information Flow

Kassir proposes the dynamic information flow (DIF) problem, which considers the trade-off between communication limits, computational limits, and information value in decentralized information gathering systems [44, 45, 46]. DIF models the agents in such a system as nodes in a graph, with communication links connecting them. The cost associated with a link corresponds to a communication or computational constraint. Kassir presents methods to maximize the flow of information through the graph subject to these constraints.

Communication Decisions with Deep Reinforcement Learning

Recent papers [26, 86] have used deep reinforcement learning to learn communication policies. Such methods do not have a defined language of symbols for the agents to transmit. Rather, the agents learn to use a continuous-valued broadcast communication channel. Foerster et al. [26] use such a method to solve communication riddles and multi-agent computer vision problems. Sukhbaatar et al. [86] show results for multi-turn games and communication at a traffic junction.

2.3 Erasure Codes Primer

A key technology underlying the algorithms of Chapters 3 and 4 is the erasure code. Because erasure codes lie outside the technological repertoire of the typical roboticist, we provide a brief primer here. We focus on describing the properties of erasure codes we use in later chapters and conveying intuition for how they work. For the interested reader, more comprehensive coverage exists in other places (e.g. [66, 67, 79]).

2.3.1 Introduction

Erasure codes are a type of forward error correction (FEC). In general, FEC is a way of ensuring robust communication over lossy channels by encoding data with an error correcting code. This encoding process provides redundancy: if some portion of the input data is lost or corrupted, the receiver can recover the original data from the redundant data.

The Erasure Channel

Erasure codes are based on a model of the communication channel known as the erasure channel. In the erasure channel, transmitted data is either received without error or else the receiver is notified that the data was lost. In a binary erasure channel (BEC), the transmitted data is a single bit $x \in \{0, 1\}$. The channel erases this bit with probability p_e. Therefore, the conditional probability of received data $y \in \{0, 1, \mathrm{X}\}$ is given by

$$\mathbb{P}(y|x) = \begin{cases} 1 - p_e & y = x \\ p_e & y = \mathrm{X} \end{cases}, \tag{2.1}$$

where X denotes an erasure.

The packet erasure channel (PEC) is analogous to the BEC but with the basic unit of transmitted data being a network packet rather than a single bit. The PEC model is suitable for the upper layers of a modern network stack, since such layers have the following properties:

1. Data reaching the upper layer is free of errors because lower layers verify its integrity through mechanisms such as cyclic redundancy checks (CRCs).

2. Packets at the upper layer contain sequence numbers, which can provide evidence of missing packets to the application.

Together, freedom from errors and notice of erasures are the key components of an erasure channel.

Note that the PEC assumes that erasures are independent and identically distributed (IID). Although packet loss in real-world networks exhibits temporal dependence [98], the PEC is still a useful coarse-grained model of network channels.

Erasure Codes over an Erasure Channel

Figure 2.1 illustrates how an erasure code operates on an erasure channel. An original message of k symbols is transformed into a larger message of n symbols through encoding. This larger, encoded message is known as the *code word*. The symbols of the code word are transmitted through the erasure channel, where some portion of them may be lost. When a subset of k' symbols are received successfully, the receiver can decode those symbols to recover the k symbols of the original message.

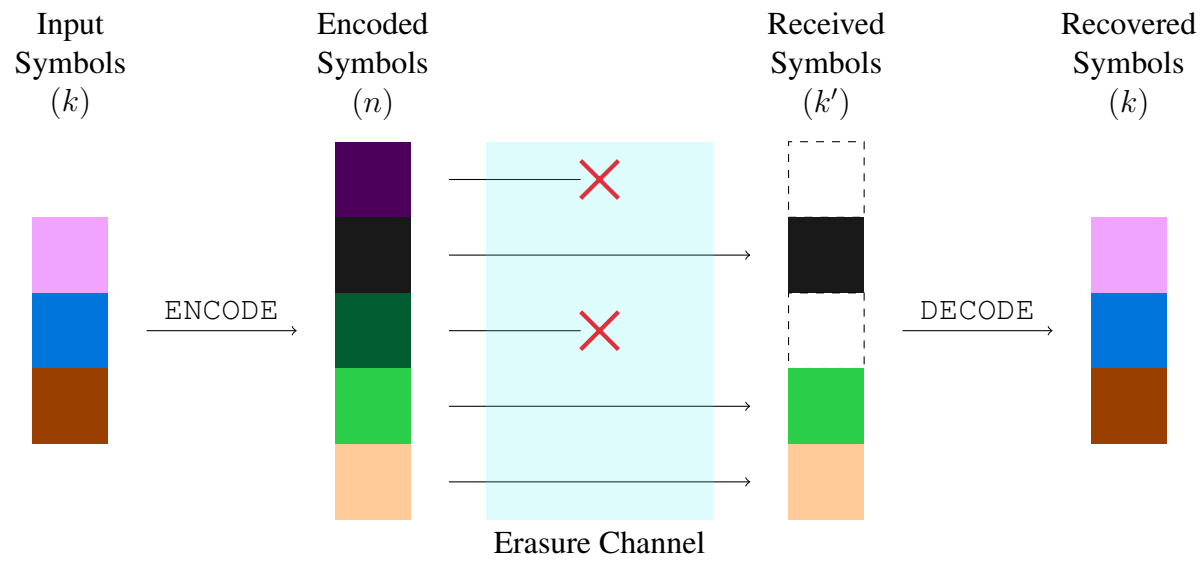

Figure 2.1: Illustration of erasure codes and their application to the erasure channel. k input symbols are encoded to produce the n symbols of the code word. These symbols are transmitted across the erasure channel, at which time some portion of the symbols may be erased. As long as a sufficient number k' of symbols are received, the original input message can be recovered.

The ratio of the size of the original message to the size of the code word, $k/n < 1$, is known as the *code rate*. The *reception efficiency* is the ratio of the number of symbols needed for decoding to the number of symbols in the original message, $k'/k \geq 1$. When n symbols are transmitted over an erasure channel with erasure probability p_e, the distribution of the number of received symbols is binomial. Therefore, the expected number of received symbols is $(1 - p_e)n$. Successfully transmitting a message of size k over such a channel requires, on average, $n = k'/(1-p_e)$ symbols for an erasure code with reception efficiency k'/k.

In practice, the erasure probability p_e is unknown, so the transmitter's estimate of it is uncertain. The reception efficiency k'/k may itself be probabilistic in nature for some erasure codes (see Section 2.3.3). Together, these uncertainties make it highly challenging to select the number of encoded symbols n. Providing solutions to these challenges represent key contributions of Chapters 3 and 4.

2.3.2 Maximum Distance Separable Codes

A maximum distance separable (MDS) erasure code has the following optimal decoding characteristic:

Property 2.3.1 (Decoding of MDS erasure codes)**.** *Any k symbols of an n-symbol code word produced by an MDS erasure code can be used to decode the original k-symbol message.*

In other words, an MDS code has $k' = k$ and reception efficiency 1. Reed Solomon codes [68], the type we will use in Chapter 3, are an example of MDS erasure codes.

We will now give two explanations to provide intuition for how MDS erasure codes work. We term these *The Polynomial Story* and *The Matrix Story* as they draw on polynomial curve-fitting and matrix algebra, respectively, for their explanations. The explanations are equivalent, but they highlight different aspects of the encoding and decoding process. We use these explanations to build intuition for MDS erasure codes rather than to provide detailed coverage. For a more rigorous introduction, refer to the textbook of Reed and Chen [67].

The Polynomial Story

Figure 2.2 tells the *polynomial story* of encoding and decoding an MDS erasure code. In the first plot of the figure, the k input data symbols appear in a 2D plane. The horizontal and

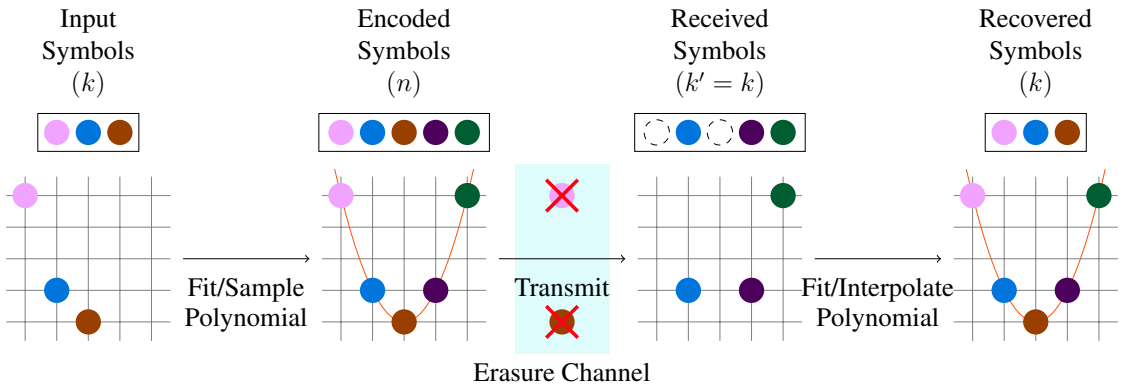

Figure 2.2: Illustration of the *polynomial story* of encoding and decoding a maximum distance separable (MDS) erasure code. A Lagrange polynomial is fit to the k input symbols. The polynomial is then oversampled to produce redundant symbols. All symbols are transmitted over the erasure channel, resulting in the erasure of some subset of the symbols. As long as $k' = k$ symbols are received, the original Lagrange polynomial can be fit to those symbols. The original input symbols can be recovered by interpolating this polynomial.

vertical positions of each symbol correspond to the symbol's order in the sequence and data value, respectively. In the second plot of the figure, we construct a Lagrange polynomial $p(x)$ such that $p(i)$ is equal to the value of each data symbol i. We produce the redundant symbols by evaluating the polynomial at $k+1, \ldots, n$.

The erasure channel may erase some portion of the encoded symbols during transmission. Recall from Section 2.3.1 that the erasure channel provides notice of the locations of these erasures, allowing us to place the received symbols as in the third plot of Figure 2.2. As long as any $k' = k$ symbols are received, we can reconstruct polynomial $p(x)$ as in the fourth plot of the figure. By interpolating $p(x)$, we can recover the missing symbols.

In practice, erasure codes perform all of these computations over a finite field $\mathcal{F}$ of order at least n.

The Matrix Story

We can also explain MDS erasure codes in terms of matrix algebra, as shown in Figure 2.3. We left-multiply the vector of k input symbols by an $n \times k$ encoding matrix to produce a vector of n encoded symbols (the code word).

The encoding matrix has a special structure. The first k rows of the encoding matrix form a $k \times k$ identity matrix. This results in the first k encoded symbols being equal in value to the k input symbols. When the input symbols appear in the code word, the erasure code

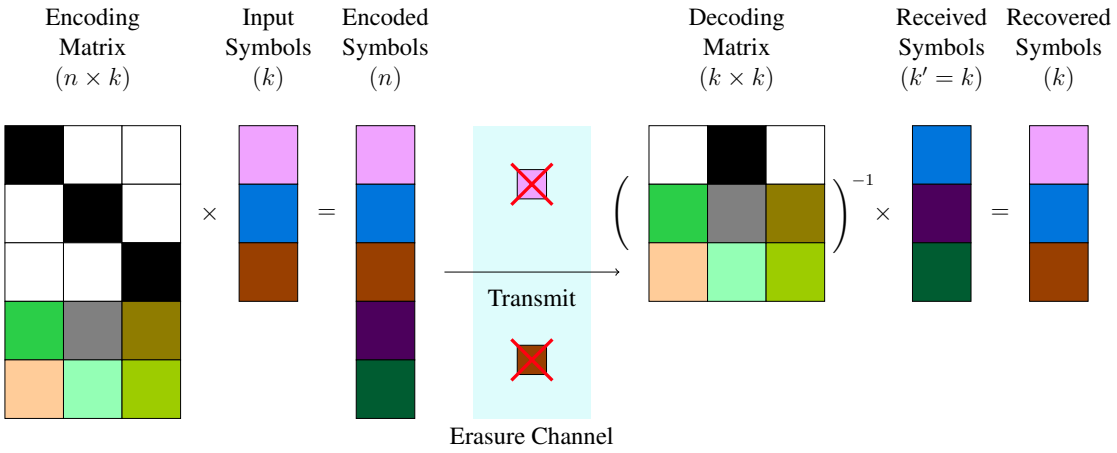

Figure 2.3: Illustration of the *matrix story* of encoding and decoding a maximum distance separable (MDS) erasure code. The vector of input symbols is left-multiplied by an encoding matrix to produce a code word. The encoded symbols pass through the erasure channel, leading to the erasure of some symbols. The original symbols can be recovered by inverting the matrix formed by a subset of k rows of the encoding matrix and multiplying by the received symbols. This is possible since any k rows of the original encoding matrix will form a nonsingular matrix.

is said to be *systematic*. Non-systematic MDS erasure codes also exist, such as the original construction of Reed and Solomon [68] using a Vandermonde matrix. Non-systematic MDS erasure codes are encoded and decoded in the same way as systematic ones, but they use a different encoding matrix without the initial identity rows.

The second key property of the encoding matrix is that any matrix formed from k of its rows is guaranteed to be nonsingular. This is trivially true for the identity matrix of the first k rows. The remaining $n - k$ rows are constructed so as to maintain this property. Note that the encoding matrix does not depend on the input data, so it can be computed once for a particular (n, k) erasure code and then reused across inputs.

The right side of Figure 2.3 shows how this property is used to recover the input symbols. Consider any k encoded symbols received from the erasure channel. Because of the row independence property stated in the previous paragraph, the matrix formed from the k rows corresponding to these received symbols is guaranteed to be nonsingular. Therefore, we can solve for the original input symbols by left-multiplying the received symbols by the inverse of this matrix.

2.3.3 Fountain Codes

The MDS codes of the previous section are one class of erasure codes. In this section, we introduce a second class: fountain codes. We describe the basic properties of fountain codes and provide intuition for their operation. For a more detailed introduction, please refer to the tutorials of Qureshi et al. [66] and MacKay [53].

A good way to understand fountain codes and how they differ from MDS codes is to examine the properties of the ideal (abstract) fountain code as described by Byers et al. [12]:

1. There should be no limit on the number of encoded symbols that can be produced from a set of source symbols.

2. It should be possible to recover the original set of source symbols from any k encoded symbols with high probability.

3. Encoding time should be linear in the symbols' size, and decoding time should be linear in the number of encoded symbols.

The first property is known as the *rateless* property, which lends fountain codes another name sometimes used in the literature: rateless codes. Recall from Section 2.3.1 that the code rate of an erasure code is the ratio of the number of input symbols k to the number of encoded symbols n. For an MDS code, the code rate is fixed at the time of encoding. Once an (n, k) MDS code has been used to encode a set of input symbols, it is not possible to produce encoded symbol $n + 1$ such that any k of the $n + 1$ symbols are guaranteed to recover the original input.

A fountain code can produce an unlimited number of encoded symbols from a given set of source symbols. To make this possible, fountain codes give up the optimal decoding property of MDS codes (Property 2.3.1). Rather, fountain codes have the following probabilistic decoding characteristic:

Property 2.3.2 (Decoding of fountain codes)**.** *With high probability, any k' symbols of an n-symbol code word produced by a fountain code can decode the original k-symbol message, for $k' = k + o$, where $o = 0, 1, \ldots$ is a small amount of overhead. The probability of decoding failure decreases with increased o.*

Because fountain codes are not optimal from a decoding perspective, it is possible that more than k encoded symbols may need to be received to decode an input of size k. Furthermore, because encoded symbols are generated independently, the probability of decoding

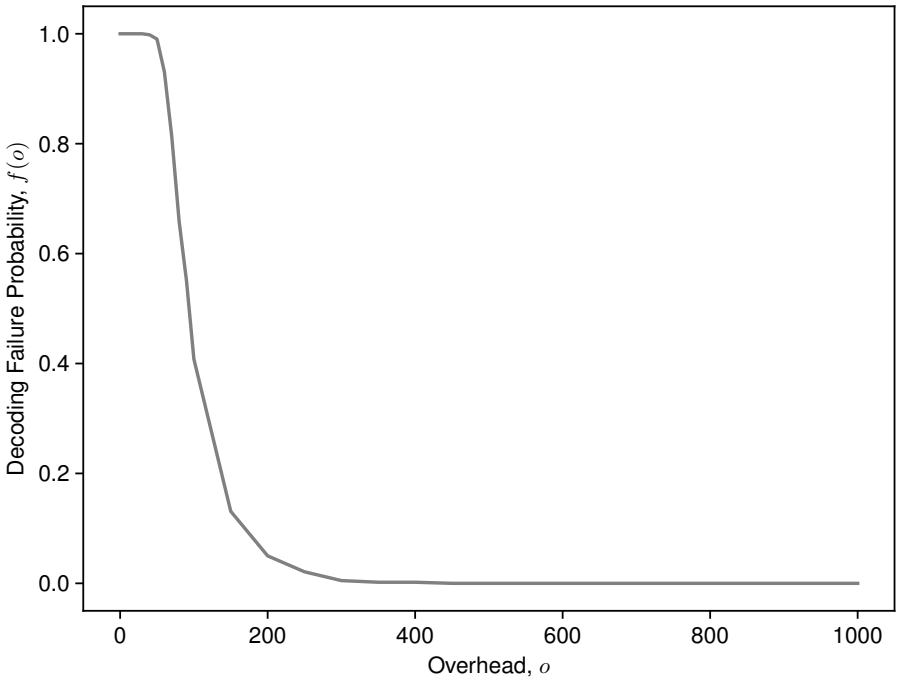

Figure 2.4: Example failure-overhead curve for online codes [56], the type of fountain code we use in Chapter 4.

failure occurring for a particular set of encoded symbols depends only on how many symbols are received. Let the *overhead* of a code be o if $k' = k + o$ symbols are needed for decoding to succeed. We define the *failure probability* $f(o)$ to be the probability that decoding fails with overhead o. The set of ordered pairs $\{(0, f(0)), (1, f(1)), \dots\}$ form the *overhead-failure curve*, which is a characteristic of a particular fountain code (see Figure 2.4 for an example failure-overhead curve for online codes [56]). The ideal overhead-failure curve should be steep: each additional received symbol beyond k rapidly decreases the probability of decoding failure. Much fountain code research focuses on approaching this ideal.

Encoding Concept

Encoding data with a fountain code is simple: randomly select a set of input symbols, XOR them with one another, and use the output as the encoded symbol. The difficulty lies in determining how to sample the input symbols for encoding. This sampling is governed by degree distribution $\mathcal{D}$, a probability distribution on the degree of each encoded symbol, that is, the number of input symbols used to produce it. The encoding sampling process

has two steps: (1) sample degree d from $\mathcal{D}$, then (2) uniformly sample d input symbols to use for encoding.

Figure 2.5 illustrates this encoding process. Input symbols $x_1, \ldots, x_4$ are used to produce encoded symbols $y_1, \ldots, y_5$. To produce each encoded symbol, we first sample a degree d from the distribution $\mathcal{D}$. We then sample d times from $\{1, \ldots, k\}$ to determine which input symbols will be used to produce the encoded symbol. The encoded symbol is then the XOR of the selected input symbols. The connections between the input symbols and the encoded symbols form a bipartite graph known as a Tanner graph [88].

The degree distribution $\mathcal{D}$ is the key element of a fountain code. In general, it should have an adequate balance between small degrees, which lead to fast encoding and decoding, and large degrees, which provide greater redundancy coverage. Figure 2.6 shows an example degree distribution for online codes [56], a type of fountain code we use in Chapter 4.

Decoding Concept

Fountain code decoding is conceptually simple. We can express the relationship between input symbols and encoded symbols in terms of a system of linear equations. For example, the equations for the symbols encoded in Figure 2.5 are given by

$$\begin{aligned} y_1 &= x_2 + x_4 \\ y_2 &= x_1 + x_3 \\ y_3 &= x_1 + x_2 + x_3. \\ y_4 &= x_2 + x_3 + x_4 \\ y_5 &= x_1 \end{aligned} \tag{2.2}$$

The encoded symbols are received from the erasure channel, and the goal is to recover the input symbols.

The fastest way to solve the system is a *belief propagation decoder* [79], which is akin to the backsubstitution step of Gaussian elimination. Figure 2.7 illustrates this procedure. Each iteration of the decoding algorithm begins by finding an encoded symbol of degree one, which in the example is y_5. We use this encoded symbol to recover the input symbol to which it is connected (x_1). For each other encoded symbol to which the input symbol is connected (y_2, y_3), we XOR the encoded symbol with the input symbol and remove the connection from the graph. We repeat this procedure until we have solved for all input

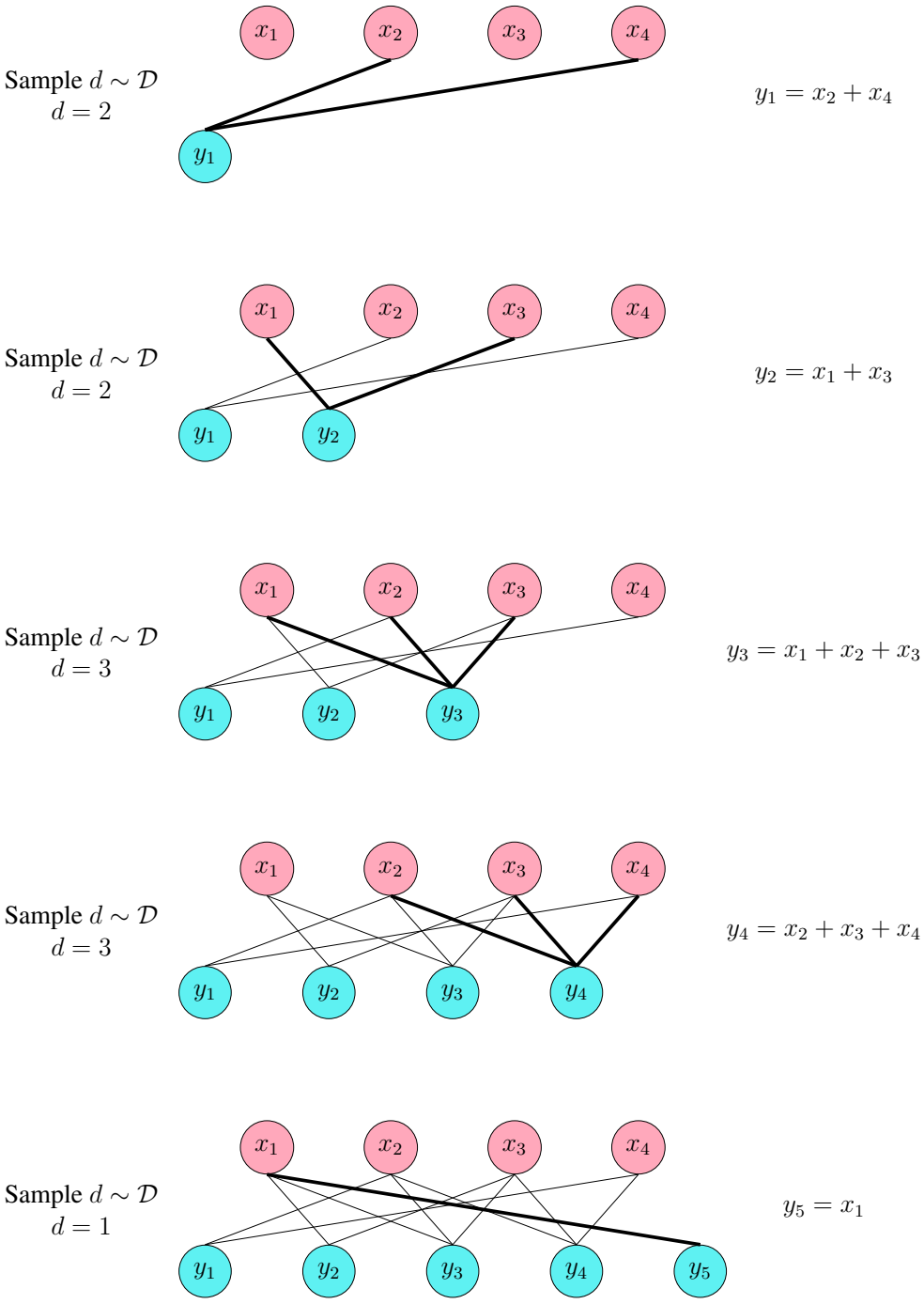

Figure 2.5: Tanner graph illustration of the fountain code encoding process. Input symbols $x_1, \ldots, x_4$ are used to produce encoded symbols $y_1, \ldots, y_5$. For each encoded symbol, a degree d is sampled from the degree distribution $\mathcal{D}$. Then, d input symbols are sampled uniformly at random; the XOR of these input symbols is the encoded symbol.

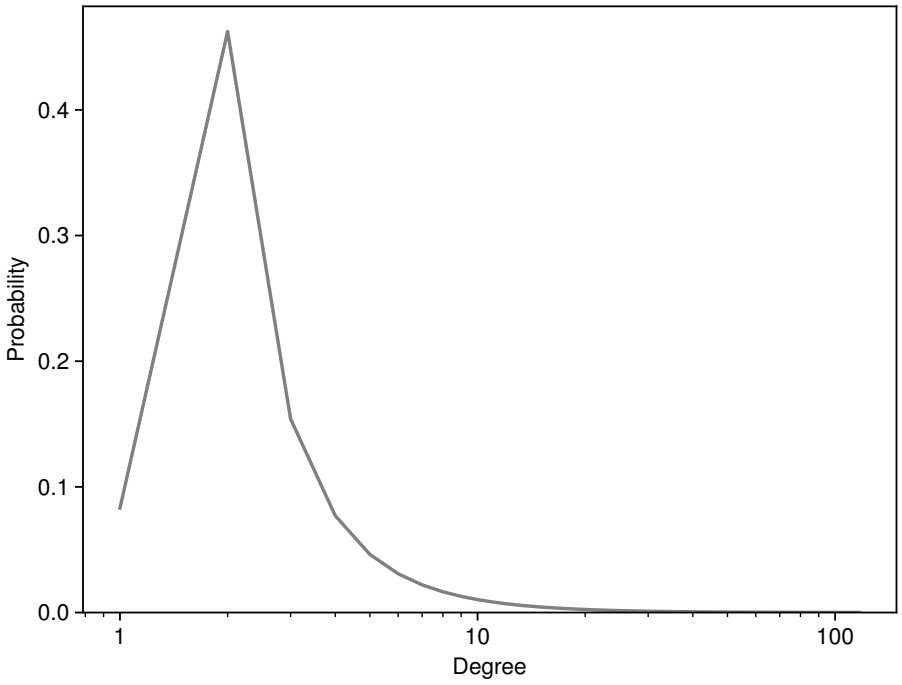

Figure 2.6: Example degree distribution $\mathcal{D}$ for online codes [56], the type of fountain code we use in Chapter 4.

symbols.

The degree distribution $\mathcal{D}$ of a fountain code can be designed such that belief propagation decoding will succeed with high likelihood for small amounts of overhead. However, it is possible that this decoding will fail, requiring additional encoded symbols to be produced. To overcome this weakness, Shokrollahi et al. introduced *inactivation decoding*, which fully implements Gaussian elimination while maintaining the efficiency of the belief propagation algorithm. Inactivation decoding is beyond the scope of this primer; we refer the interested reader to [79] for a thorough introduction and discussion.

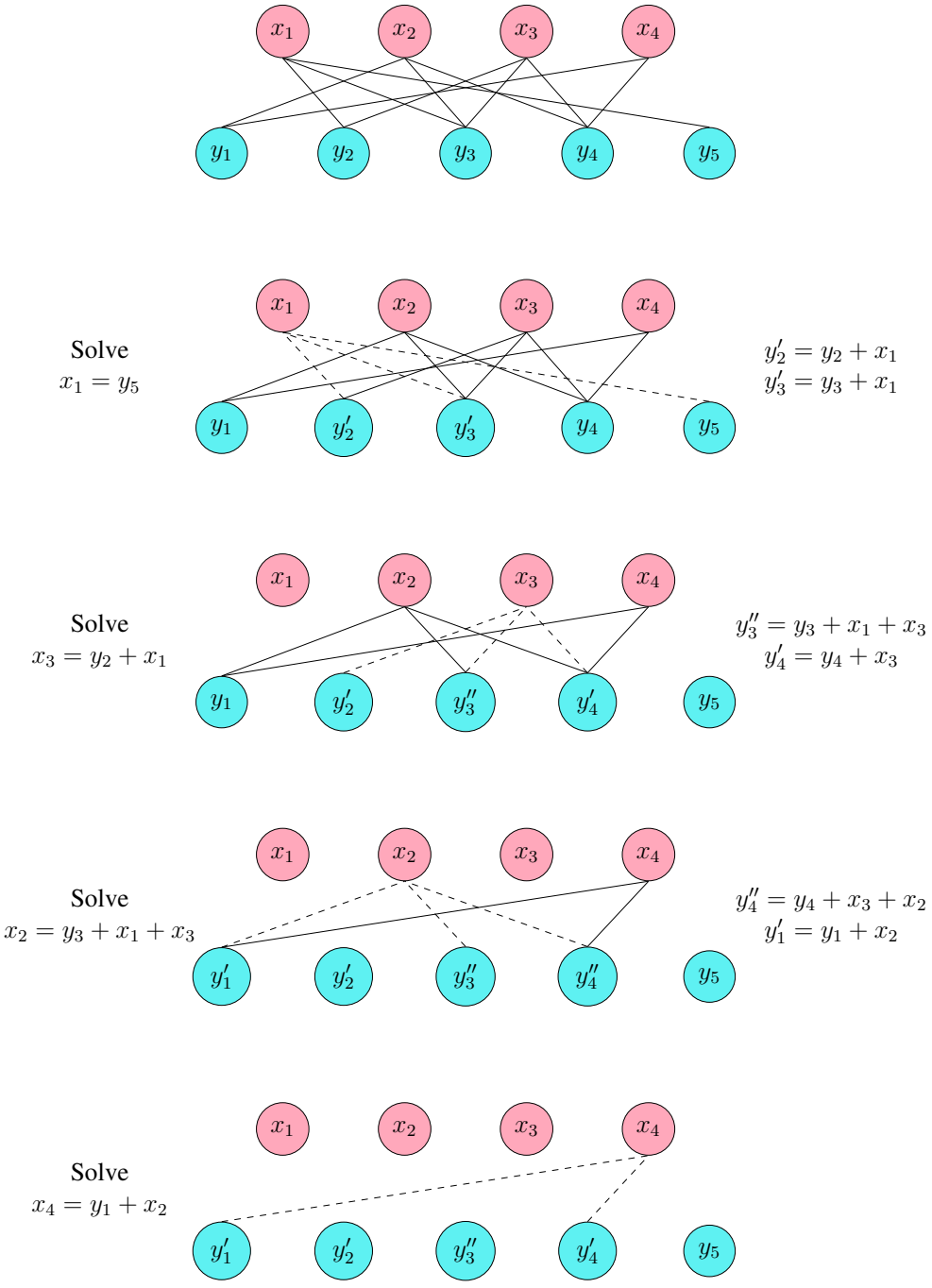

Figure 2.7: Tanner graph illustration of the fountain code decoding process. Encoded symbols of degree one are used to solve for input symbols. Any connections between the solved input symbols and encoded symbols are removed from the graph. This process repeats until all input symbols have been solved for.

CHAPTER 3

Delivering Latency-Tolerant Traffic[1]

3.1 Introduction

In packet-switched network communication, *packet loss* occurs when a packet of data fails to reach its intended destination. *Latency* is the amount of time it takes for a packet to travel from sender to receiver. These two phenomena may share a loose relationship with one another, such as when lost TCP segments are retransmitted, resulting in higher latency in the overall transmission process. In other cases, packet loss and latency are less correlated, with respective causes such as interference (packet loss) or travel distance (latency). In this chapter, we show a means by which packet loss and latency can be intentionally related to bring about improved network performance. More specifically, we describe how applications can exploit latency tolerance to mitigate packet loss.

Data loss and corruption comprise one of the most widely recognized and well-studied issues in network communications. Loss and corruption are most prevalent in wireless applications, and mobile robotic systems are dependent on such wireless networks for communication. In this dissertation, we focus on mobile robotic systems that communicate with signals propagated in the form of electromagnetic radiation (for issues related to acoustic communication in underwater robotic systems, please refer to [41] or [101]).

Three key physical factors contribute to packet loss in this type of wireless network: *small-scale fading*, *shadowing*, and *path loss* [59]. Small-scale fading, also known as multipath fading, occurs when out-of-phase replicas of the transmitted signal arrive at the receiver and interfere constructively or destructively, a result of the signal reflecting off of objects in the environment and arriving at the receiver via different paths. Shadowing varies over larger distances but is also due to the interaction of the signal with objects in

[1] Adapted from Marcotte and Olson [54]

the environment. Path loss is the distance-dependent component, which is caused by the reduction in signal power as distance from the transmitter increases. All three of these factors affecting packet loss are present in mobile, multi-robot systems: these systems may operate in harsh, rugged environments (multipath and shadowing), and robots may travel long distances from one another or from a base station (path loss). For this reason, packet loss is a particularly acute problem in mobile robotic systems (see [31] for more on wireless channels and their relation to robotic systems).

On the other hand, multi-robot systems may be capable of tolerating latency in their network traffic. For example, consider the types of network communication occurring in a team of robots performing an urban reconnaissance task, such as Team Michigan from the MAGIC 2010 competition [62, 63]. Some of the team's network traffic must be delivered promptly, such as camera imagery for teleoperation or broadcasted warnings of dangerous objects. Other traffic, such as map updates, is less time-sensitive since individual robots can dead reckon safely until they receive a loop closure. In still other cases, such as messages allocating tasks to individual robots, added latency is not catastrophic but may degrade team performance.

In this chapter, we introduce a mechanism by which a mobile robotic system can exploit latency tolerance to improve packet delivery performance while minimizing added overhead. Our proposed system groups packets and applies erasure coding to the group to produce redundant packets. This added redundancy can be used to recover data packets dropped in transmission. As illustrated in Figure 3.1 the recovered packet is passed to the receiving application once enough redundant packets have been received for decoding to take place. The time between the transmission of the data packet and the recovery of it is the total latency of that packet.

We call our system Adaptive Erasure Coding (AEC). In AEC, an application can specify a tolerable amount of latency (or equivalently, a delivery deadline) for each packet, as well as a maximum tolerable likelihood of the packet being dropped. Packets with similar delivery deadlines are grouped together and encoded with a maximum distance separable (MDS) erasure code (see Section 2.3.2). Lost packets can be recovered by decoding the redundant data transmitted along with the group. We compute the minimum amount of redundancy that must be added so that the probability of a packet being dropped and not recovered is within the application-specified limits. To enable this, we describe a technique to estimate the distribution of possible instantaneous packet loss rates from historical measurements of packet loss.

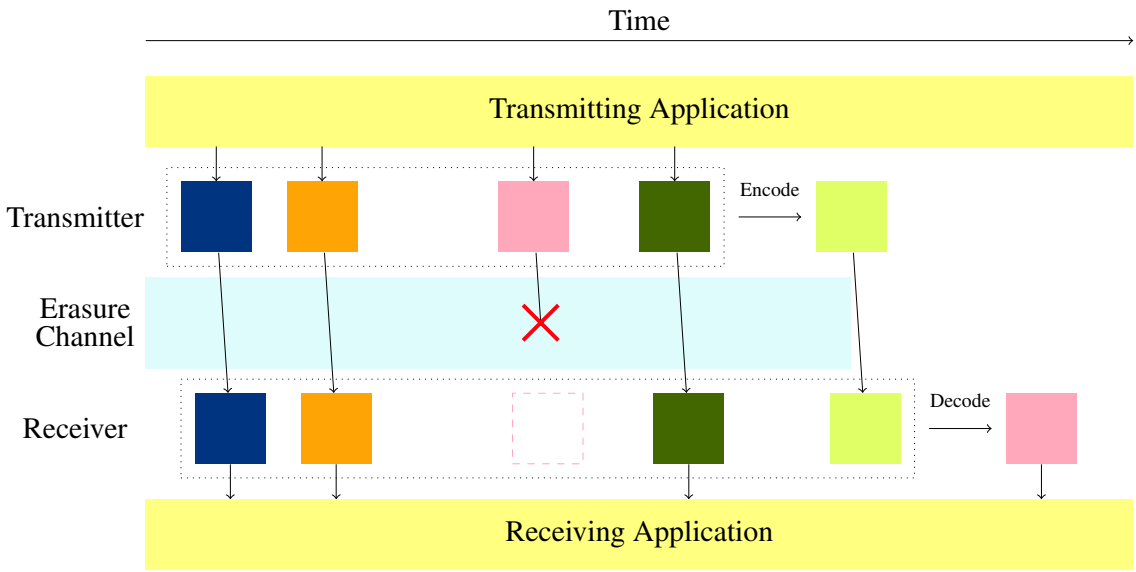

Figure 3.1: Illustration of the Adaptive Erasure Coding (AEC) mechanism for exchanging latency tolerance for improved resiliency to packet loss. A sequence of data packets arrive at the AEC transmitter and are grouped together and encoded to produce a redundant packet. When one of the data packets is lost during transmission over the erasure channel, the received packets from the group can be decoded to recover the lost packet. The AEC system meets the Quality of Service (QoS) needs of the traffic by ensuring that the time between transmission and eventual recovery fits within the application-specified latency tolerance of a packet.

The key insight of this chapter is that we can exchange latency tolerance to improve communication reliability. Most existing work considers latency to be undesirable, but we recognize tolerance to latency as a resource that can be exploited. Further, we argue that this resource exists in many multi-robot systems. By injecting this application-specific characteristic into the network protocols of a multi-robot system, we can achieve more reliable communication.

Our technical contributions in this chapter are:

- A mechanism by which tolerance to latency can be exchanged for improvements in the probability of successful packet delivery (Section 3.3),
- An algorithm to determine the minimum amount of redundancy to transmit such that the probability of unrecoverable packet loss is sufficiently small (Section 3.4.1), and
- A technique to estimate the distribution of possible instantaneous packet loss rates from historical packet loss measurements (Section 3.4.2).

3.2 Existing Approaches to Mitigating Packet Loss

A modern network stack has many mechanisms in place that mitigate the effects of packet loss. This chapter's method complements some of these mechanisms or else serves as an alternative in the networks of mobile multi-robot systems.

A cyclic redundancy check (CRC) is an error-detecting code used to ensure the integrity of raw data after storage or transmission [64]. The CRC is a function that takes the raw data as input and outputs a fixed-length sequence of redundant data. In network communications, the sender includes this redundancy along with the transmitted data, allowing the receiver to verify (with high probability) the integrity of the received data. This check occurs at the link layer in a IEEE 802.11 network stack. As a result, data is either passed to higher layers of the network stack free from errors or else not at all. That is, the channel can be modeled as a packet erasure channel (PEC) (see Section 2.3.1). Our method assumes that this property holds; thus, a CRC must be used alongside it.

The automatic repeat request (ARQ) error control method employs acknowledgments (ACKs) and timeouts to transmit data reliably over a lossy channel. In its simplest form, the sender transmits a message and waits for an ACK from the receiver. If the ACK does not arrive within a specified amount of time, the sender retransmits the original message.

ARQ is widely used in network communications due to its simplicity and effectiveness, with common examples being TCP and link-layer unicast. ARQ can and should be used in many robotics applications, but it has several drawbacks that prevent it from being a panacea for packet loss. First, ARQ does not generalize to broadcast (one-to-many) communication, which is a common communication model used in multi-robot systems. With a potentially unknown number of receivers, the sender cannot know when a message has been universally received and acknowledged.

The second major issue with using ARQ in robotic networks is that it is often combined with congestion control (e.g. as in TCP [20]). Congestion or flow control is a mechanism by which a sender may increase the amount of time between transmissions in order to alleviate potential overuse of a network channel. In traditional (i.e. wired or managed wireless) networks, packet loss is assumed to result from congestion. Thus, the timeout interval is increased when packet loss is detected. However, packet loss in wireless networks may be caused by environmental factors such as multipath or path loss (see Section 3.1). When congestion control is activated due to this type of packet loss, the transmission process is unnecessarily delayed. We discuss this problem further in Chapter 4.

Rate adaptation is the process of varying the transmission bitrate in response to packet loss. Data transmitted at a lower bitrate can use more robust encoding, leading to a higher likelihood of successful delivery [59]. However, lowering the bitrate reduces the amount of data that can be transmitted over the link. Rate adaptation aims to select the bitrate that optimizes the expected throughput of a link [8]. Our method can operate on links that also employ rate adaptation, or it can be applied to fixed-bitrate links (e.g. IEEE 802.11 broadcast).

Forward error correction (FEC) is already used in some existing network protocols. A prominent example is Google's Quick UDP Internet Connections (QUIC) protocol [74]. QUIC forms packets into groups, then computes a single redundant packet that is the XOR of the packets within the group. Google's experiments show that 28-65% of packet loss events occurring over the Internet involve a single packet [87]. Losses of multiple packets would not be mitigated by the single redundant packet of QUIC, but using a larger fixed amount of FEC would impose unnecessary overhead over high-quality connections. For this reason, we consider the problem of determining a suitable amount of FEC to include with each transmission. To the best of our knowledge, our system is the first to apply variable amounts of FEC based on a probabilistic estimate of packet loss.

3.3 Exchanging Latency Tolerance for Reduced Packet Loss

The AEC system, which is illustrated in Figure 3.1, includes modules at both the source and destination nodes. In our implementation, these modules exist in user-space and transmit datagrams using UDP. Concretely, they exist at the same level of abstraction as the C language's `sendto` and `recvfrom` functions.

At the source node, packets are passed to the AEC module by the robot's other applications. Each packet that arrives at the transmitting module is marked with a maximum tolerable latency and a target reception probability. The AEC module attempts to meet both of these QoS requirements, though the transmission process is still best-effort rather than guaranteed. The transmitting module immediately sends each data packet but retains a copy of it. The transmitting module groups packets together based on their delivery deadline, which is computed based on the packet's arrival time and specified latency tolerance. The assigned delivery deadline and target reception probability of the group are the earliest deadline and the maximum reception probability of the individual packets, respectively. Once the group has been formed and assigned a target reception probability, the AEC module computes the minimum amount of redundancy needed in order to meet that reception probability (see Section 3.4). The data packets are encoded to produce redundant packets, which are then transmitted.

The receiving module collects packets from different groups as they arrive. Received data packets are passed to the application immediately, with a copy of their contents retained for use in decoding other packets from the group. If a sufficient number of data and redundancy packets from a group are received, they are decoded to produce missing data packets.

3.4 Adjusting Encoding Strength

In this section, we consider the question of how much redundancy to transmit with a group of packets over a lossy channel. This decision includes a trade-off between the likelihood of a successful delivery and the cost of the overhead added to the network. If too little redundancy is added, some dropped packets from the group may be unrecoverable. However, the amount of bandwidth available in a wireless network is limited (see Chapter 5), so adding too much redundancy wastes that bandwidth, leaving less of it available for other

traffic.

Some existing systems using FEC, such as Google's QUIC, apply a constant amount of redundancy regardless of network conditions. In particular, QUIC adds a single redundant packet per group of data packets. Such an approach might be wasteful under benign network conditions when loss is rare, or it may be inadequate when loss is more frequent. The networks of mobile robotic systems in particular exhibit large variability in packet loss rates. Therefore, it is important that AEC accounts for this variability and adapts the amount of redundancy based on network conditions.

3.4.1 Calculating Encoding Strength

We formulate the encoding strength decision as an optimization problem. Recall the optimal decoding property (Property 2.3.1) of MDS erasure codes:

Property 2.3.1 (Decoding of MDS erasure codes)**.** *Any k symbols of an n-symbol code word produced by an MDS erasure code can be used to decode the original k-symbol message.*

Therefore, for a group of k packets, we determine the minimum amount of redundancy $n - k$ such that the probability of fewer than k packets being received is below some threshold ϵ. We can write this optimization problem as

$$\begin{aligned} \text{minimize} \quad & n - k \\ \text{s.t} \quad & \mathbb{P}\left(X_n < k\right) < \epsilon \end{aligned}, \tag{3.1}$$

where X_n is a random variable corresponding to the number of packets received when n packets are transmitted.

Recall from Section 2.3.1 that the packet erasure channel is parameterized by p_e, which is the probability of an erasure occurring during transmission. Let $y = 1 - p_e$ be probability of successful reception over such a PEC. When n packets are transmitted over a PEC with reception rate y, the number of received packets follows the binomial distribution $\mathcal{B}\left(n, y\right)$. Further recall that the PEC is an abstraction and that, in practice, the true y is unknown. Let $f_Y(y)$ be the estimated probability density function (PDF) of y (see Section 3.4.2 for our method of estimating $f_Y(y)$) and $P_X(x; n, y)$ be the binomial probability mass function (PMF) of x given y. X_n follows a compound distribution where $f_Y(y)$ are the weights for

components $P_X(x; n, y)$. We can then express the PMF of x as

$$P_X(x; n) = \mathbb{P}(X_n = x) = \int_0^1 P_X(x; n, y) f_Y(y) dy. \tag{3.2}$$

From this, we get the cumulative distribution function (CDF) of X_n,

$$F_X(x; n) = \mathbb{P}(X_n \leq x) = \int_0^1 F_X(x; n, y) f_Y(y) dy, \tag{3.3}$$

where $F_X(x; n, y)$ is the binomial CDF of x given y. Because X_n is a discrete random variable, we can obtain the strict inequality needed for the constraint of Equation (3.1) directly from the CDF, that is,

$$\mathbb{P}(X_n < x) = \mathbb{P}(X_n \leq x - 1) = F_X(x - 1; n). \tag{3.4}$$

Finally, we can re-write Equation (3.1) as

$$\begin{aligned} \text{minimize} \quad & n \\ \text{s.t} \quad & F_X(k - 1; n) < \epsilon \end{aligned}. \tag{3.5}$$

Expressed in words, we seek to find the minimum number of packets to transmit, n, such that the probability of decoding failure is less than some threshold ϵ.

Solving Equation (3.5) in closed form would require a quantile function (inverse CDF) for X_n, but no such function exists for an arbitrary compound distribution. Furthermore, no closed form exists for $F_X(x; n)$, so no analytical method exists to determine whether the constraint of Equation (3.5) is satisfied.

We therefore must resort to numerical techniques to solve Equation (3.5). This is tractable in part because n is discrete and has a finite range, that is $k \leq n \leq n_{\text{max}}$, where the upper limit n_{max} is a consequence of the type of erasure code used (see Section 2.3.2) as well as the available channel bandwidth. Given a value of n, we can estimate $F_X(x; n)$ empirically through sampling. Concretely, because X_n follows a compound distribution, we can first sample $y \sim Y$, then sample $x \sim \mathcal{B}(n, y)$.

Algorithm 3.1 details our method for solving for n^* (the optimal value of n) for a group of packets of size k and an estimated reception distribution Y. The function performs a binary search through the possible values of n. For each tested value of n, it samples to empirically estimate the CDF $\hat{F}_X(x; n)$. Through the binary search, the function determines

Algorithm 3.1 Calculating Optimal Encoding Strength

```
1:  function CalculateEncodingStrength(k, Y, ε)
2:      Initialize n* ← n_max
3:      Initialize left ← k, right ← n_max
4:      while left ≤ right do
5:          n ← ⌊(left+right)/2⌋
6:          for S iterations do
7:              Sample y ∼ Y
8:              Sample x ∼ B(n, y)
9:              Update F̂_X(x; n)
10:         if F̂_X(k − 1; n) < ε then
11:             n* ← n
12:             right ← n − 1
13:         else
14:             left ← n + 1
15:     return n*
```

the minimum value of n that satisfies the constraint of Equation (3.5). If no value of n satisfies the constraint, the function returns n_{max}.

Using Monte Carlo sampling to estimate $\hat{F}_X$ has the added benefit that it could be extended to more complex models of the communication channel. For example, in this dissertation we use the PEC model, which assumes that packet loss occurs independently. However, packet loss is known to exhibit temporal correlation [98]. Future work could extend the empirical estimation process to model this type of loss.

3.4.2 Link Quality Estimation

The encoding strength calculation described in the previous section and detailed in Algorithm 3.1 requires an estimate of the packet reception ratio (PRR) in the form of the distribution Y. Before describing our method of estimating link quality, we briefly review relevant existing work in the area. The interested reader may refer to the surveys of Baccour et al. [2] or Lowrance and Lauf [51] for further information.

The two primary means of estimating link quality employ physical and logical metrics, respectively. By physical metric, we mean measurements obtained from the hardware of a wireless receiver, such as the signal-to-noise ratio (SNR) or received signal strength indicator (RSSI). A logical metric (e.g. PRR) is one that is made available through software, typically at the application layer of the networking stack.

SNR strongly correlates with PRR [78], but it is not defined as a part of IEEE 802.11 and thus is not provided by all wireless vendors. RSSI has a weaker correlation with PRR, but is more readily available [51]. Methods that use physical metrics like RSSI (e.g. [78]) may map those values to an estimated PRR value according to some empirically generated curve. When combined with spatial estimation of signal strength, this type of approach can enable predictions of link quality across space and time [24, 25, 85].

Physical metrics have three primary drawbacks for our application. First, they are not as readily available to high-level (i.e. user-space) applications. They are also hardware-dependent: the RSSI-PRR mapping curve might vary depending on the vendor's implementation of RSSI [51]. Finally, physical metrics are limited to a single link. If the communication channel comprises multiple links, such as in a multi-hop network, the physical metrics available at a single node are insufficient.

In this dissertation, we employ the logical metric PRR. More specifically, we estimate the instantaneous PRR through historical measurements of PRR. One can measure PRR by including a sequence number in each transmitted packet; packet losses can be inferred through gaps in received sequence numbers. For example, we use the standard formula

$$z_t = \begin{cases} \frac{\max \Omega_t - \min \Omega_t + 1}{|\Omega_t|} & |\Omega_t| > 0 \\ 0 & \text{o.w.} \end{cases}, \tag{3.6}$$

where z_t is the measured PRR during time interval t based on the set of received packet sequence numbers Ω_t.

A common approach to estimating instantaneous PRR is to apply a simple moving average (SMA) or exponentially weighted moving average (EWMA) filter to a window of historical measurements [51]. However, these methods do not provide a means for capturing the uncertainty associated with a particular estimate (i.e. the distribution Y introduced in the previous section). To solve this problem, we apply a one-dimensional extended Kalman filter (EKF) to the series of measurements. The mean of a one-dimensional EKF tracks the estimate of a EWMA filter (with variable weight) while the variance estimate captures the uncertainty involved in the estimate.

An EKF operates on infinite-range state variables, but our measurements z_t and state y_t are proportions falling in the range $[0, 1]$. To remedy this, we maintain a shadow state

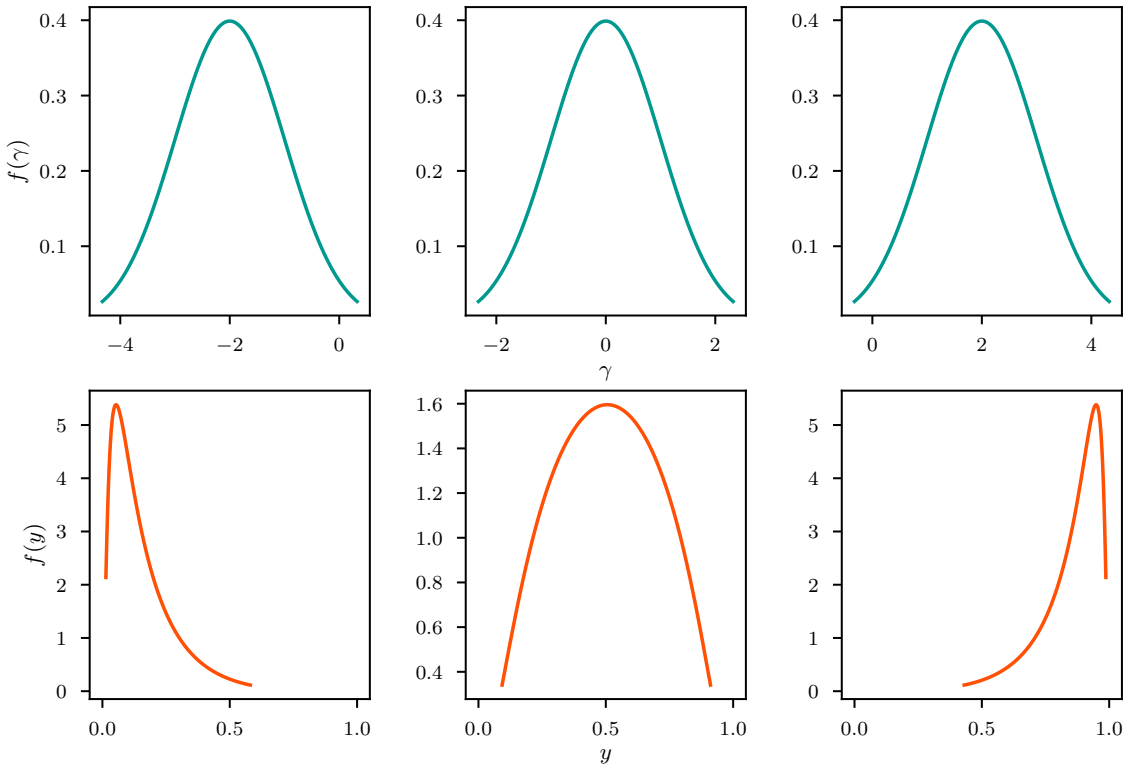

Figure 3.2: Distribution of packet reception ratio (PRR) state y and different distributions of shadow state $\gamma \sim \mathcal{N}(\mu, \sigma)$. Note that the support of y is in the range $(0, 1)$ since y represents a proportion, whereas the support of γ is $\mathbb{R}$.

estimate $\gamma_t \in \mathbb{R}$ based on the bijection

$$\begin{aligned} h(\gamma) &= y = \frac{1}{1+e^{-\gamma}} \in (0,1) \\ h^{-1}(y) &= \gamma = -\log\left(1/y - 1\right) \in \mathbb{R} \end{aligned} \tag{3.7}$$

The derivative of h is given by

$$h'(\gamma) = \frac{e^{-\gamma}}{(1+e^{-\gamma})^2} \tag{3.8}$$

At time t, we characterize the shadow state estimate $\gamma_t \in \mathbb{R}$ with mean μ_t and standard deviation σ_t. Measurement $z_t \in (0,1)$ is distributed according to $z_t = h(\gamma_t) + \delta_t$, where δ_t is zero-mean Gaussian noise with variance q_t. With this, we can write the EKF update step as

$$k_t = \frac{\sigma_{t-1} h'_t}{{h'_t}^2 \sigma_{t-1} + q_t} \tag{3.9a}$$

$$\mu_t = \mu_{t-1} + k_t \left(z_t - h\left(\mu_{t-1}\right)\right) \tag{3.9b}$$

$$\sigma_t = \left(1 - k_t h'_t\right) \sigma_{t-1}. \tag{3.9c}$$

Algorithm 3.1 requires that we be able to sample from distribution Y. To do this, we sample $\gamma \sim \mathcal{N}\left(\mu_t, \sigma_t\right)$ then compute $y = h(\gamma)$.

Figure 3.2 compares the distribution of the state y and shadow state γ for values of μ and σ. Observe that the support of y is in the range $(0,1)$, whereas the support of γ is in $\mathbb{R}$.

3.5 Evaluation

We evaluated the proposed system on a real-world mobile robotic platform to evaluate its effect on the robustness of network links. This evaluation has three main components. First, we demonstrate that AEC can achieve a target effective PRR set by the user. Second, we compare the performance of AEC to a fixed FEC scheme similar to QUIC [74]. Finally, we show that AEC can exploit increasing amounts of latency tolerance to improve effective PRR with less overhead.

3.5.1 Experimental Setup

Our test platform was a robotic team previously designed for urban reconnaissance [63]. We performed our evaluation with a single mobile robot and a fixed ground station. Both the robot and the ground station were outfitted with a 2.4 GHz Open-Mesh OM2P-HS Wi-Fi radio. We transmitted 1000-byte packets from the robot back to the ground station at a frequency of 10 Hz. The radios were configured to operate in IEEE 802.11 broadcast mode, which transmits at a 1 Mbps bitrate.

Our test environment consisted of both indoor and outdoor segments. The robot conducted a simulated exploration or mapping mission, moving through its environment without considering its network connectivity during motion planning. This allowed us to evaluate the performance of AEC on links of varying quality. We fixed the robot speed at 0.5 m/s throughout the experiment.

Recall that in our system, latency tolerance can be specified by the user on a per-packet basis in order to improve delivery reliability. We therefore marked each packet with a latency tolerance value of 0 ms, 400 ms, or 2400 ms. This corresponded to packets being members of groups of size 1 (0 ms), 5 (400 ms), or 25 (2400 ms). Note that we tested each of these values separately, so all traffic within a trial had the same latency tolerance.

For each group of packets transmitted during the experiments, we measured the *raw* and *effective* PRR of the transmitted network traffic. The raw PRR represents the PRR that would occur if the FEC system were not used; that is,

$$\text{Raw PRR} = \frac{\text{Data Received}}{\text{Data Transmitted}} \tag{3.10}$$

The effective PRR is the proportion of transmitted data packets that are either received directly or recovered from parity packets. Given the optimal decoding property of MDS erasure codes (see Property 2.3.1), this is given by

$$\text{Effective PRR} = \begin{cases} 1.0 & \text{if Data Received + Parity Received} \geq \text{Data Transmitted} \\ \text{Raw PRR} & \text{otherwise} \end{cases} \tag{3.11}$$

We also measured the overhead associated with the transmission of each group. We consider two types of overhead: transmitted overhead and received overhead. Transmitted

overhead is the amount of parity transmitted as a proportion of the amount of data. That is,

$$\text{Transmitted Overhead} = \frac{\text{Parity Transmitted}}{\text{Data Transmitted}} \tag{3.12}$$

Received overhead is the proportion of the number of excess packets received to the number of data packets transmitted. That is,

$$\text{Received Overhead} = \frac{\text{Data Received} + \text{Parity Received} - \text{Data Transmitted}}{\text{Data Transmitted}} \tag{3.13}$$

We associated each of these measurements with average received signal strength indicator (RSSI) of the packets received from the group. To obtain our overall results, we averaged all measurements associated with each RSSI value.

3.5.2 Results

In our first experiment, we fixed the latency tolerance at 400 ms and varied the target PRR, $1 - \epsilon$, where ϵ is the decoding failure probability explained in Section 3.4.1. We tested ϵ values of 0.01, 0.05, and 0.10, which correspond to target PRRs of 0.99, 0.95, and 0.90, respectively.

Figure 3.3 shows the raw and effective PRR achieved for each of these three targets. We first point out the behavior of the raw PRR across the range of link qualities: raw PRR is approximately 1.0 for high-quality links (-30 to -60 dBm), then decreases steadily through intermediate-quality links before reaching 0 at approximately -95 dBm. We observe that the effective PRR meets the target PRR through a wide range of link qualities before eventually breaking down at about -85 dBm. This effectively increases the range of these communication links by a significant amount, supporting a high rate of packet delivery even over intermediate- and low-quality links. In some cases, the effective PRR exceeds the target, which may occur when the raw PRR itself exceeds the target. This may also result from being overly conservative in the face of uncertainty about the packet loss rate estimate (see Section 3.4.2).

Figure 3.4 shows the overhead required to achieve the results of Figure 3.3. As expected, the amount of transmitted overhead (*top*) increases as link quality worsens. The explanation for this is simple: more parity packets are needed to mitigate the effect of worsening link qualities. Furthermore, the amount of transmitted overhead is greater for higher PRR targets. Because of the uncertainty involved in predicting packet delivery, achieving a

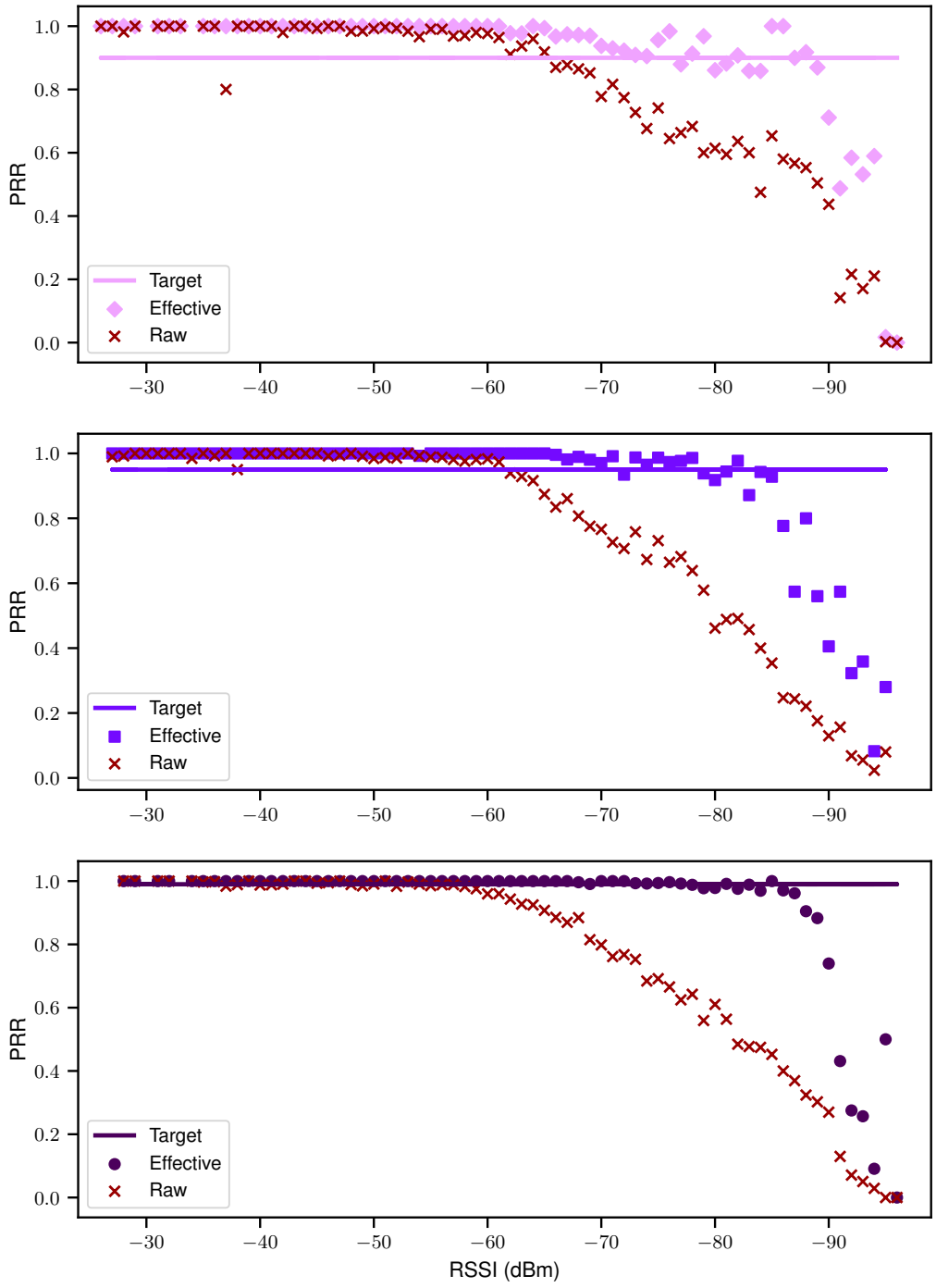

Figure 3.3: Effective packet reception ratio (PRR) achieved by Adaptive Erasure Coding (AEC) for varying target performance levels. The raw PRR declines as link quality decreases, but AEC is able to track a target effective PRR level. See Figure 3.4 for information on overhead levels required to achieve these improvements.

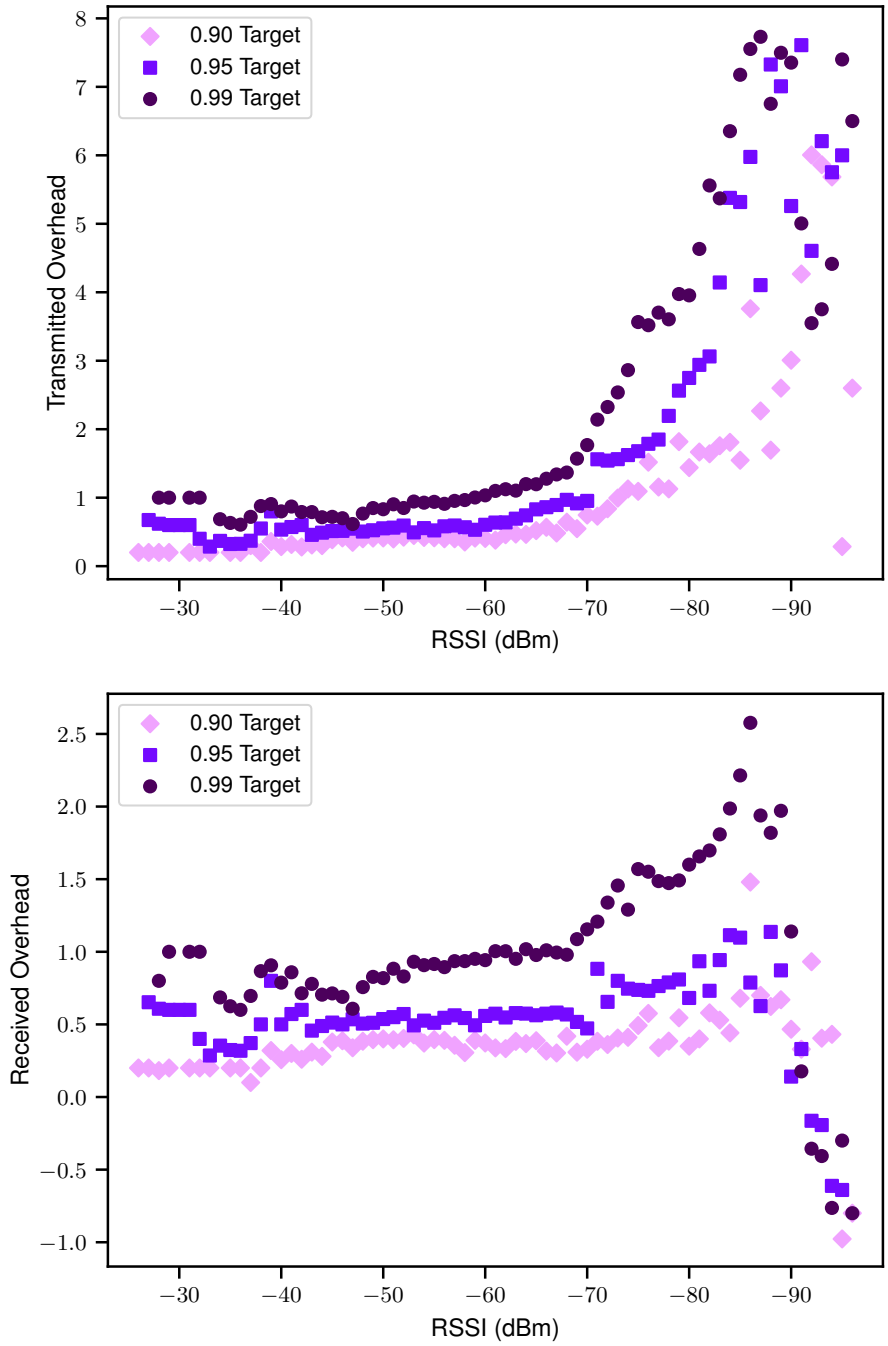

Figure 3.4: Measured overhead needed to achieve packet reception ratio (PRR) results of Figure 3.3. Transmitted overhead (amount of parity transmitted relative to amount of data transmitted) increases as link quality degrades and is greater for higher target PRR levels. Received overhead (number of packets received relative to amount of data transmitted) is also greater for the higher target PRR levels.

higher likelihood of successful group recovery requires transmitting more parity.

The bottom plot of Figure 3.4 shows the corresponding received overhead. Note that a positive received overhead value indicates that more packets were received than were needed. A negative value indicates that too few packets were received to recover all packets of a group successfully. Given the uncertainty involved in predicting packet delivery, the system will inevitably transmit excess parity in order to achieve the high PRR targets we specified. Observe that the received overhead is greater the higher the PRR target. We further point out that received overhead tends to increase as link quality deteriorates, which is a consequence of the greater degree of uncertainty associated with low-quality links. At the extreme low end of the link quality spectrum, received overhead decreases again as the system can no longer mitigate the packet loss and the effective PRR drops to zero.

Recall that a primary motivation for AEC in this chapter is the ability to exploit latency tolerance for improvements in communication reliability. To that end, we conducted an experiment in which we employed a constant PRR target ($\epsilon = 0.01$) and varied the latency tolerance of packets between 0 ms, 400 ms, and 2400 ms. With the traffic profile of transmitting packets at a fixed rate of 10 Hz, this corresponds to group sizes of 1 (0 ms), 5 (400 ms), and 25 (2400 ms).

Figure 3.5 shows the results of this experiment. The top plot is the raw and effective PRR for each level of latency tolerance, and Figure 3.6 gives a close-up view of the most interesting portion of the same data. We first observe that across all levels of latency tolerance, the previously discussed trends still hold. Namely, using the AEC method results in the effective PRR exceeding the raw PRR for intermediate- and low-quality links. An additional trend emerges in Figure 3.6: increasing the amount of latency tolerance results in a higher effective PRR in the low-quality link regime. To understand why this effect appears, recall that packet loss tends to occur in bursts [98]. Increasing the latency tolerance of packets in our system allows us to use packets transmitted over a wide time interval to recover from packet loss. This increases the effective temporal diversity of our system's transmissions and reduces susceptibility to burst loss.

The bottom two plots of Figure 3.5 show the transmitted and received overhead for traffic with varying amounts of latency tolerance. We observe that through much of the high- and intermediate-quality link range the traffic with higher latency tolerance is delivered with lower overhead than traffic with no latency tolerance. With higher amounts of latency tolerance, the size of each group of packets is larger. This results in a finer-grained capability for tuning the amount of redundancy relative to the amount of data. For example,

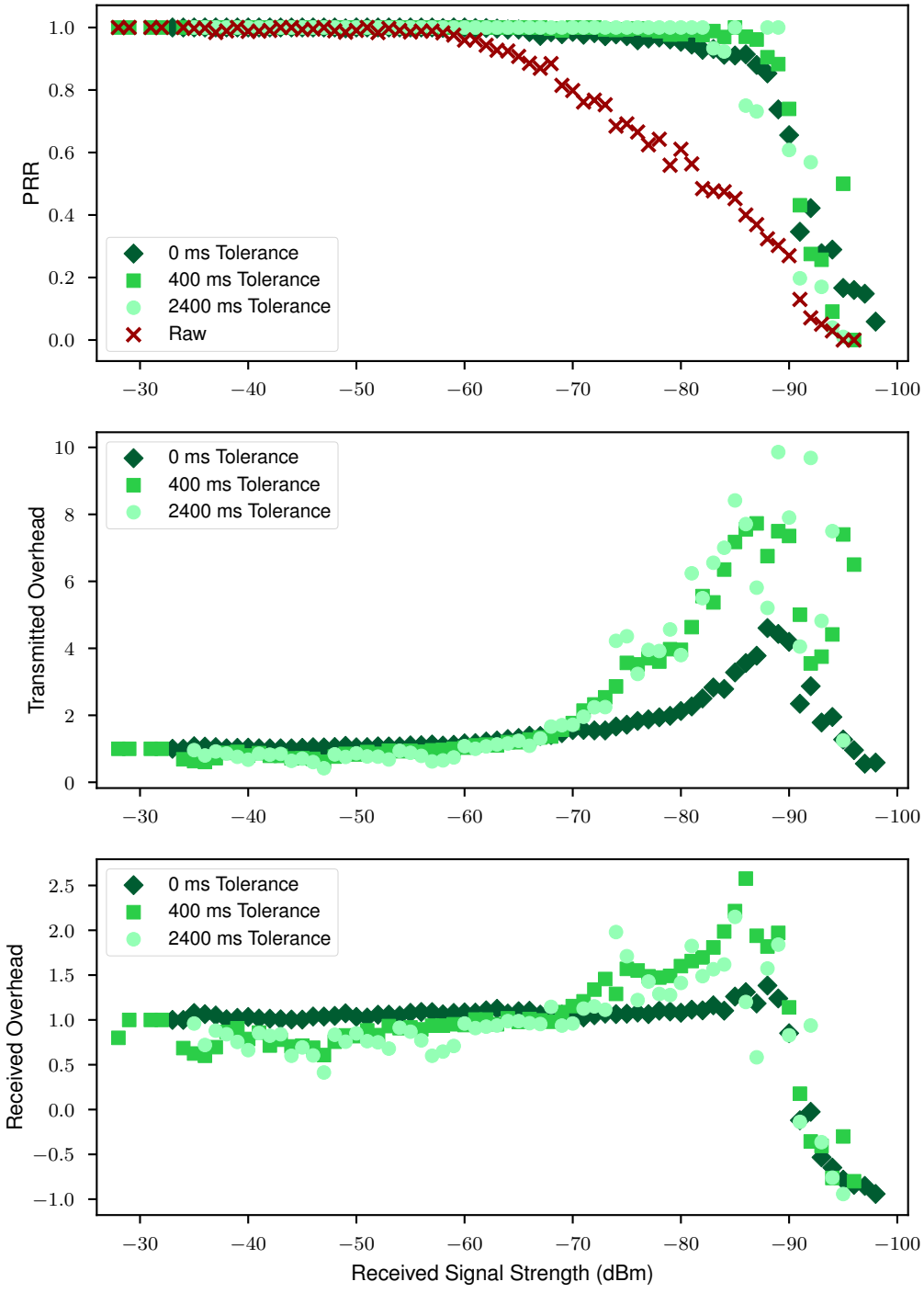

Figure 3.5: Packet reception ratio (PRR) and overhead results for an experiment with a fixed PRR target and variable latency tolerance. Increasing the amount of latency tolerance yields benefits in terms of effective PRR and overhead.

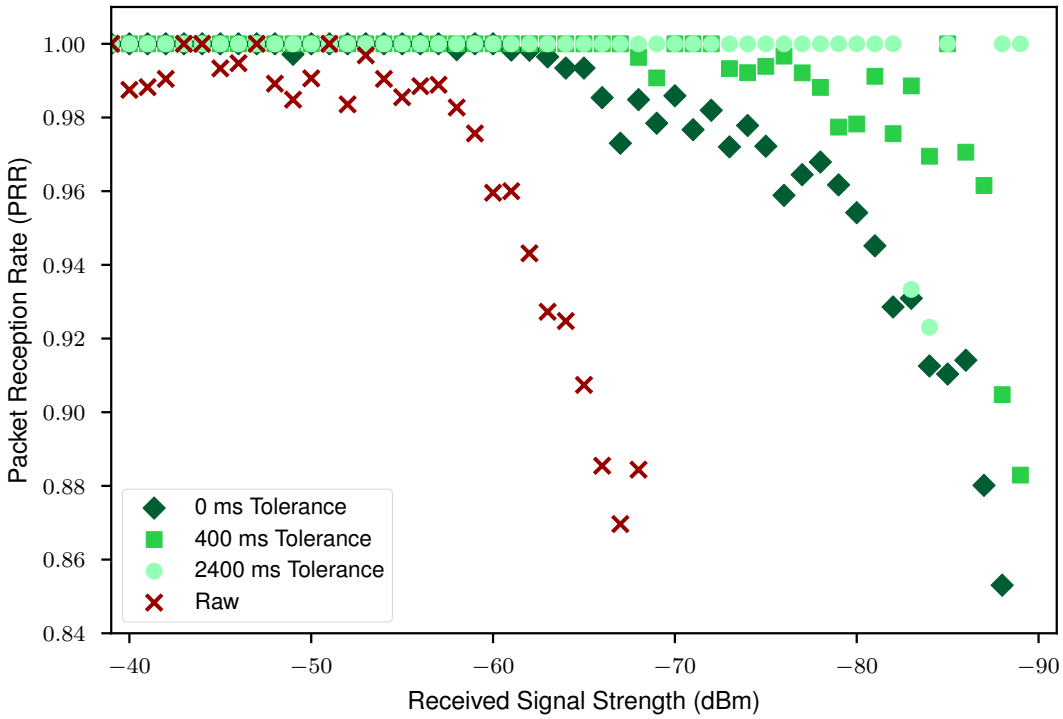

Figure 3.6: Close-up view of packet reception ratio (PRR) data from Figure 3.5. Increasing the amount of latency tolerance leads to a higher effective PRR.

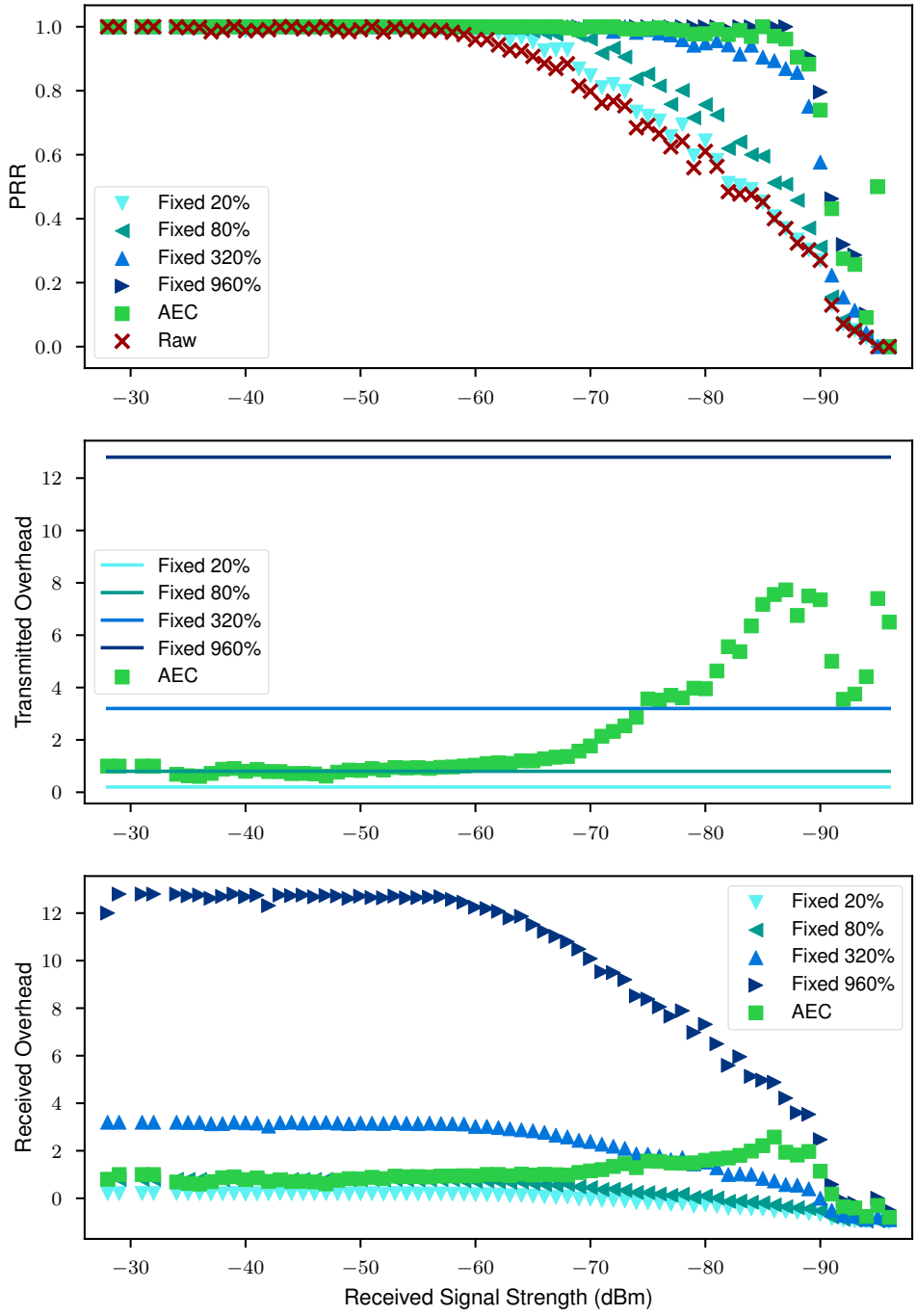

Figure 3.7: Comparison of Adaptive Erasure Coding (AEC) to a fixed erasure coding system similar to Google's Quick UDP Internet Connections (QUIC) [74]. The adaptive system yields effective PRR performance similar to the highest fixed method while using only a small fraction of the overhead.

adding a single redundant packet to a group of size 25 results in a transmitted overhead of only 4 percent, whereas the same amount of redundancy in a single-packet group yields a 100-percent overhead. Figure 3.5 indicates that this effect is reversed in the low-quality link regime. We hypothesize that this is due to the uncertainty involved in the calculating the encoding strength for low-quality links.

Finally, we conducted an experiment to highlight the benefit of the AEC method we propose compared to a fixed amount of erasure coding as used in systems such as Google's QUIC [74]. To do this, we set the latency tolerance to 400 ms (i.e. group size 5) and fixed the amount of redundancy to either 1 (20 percent), 4 (80 percent), 16 (320 percent), or 64 (960 percent). We compared these fixed-redundancy results to our adaptive results targeting an effective PRR of 99 percent (i.e. $\epsilon = 0.01$). Figure 3.7 shows the results of this experiment. We observe that across a wide range of link qualities, the adaptive system outperforms the fixed system in terms of PRR for all but the highest level of redundancy (960 percent). It does this while maintaining significantly lower overhead through the range of link qualities. These results underscore the importance of adapting our system based on measured link quality.

3.6 Summary

In this chapter, we introduced AEC, an adaptive erasure coding method designed for mobile robotic networks. AEC exploits application latency tolerance by grouping packets together and encoding, thus reducing the entire group's resiliency to loss. AEC is adaptive, varying encoding strength based on the estimated quality of the communication channel. We provided a technique for computing the minimum amount of redundancy needed to reduce the probability of unrecoverable packet loss below some threshold value. To enable this method, we described a technique to estimate the quality of the communication channel based on historical measurements of packet loss. We showed that the resulting system is capable of achieving target PRRs across a range of link qualities and can successfully exploit latency tolerance for improvements in packet delivery performance.

CHAPTER 4

Transporting Bandwidth-Intensive Traffic[1]

4.1 Introduction

In Chapter 3, we introduced a system, Adaptive Erasure Coding (AEC), through which an application can exchange latency tolerance for improved packet delivery in mobile ad-hoc networks (MANETs). AEC groups small packets together and applies redundancy to the group so that it can use the redundancy to recover packets from the group that are lost during transmission. This chapter builds on the ideas of AEC to solve a different problem: transporting a single large packet (e.g. a high-resolution image) over lossy, time-varying links in a MANET.

Modern network stacks require that data be received error-free before being passed from the physical layer (PHY) to upper layers. The PHY verifies the integrity of the data using a cyclic redundancy check (CRC), discarding corrupted data (see Section 3.2). In light of this, higher layers fragment large packets into smaller chunks before passing them to the PHY for transmission. That way, even if some of the fragments are dropped, others may still be received successfully. However, a typical best-effort transport protocol (e.g. UDP) will still fail to deliver the complete packet if any of the fragments are lost.

This is a primary motivation for reliable delivery mechanism such as TCP. TCP retransmits lost fragments until they are received. This feature is a primary reason why TCP is the dominate transport protocol for the modern Internet. However, as we have introduced previously in this dissertation, MANETs have different characteristics than the general Internet. Typical routes over the Internet consist of managed wired networks, with a possible single wireless link to reach the end-user. The primary cause of packet loss in such channels occurs when too many transmissions occur for the channel to support, a condition known

[1] Adapted from Marcotte et al. [55]

as congestion. TCP introduced congestion control to combat this issue, interpreting packet loss as a sign of congestion on the channel and responding by waiting for a time interval before continuing transmission.

Interpreting packet loss as a sign of congestion can be problematic in MANETs, where significant environmental factors play a role in loss. Robots operating in complex environments are subject to multipath effects and shadowing, and path loss results when significant distances separate robots [31]. These environmental factors occur independently of congestion, and backing off in the face of them is not a suitable response. When TCP encounters significant packet loss such as is seen in MANETs, its back-off mechanism can result in long stalls, during which it transmits no data. As we will show in Section 4.5, this occurs even for variants of TCP that are intended for wireless networks (e.g. TCP Westwood [16]).

In this chapter, we show a partially-reliable protocol built on top of UDP that uses forward error correction (FEC) to improve the likelihood of packet delivery. We call this protocol High Bandwidth Adaptive Erasure Coding (HBAEC) since it is an adaptation of AEC for bandwidth-intensive traffic. HBAEC fragments large packets into shards, produces additional redundant shards, and then uses that redundancy to recover original data shards that have been lost. This type of system requires a different choice of erasure code compared to the previous chapter. The maximum distance separable (MDS) erasure codes of the previous chapter have a practical upper limit of about $n = 256$, where n is the total number of encoded symbols. Beyond this upper limit, decoding may be too slow for real-time usage [72]. In a network with a maximum transmission unit (MTU) of 1500 bytes, this corresponds to less than 400 KB of total data, including redundancy. We therefore use fountain codes, which have faster encoding and decoding than MDS erasure codes (see Section 2.3 for more information about fountain codes and their relation to MDS erasure codes). In exchange for this speedup, however, fountain codes give up the optimal decoding property of MDS erasure codes (Property 2.3.1). Rather, the probability of decoding success is non-zero when at least k encoded symbols are received and increases with each additional received symbol (Property 2.3.2). In Section 4.2, we generalize the encoding strength calculation of Section 3.4.1 to account for this probabilistic decoding characteristic.

When dealing with large quantities of data, it is critical that the estimation process be accurate to avoid wasting bandwidth transmitting unnecessary redundancy. In the previous chapter, we used an extended Kalman filter (EKF) to estimate the distribution of instantaneous packet loss based on recent packet loss measurements. In practice, however, this

distribution is often multi-modal and non-Gaussian. We therefore augment the previous approach and use kernel density estimation (KDE), considering a fixed number of historic packet loss measurements to be samples drawn from the current distribution. We describe this estimation technique in Section 4.3 and show that its estimates have higher relative likelihood compared to the approach of Section 3.4.2.

In evaluating network protocols for robotic systems, we encounter the common dilemma of whether to test in the real world or in simulation. Real-world testing (e.g. Section 3.5) has the highest possible fidelity, but it is difficult to carry out repeatable experiments. Simulations are repeatable and cheap to execute, but existing network simulators (e.g. [71]) have limited support for MANETs [49]. In Section 4.5, we will introduce a hybrid approach known as trace-driven simulation [36] that infuses real-world data into simulation. Namely a log of real-world network data is input to a network emulator [38], resulting in a high-fidelity, repeatable evaluation.

The key insight in this chapter is that transporting bandwidth-intensive traffic over a lossy network requires that packet loss be mitigated proactively. Packet loss in Internet traffic is largely caused by congestion, but multi-robot systems experience significant packet loss due to environmental conditions. Handling this packet loss requires the use of redundancy, the amount of which must be carefully controlled, which is the role of HBAEC.

Our technical contributions in this chapter are:

- A method to optimize the amount of redundancy to transmit when erasure codes with probabilistic (sub-optimal) decoding characteristics are used for encoding (Section 4.2),
- A technique that uses KDE to estimate the distribution of instantaneous packet loss based on historical packet loss measurements (Section 4.3), and
- A repeatable evaluation that uses packet traces collected from a real-world robotic network to drive simulations (Section 4.5).

4.2 Calculating Encoding Strength with Fountain Codes

In Section 3.4, we developed a method that minimizes the amount of redundancy to transmit such that the probability of decoding failure is sufficiently small. The method applies to MDS erasure codes, which have the following optimal decoding property:

Property 2.3.1 (Decoding of MDS erasure codes)**.** *Any k symbols of an n-symbol code word produced by an MDS erasure code can be used to decode the original k-symbol message.*

However, fountain codes give up this optimal decoding property in exchange for improved encoding and decoding efficiency, replacing it with the following probabilistic decoding property:

Property 2.3.2 (Decoding of fountain codes)**.** *With high probability, any k' symbols of an n-symbol code word produced by a fountain code can decode the original k-symbol message, for $k' = k+o$, where $o = 0, 1, \ldots$ is a small amount of overhead. The probability of decoding failure decreases with increased o.*

The encoding strength calculation method of Section 3.4.1 assumed that data was encoded with a MDS erasure code such that Property 2.3.1 held. In this section, we generalize the calculation method to erasure codes with probabilistic decoding properties, such as fountain codes.

The probability of decoding failure for fountain codes is a function of the overhead o, parameterized by the number of data symbols k and the erasure probability p_e (or, equivalently, the probability of successful reception y). Let $O_{k,y}$ be a random variable distributed according to the probability of decoding *success* when k data symbols are encoded and transmitted over a packet erasure channel (PEC) with PRR y. Note that, in general, the distribution $O_{k,y}$ is not available in closed form and must be obtained through empirical means. Figure 4.1 shows the distribution $O_{k,y}$ for online codes [56] for a range of k values.

Let $K'_{k,y} = k + O_{k,y}$ be a random variable distributed according to $O_{k,y}$ but shifted by constant k. K' represents the number of encoded symbols that must be received in order to successfully decode the message. We can then write the objective function for calculating encoding strength with fountain codes (c.f. Equation (3.1)):

$$\begin{aligned} \text{minimize} \quad & n \\ \text{s.t} \quad & \mathbb{P}\left(X_n < K'_{k,y}\right) < \epsilon \end{aligned} \tag{4.1}$$

To transform Equation (4.1) into a workable form, we first define random variable $W_{n,k,y} = X_n - K'_{k,y}$. Because X_n and $K'_{k,y}$ are independent random variables, the probability mass function (PMF) of $W_{n,k,y}$ is the convolution of the PMFs of X_n and $K'_{k,y}$,

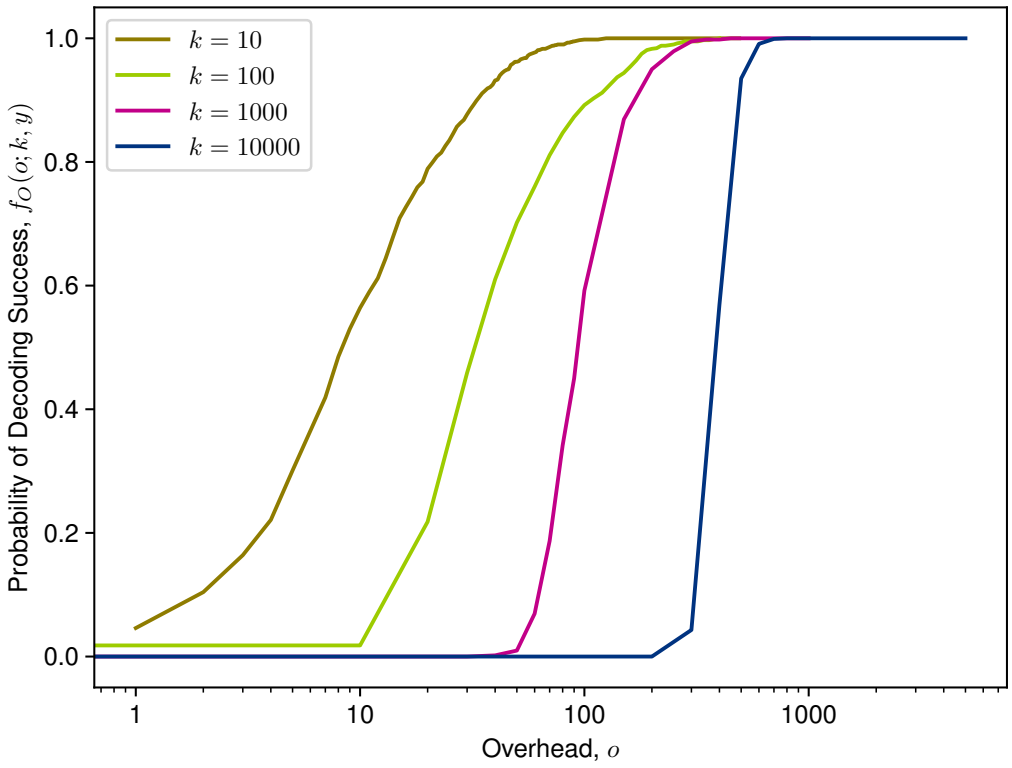

Figure 4.1: Decoding probability at varying levels of overhead for online codes [56]. The decoding probability of online codes is parameterized by the number of data symbols, k. For online codes, the decoding probability is independent of the packet reception ratio (PRR) y.

that is,

$$P_W(w;n,k,y) = \sum_{k'=k}^{\infty} P_X(w+k';n) P_{K'}(k';k,y). \tag{4.2}$$

P_W follows a compound distribution where $f_Y(y)$ are the weights for components $P_W(w;n,k,y)$. We can then re-write the PMF as

$$P_W(w;n,k) = \int_0^1 P_W(w;n,k,y) f_Y(y). \tag{4.3}$$

The cumulative distribution function (CDF) of $W_{n,k,y}$ is given by

$$F_W(w;n,k) = \mathbb{P}(W_{n,k,y} \leq w) = \sum_{v=-\infty}^{w} P_W(v;n,k). \tag{4.4}$$

Analyzing the form of these distributions, we recall from Chapter 3 that P_X, the PMF of the distribution of received packets over a PEC with uncertain erasure probability, is given by Equation (3.2):

$$P_X(x;n) = \mathbb{P}(X_n = x) = \int_0^1 P_X(x;n,y) f_Y(y) dy, \tag{3.2}$$

where $P_X(x;n,y)$ is the PMF of binomial distribution $\mathcal{B}(n,y)$ and $f_y(y)$ is the probability density function (PDF) of the estimated PRR distribution. Further, the decoding probability $P_{K'}$ can be obtained offline through empirical means for a given fountain code. Though we have parameterized $P_{K'}$ by the PRR, y, the decoding distributions of many fountain codes are independent of y; that is, $P_{K'}(k';k,y) = P_{K'}(k';k)$ for all y. From a practical standpoint, this independence simplifies the task of empirically obtaining the decoding distributions. This is the case with the fountain codes we use, online codes [56]; Figure 4.1 shows their decoding distributions.

From here, we can re-write Equation (4.1) in terms of the CDF of Equation (4.4):

$$\begin{aligned} \text{minimize} \quad & n \\ \text{s.t} \quad & F_W(-1;n,k) < \epsilon \end{aligned}. \tag{4.5}$$

Note that we are interested in the value of the CDF at -1, which corresponds to a deficit of packets being received.

Just as in the previous chapter, no closed-form solution exists for Equation (4.5). We

Algorithm 4.1 Calculating Optimal Encoding Strength for Fountain Codes

function CALCULATEENCODINGSTRENGTH(k, Y, K'_k, ϵ)
  Initialize $n^* \leftarrow n_{\max}$
  Initialize left $\leftarrow k$, right $\leftarrow n_{\max}$
  while left $\leq$ right **do**
    $n \leftarrow \lfloor \frac{left+right}{2} \rfloor$
    for S iterations **do**
      Sample $y \sim Y$
      Sample $k' \sim K'_{k,y}$
      Sample $x \sim \mathcal{B}(n, y)$
      Update $\hat{F}_W(x - k'; n)$
    if $\hat{F}_W(-1; n) < \epsilon$ **then**
      $n^* \leftarrow n$
      right $\leftarrow n - 1$
    else
      left $\leftarrow n + 1$
  return n^*

therefore generalize the empirical method of Algorithm 3.1 to estimate the distribution of $W_{n,k}$ and compute the optimal encoding strength. We detail this method in Algorithm 4.1, which takes in the number of data symbols k, the estimated PRR distribution Y, the set of possible decoding probability distributions $K'_k = \bigcup_{y \in (0,1)} K'_{k,y}$, and the acceptable failure probability ϵ.

4.3 Estimating the Packet Loss Rate Distribution

In the previous chapter, we estimated the instantaneous packet loss rate by filtering historical packet loss rate measurements. This also quantified the uncertainty involved in a particular estimate, thus allowing us to solve for the optimal encoding strength. The EKF approach assumed that the distribution underlying the packet loss rate measurements was Gaussian, and therefore unimodal. In practice, however, the distribution of packet loss may be multimodal, with bursts of significant loss interspersed among periods of zero loss.

In this section, we describe an improved technique to estimate the packet loss rate distribution. Rather than filtering historical measurements, we treat recent measurements as samples from the current loss rate distribution. We use these samples to estimate the underlying distribution through kernel density estimation.

This type of approach assumes that the packet loss rate distribution remains constant

over the time interval during which measurements are taken. To better understand the validity of this assumption, recall the three key physical factors contributing to packet loss in a wireless network (see Section 3.1 and [31, 59]): *path loss*, *small-scale fading*, and *shadowing*. Path loss varies over a period of seconds to tens of seconds for mobile ground robots as they traverse their environment [31]. Small-scale fading (multipath) and shadowing occur over much smaller time scales (e.g. microseconds to milliseconds). Taking measurements at a rate of approximately 1 Hz, the path loss component is static across a small sample window. The higher-frequency effects occur at a higher resolution than our measurements, which is the primary source of uncertainty in our estimates of the packet loss rate. However, our measurements can still provide a coarse-grained estimate of the instantaneous packet loss rate, and we will show that our KDE technique improves upon the EKF method of the previous chapter.

Just as in Section 3.4.2, our target distribution is a probability and thus has bounded support (i.e. $[0, 1]$), but KDE assumes the distribution is over all reals. We therefore apply the same technique as before, using a bijection to transform measurements and estimate a shadow distribution with unbounded support. Specifically, we use the bijection

$$\begin{aligned} h(\gamma) = y &= \frac{1}{1+e^{-\gamma}} \in (0,1) \\ h^{-1}(y) = \gamma &= -\log\left(1/y - 1\right) \in \mathbb{R} \end{aligned} . \tag{3.7}$$

Let y_τ be a measurement of packet loss during time interval τ. At time t, we consider the w most recent measurements as samples from the packet loss rate distribution. We use the mapped value of these measurements to estimate the distribution of shadow loss rate variable Γ_t as

$$\hat{f}_{\Gamma_t}(\gamma; b, w) = \frac{1}{bw} \sum_{i=t-w}^{t-1} K\left(\frac{\gamma - h(y_i)}{b}\right), \tag{4.6}$$

where b is the kernel bandwidth. We use a Gaussian kernel, that is,

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}, \tag{4.7}$$

though the choice of kernel is of minimal consequence [83]. The bandwidth b is known to have significant effect on the accuracy of KDE. Because the distribution changes over time, we use automatic kernel bandwidth selection. Specifically, we use the method of Silverman [83], which computes an approximately optimal bandwidth (i.e. the one that minimizes the

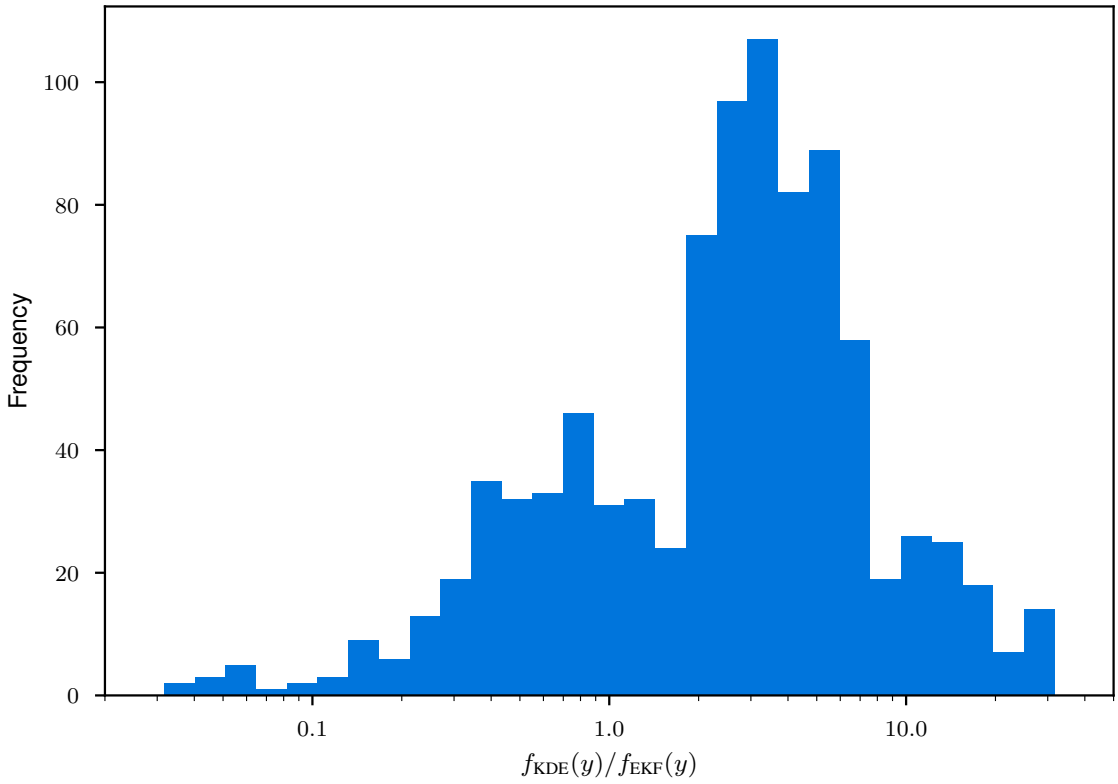

Figure 4.2: Frequency of relative likelihood of the kernel density estimation (KDE) and extended Kalman filter (EKF) distributions given samples collected from a real-world robotic network (see Section 4.5.1). We observe that for 82 percent of the samples, the KDE-based approach yielded a distribution of higher likelihood than the EKF-based approach of Section 3.4.2.

mean integrated square error) based on the collected samples. This bandwidth b is given by

$$b = 0.9 \times \min\left(\sigma, \frac{\text{IQR}}{1.34}\right) \times w^{-1/5}, \tag{4.8}$$

where σ and IQR are the standard deviation and inter-quartile range of the samples, respectively.

We evaluated the performance of this KDE-based method in comparison to the EKF-based method of the previous chapter. To do so, we applied each estimator to packet loss time series data collected from a real-world robotic network (see Section 4.5.1). This data consists of a sequence of packet loss measurements taken at a rate of 1 Hz. To get an estimated PRR distribution at a given time t, we applied each estimator to the sequence of

measurements leading up to that time, yielding distributions $Y_{t_{\text{KDE}}}(y)$ and $Y_{t_{\text{EKF}}}(y)$. We then evaluated the relative likelihood of each distribution given the actual observation at time t, y_t, that is,

$$\frac{f_{t_{\text{KDE}}}(y_t)}{f_{t_{\text{EKF}}}(y_t)}. \tag{4.9}$$

Figure 4.2 shows a histogram of the relative likelihoods of the distributions across all time steps of the collected data. A relative likelihood value greater than 1.0 indicates that the KDE distribution is more likely than the EKF distribution for a given data point. We observe that for 82 percent of samples, the KDE distribution had higher likelihood than the EKF distribution. This indicates that the KDE method yields a better estimate of the instantaneous PRR distribution than the EKF method.

4.4 Transporting Bandwidth-Intensive Traffic with AEC

In order to facilitate comparison of AEC to TCP, we developed a simple transport protocol around it. We call the resultant system High Bandwidth Adaptive Erasure Coding (HBAEC). HBAEC breaks each packet to be transmitted into fragments, encodes those fragments to produce additional redundant fragments, then transmits those fragments as datagrams on top of UDP.

Because UDP does not have any form of congestion control, HBAEC implements a mechanism that attempts to minimize self-inflicted congestion on the network. Recall that Algorithm 4.1 estimates the likelihood of decoding failure for candidate levels of encoding strength and selects the minimum encoding strength necessary to reduce the likelihood of decoding failure below a threshold ϵ. When transmitting large quantities of data over high-loss network links, setting a low fixed value for ϵ would result in prohibitively large amounts of redundancy. Most of the redundancy would be excessive, taxing the network unnecessarily. In light of this, HBAEC begins transmitting packets with a large ϵ value, producing a small amount of redundancy. HBAEC then waits for an acknowledgment (ACK) from the receiver. In the absence of an ACK, the transmitting HBAEC module decreases ϵ and repeats the calculation of Algorithm 4.1, sending additional redundancy fragments as necessary. This process continues until either an ACK is received or a lower limit on ϵ is reached.

4.5 Evaluation

4.5.1 Trace-Based Network Emulation

Evaluating the performance of network algorithms in a multi-robot system is challenging. Ideally, all evaluation might take place in the real world so as to avoid any compromise on fidelity. However, real-world evaluation is expensive; building, maintaining, and deploying a fleet of mobile robots takes time and resources. Real-world tests are also difficult to reproduce and control, in particular because network conditions vary over time. For example, consider a set of tests conducted in an office building. During the day, the level of interference is high as occupants of the building use the wireless spectrum, but the interference level drops during the night as occupants leave the building. Such variable conditions can affect results if they are not controlled.

One proposed solution for easing the burden of real-world testing is shared testbeds. For example, Doddavenkatappa et al. [22] and Appavoo et al. [1] implemented Indriya, a set of more than 100 networked wireless sensors distributed across three floors of a university building. Chiueh et al. [18] proposed MiNT, a platform consisting of iRobot Roomba robots equipped with IEEE 802.11 network radios. However, neither of these testbeds were viable for testing the systems in this dissertation: Indriya's nodes are stationary, and MiNT is no longer available for use.

A common alternative to real-world testing is simulation. For example, Kudelski et al. [49] implemented a framework for integrating common robotic simulators into the `ns3` network simulator [71]. Such simulated approaches are cost-effective and scalable, with large simulations possible on consumer-grade hardware. Simulation is also highly repeatable, allowing for fair tests of competing methods or meaningful sweeps across algorithm parameter settings. However, fidelity is an ever-present issue in simulation [37]. `ns3` supports MANET implementations with arbitrary node mobility, but it has little support for building realistic simulated environments. Interaction with the environment (e.g. through shadowing or multipath effects, see Section 3.1) is a primary challenge in the networks of multi-robot systems, so this limitation prevents us from testing our systems in `ns3` or related simulators.

In this chapter, we employ a hybrid evaluation technique that has seen little use in the robotics community: trace-based network emulation. An emulator combines software and hardware to mimic the behavior of a network, with some components implemented in the real world and others being simulated [47]. Trace-based network emulation in particular

involves collecting traces of real-world network statistics, then using that data to drive a network emulator [61]. Such an approach retains advantages from both real-world testing and simulation while minimizing the drawbacks.

We collected network traces from a real-world mobile robotic system operating in indoor and outdoor environments. The trace collection runs consisted of a single point-to-point link between one mobile node and one stationary node (an "operator station"), both equipped with an OpenMesh OM2P-HS router. The routers run a stock version of OpenWRT 15.05, which implements IEEE 802.11s [39].

The network traces consist of the instantaneous packet loss rate and transmission bitrate measured on the link at a frequency of 1 Hz. We used the `iperf3` tool [91] to measure packet loss for MTU-sized (i.e. 1500 bytes) UDP packets transmitted at a target bandwidth of 10 Mbps. We obtained the bitrate data by querying the operating system with netlink sockets.

We conducted two trace collection tests, one in an indoor office environment and one in an outdoor environment with line-of-sight obstructions, including trees, hills, and small buildings. Both environments were slightly larger than the effective range of the routers, and the mobile node operated without consideration for the status of the network. As a result, the traces contain a wide range of network conditions, including some outages. In total, these traces comprise nearly 50 minutes of real-world network data. We visualize the packet loss rates of the network traces in Figure 4.3.

After collecting the traces, we used them to drive the `netem` network emulator [38], a utility that manipulates Linux traffic control facilities to emulate network conditions such as delay, packet loss, or bitrate. Given a sequence of network statistics, `netem` can emulate the conditions of the real-world wireless network on the local network of a single host.

It is important to note that trace-based network emulation does not provide a generative model as a simulator does. Rather, it provides a means to measure the conditions on a real-world network during the trace collection phase and then repeat those conditions later in the emulator. In other words, trace-based network emulation provides a way of repeating a real-world experiment in a controlled fashion, allowing for a fair comparison of different algorithms.

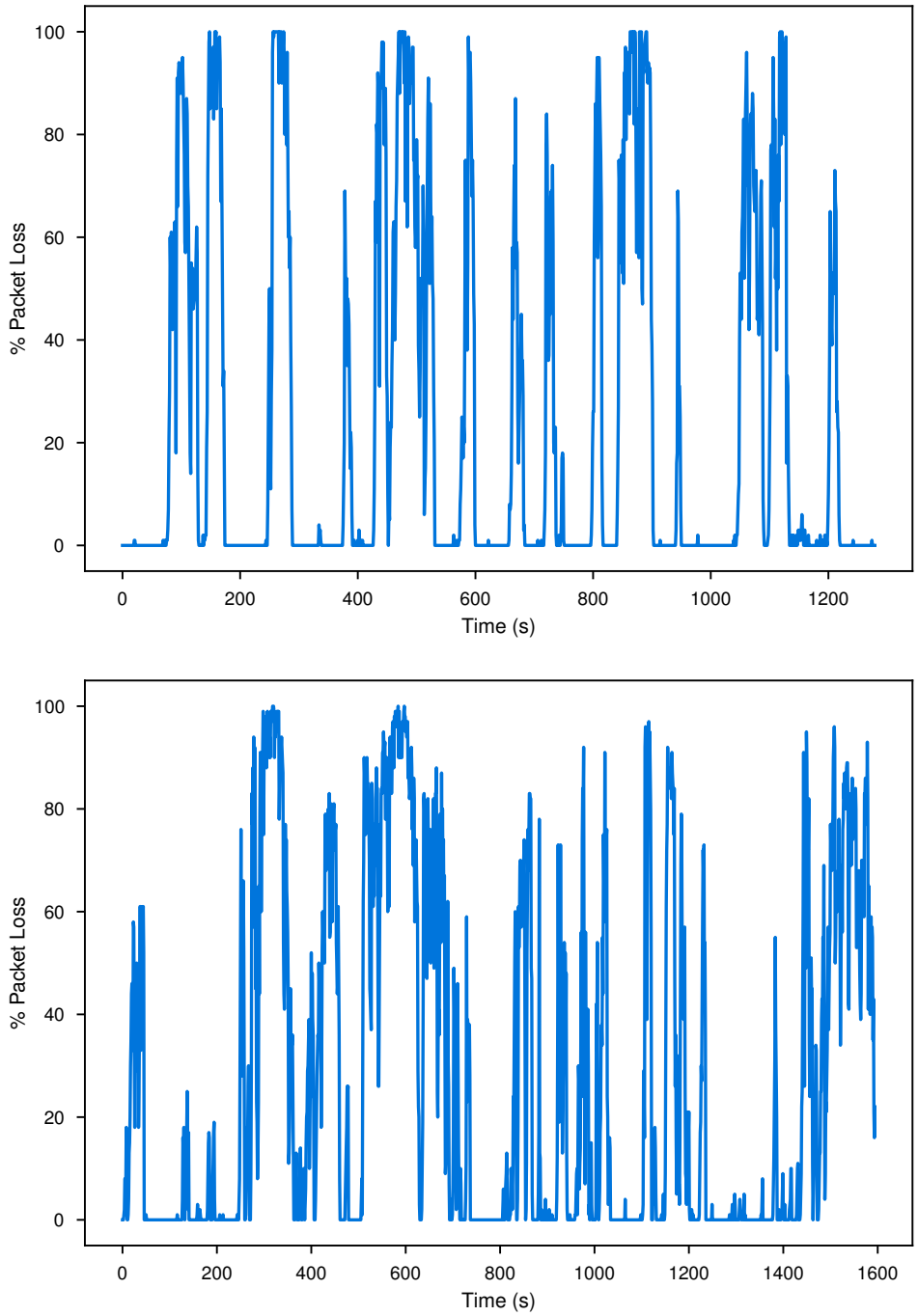

Figure 4.3: Time-series data of packet loss rates taken from mobile robotic network traces in indoor (*top*) and outdoor (*bottom*) environments. Note the frequent and rapid fluctuations in loss rates as well as the outages and near outages that occur. In total, the two traces comprise nearly 50 minutes of real-world network data.

4.5.2 Continual Image Delivery Task

In the continual image delivery task, a source node sends a series of images to a destination node. On the source node, the images are made available to the transmitting application at a rate of 1 Hz. The transmitting application then attempts to deliver each image with one of four underlying transport protocols: HBAEC (Section 4.4), UDP, TCP, or UDP-based Data Transfer Protocol (UDT) [34]. The goal of the task is to deliver as many of the images as possible to the destination node.

As discussed in Section 4.1, TCP is the standard protocol used to transmit data for this type of task in a traditional network. By design, TCP ensures reliable delivery by retransmitting segments until they are successfully acknowledged. When faced with the significant packet loss of mobile robotic networks, TCP's congestion control mechanisms cause extensive delays in delivery. TCP may deliver a 1 MB image in 100 milliseconds or less over a high-quality link, but that same image may take TCP as long as 10-30 seconds to deliver over a highly lossy link. In our implementation, the new images made available to the transmitting application during that process are discarded. Once the delivery finally occurs, transmission will begin again with the most recently produced image.

In our evaluation, we set the congestion control mechanism of TCP to be Westwood, an algorithm optimized for communicating over wireless networks [16]. This differs from the default TCP congestion control mechanism for Linux, TCP Cubic [35]. In addition to TCP, we also compare against UDT [34], which is a high-performance data transfer protocol built on top of UDP and designed for use over high-speed wide area networks.

Note that although we transmit images in this case, the task is agnostic to the type of data transported. Because the type of data transported is opaque to UDP and TCP, we follow the same technique in HBAEC. A more advanced system for transporting images or video with FEC might exploit properties of particular media formats when designing the coding scheme [43] or compression method [70].

4.5.3 Results

We conducted a series of trials of the continual image delivery task using the trace-based network emulation technique described in Section 4.5.1. We varied the size of the transmitted images between 200 KB and 2 MB. Transporting these images over a lossy network is highly challenging. The peak empirical bandwidth of our hardware is approximately 80 Mbps, so transporting the 2 MB image takes approximately 200 milliseconds even un-

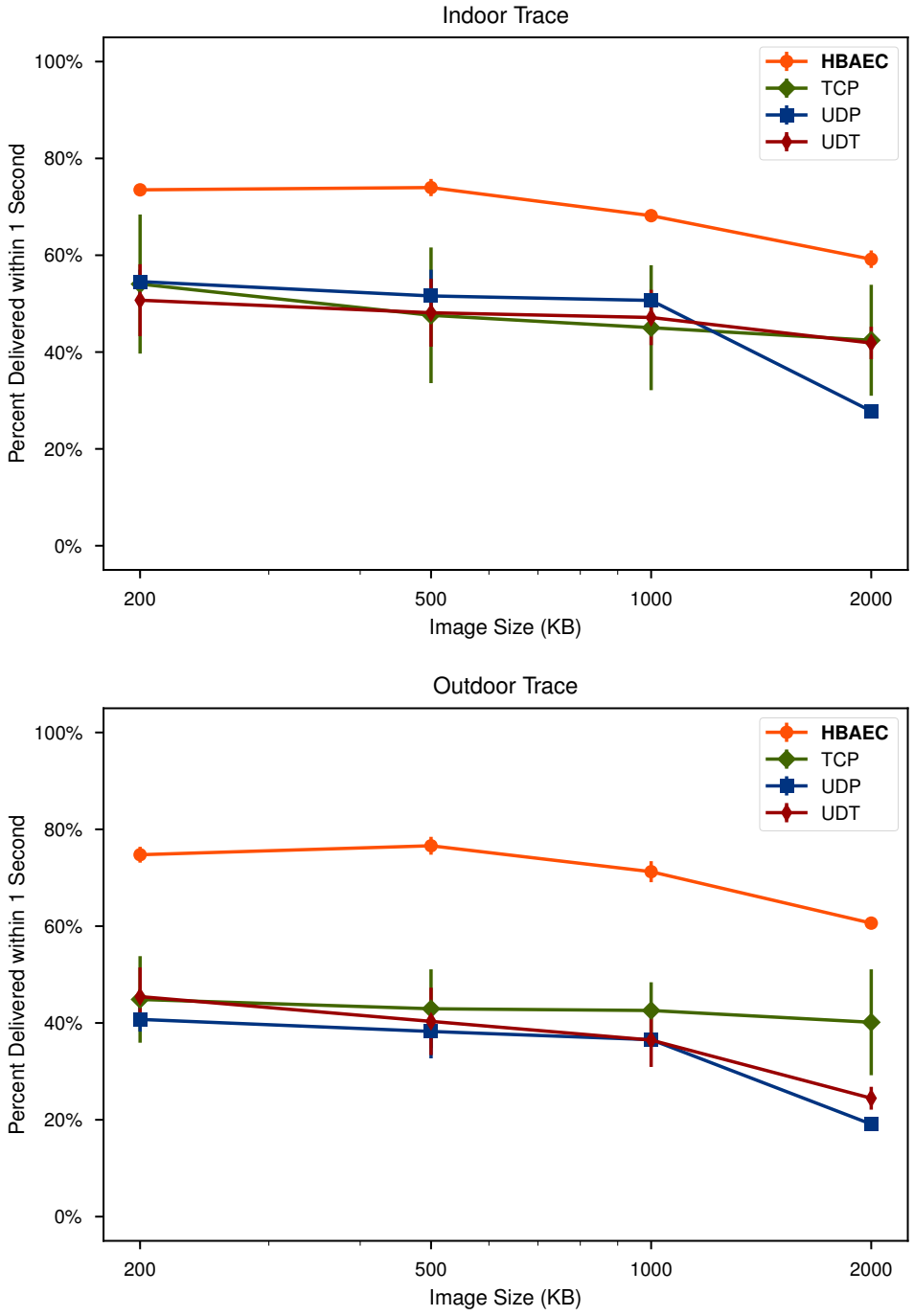

Figure 4.4: Percent of images delivered during trace-driven network emulation for High Bandwidth Adaptive Erasure Coding (HBAEC), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and UDT. HBAEC delivers a higher percentage of images compared to the other protocols.

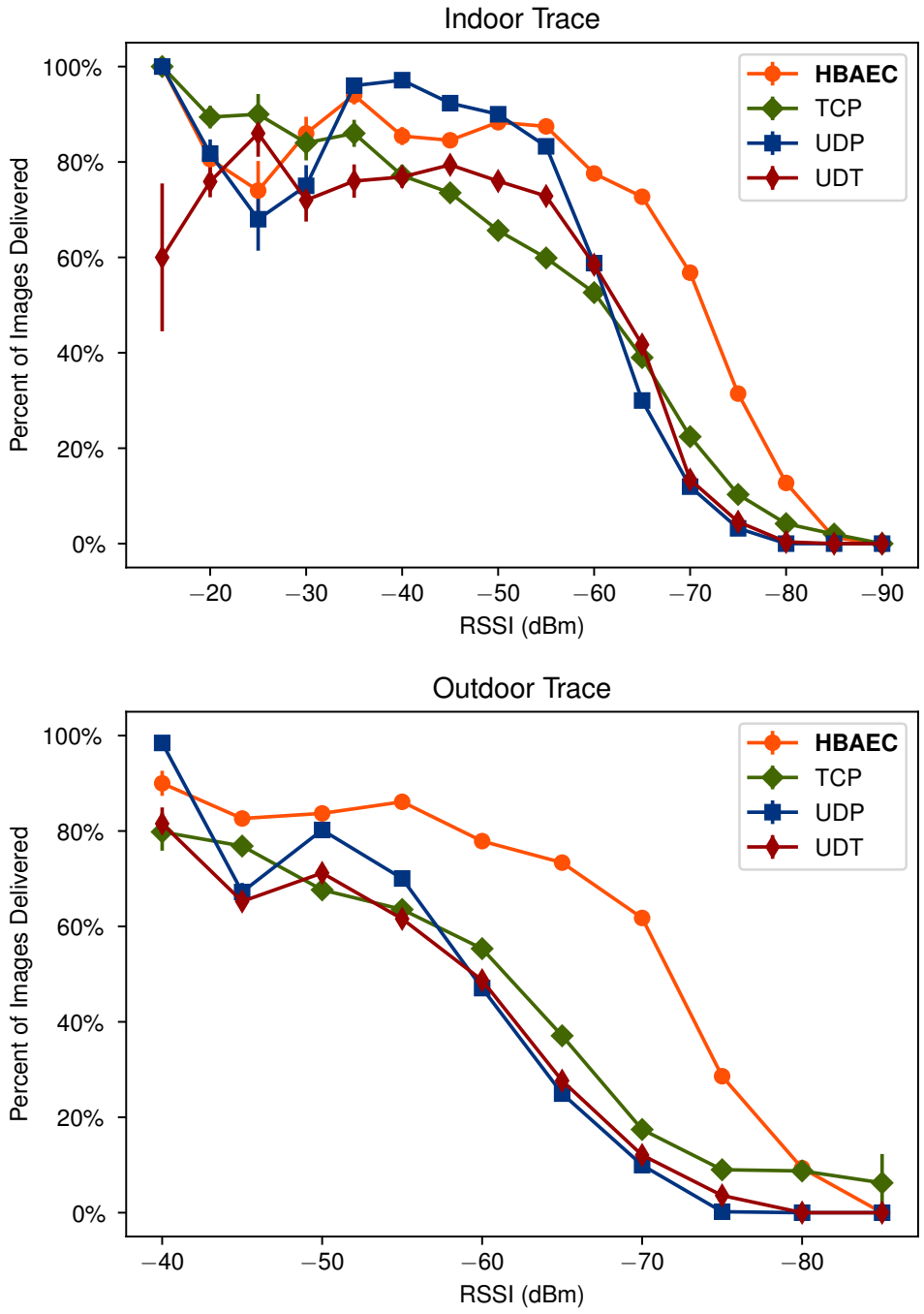

Figure 4.5: Image delivery performance of Figure 4.4 broken down by link quality. High Bandwidth Adaptive Erasure Coding (HBAEC) performs similarly to Transmission Control Protocol (TCP) in the high-quality link regime and outperforms it as link quality deteriorates.

der ideal network conditions. As shown in the visualizations of network traces in Figure 4.3 and argued throughout this dissertation, conditions are rarely ideal in mobile robotic networks.

For each of the image sizes (200 KB, 500 KB, 1 MB, and 2 MB), we conducted 10 trials on both the indoor and outdoor traces for each transport protocol. Figure 4.4 shows the results of this experiment. Error bars represent one standard error of the mean across the 10 trials.

For all protocols, the percent of images delivered generally decreases as the image size increases. This is unsurprising since delivering larger images is more demanding than delivering smaller images. HBAEC outperformed the other protocols across the range of image sizes in both trials, with TCP only approaching its performance in the 200 KB image experiment of the indoor trace. HBAEC delivered approximately 75 percent of 200 KB images and 60-65 percent of the 2 MB images.

TCP exhibited the most variable performance across the trials, as indicated by its significant error bars. We hypothesize that this is due to the interaction of its exponential backoff mechanism and the stochastic network emulator. Depending on whether certain segments are successfully delivered or not, TCP may back off for significant time periods, the duration of which greatly affects performance on the continual image delivery task. This stalling behavior also resulted in TCP's performance approaching that of UDP on the intermediate-sized images during the indoor trace trials. The stateful TCP stalled for long enough time periods that the stateless, best-effort UDP delivered as many images successfully without any mechanism for recovering from loss of even a single packet.

Figure 4.5 breaks down the protocols' performance as a function of link quality for the 1 MB image trials. To produce this data, we associated each image produced with the signal strength measured at the time of the image's capture. We then calculated the average percent of images delivered for each level of signal strength. Note that an undelivered image is not necessarily due to a failed transmission; rather, an image may simply have been discarded if a previous image was still in the process of being transmitted.

The percentage of images delivered generally decreases as link quality worsens, reaching 0 percent for both protocols around -85 dBm. This suggests that an extremely low-quality link is incapable of supporting this type of bandwidth-intensive traffic, regardless of the transport protocol.

HBAEC generally outperforms TCP and UDT across the range of link qualities. However, TCP performs similarly to HBAEC in the high-quality link regime. This is unsur-

prising since TCP is designed to transport data across low-loss channels. As link quality worsens, HBAEC's relative performance advantage increases. HBAEC outperforms TCP in the range of approximately -40 to -75 dBm.

Together these results indicate the suitability of HBAEC for transmitting bandwidth-intensive traffic across the lossy, time-varying networks of mobile multi-robot systems.

4.6 Summary

In this chapter, we introduced HBAEC, an extension of AEC for high-bandwidth traffic. Encoding large quantities of data efficiently requires a particular type of erasure code with probabilistic decoding characteristics. We therefore generalized the methods of Chapter 3 to handle these types of erasure codes. We additionally improved upon the link quality estimation technique of Chapter 3, introducing a technique based on kernel density estimation that yields higher likelihood packet loss distribution estimates using the same historical measurements of packet loss. We incorporated these techniques into a simple transport protocol built on top of UDP. We described a hybrid evaluation technique, trace-based network emulation, that bridges the gap between real-world testing and simulation for robotic network testing. Finally, we demonstrated that HBAEC outperforms TCP at continually delivering images across a mobile robotic network.

CHAPTER 5

Evaluating Communication Decisions under Bandwidth Constraints

5.1 Introduction

Robots that collaborate on tasks can share information about their environment, helping their teammates make better decisions. When wireless bandwidth is limited, however, a robot may not be able to communicate all its observations. How should a robot decide what to communicate when faced with such a constraint?

Answering this question requires estimating the value of a proposed communication. A direct approach is to predict the effect of the message on team reward. However, such a direct approach is intractable for standard multi-agent models. For example, deciding a decentralized Markov decision process (Dec-MDP) or decentralized partially observable Markov decision process (Dec-POMDP) with communication is NEXP-complete [30, 65]. As a consequence, existing direct methods are limited to problems with only a few agents, states, and actions [14, 15, 76].

This complexity motivated indirect methods of estimating communication value. For example, some methods measure the information content contained in a message and assign high value to information-rich messages [94, 95]. Others track the coherence of the ego-agent's internal models of its teammates and communicate to keep the team's beliefs consistent [6, 76, 97].

In this chapter, we present a direct yet tractable method for reasoning about communication actions. To enable this tractability, we employ a specialized model of multi-agent decision-making. This modified Dec-MDP model (c.f. [3, 92]) makes assumptions that aid in tractability while maintaining fidelity to real-world problems. It features a deterministic transition function that is initially unknown to the agents, which corresponds to the

unknown map in exploration-style robotics tasks. Furthermore, the model is factored, that is, each agent makes decisions independently. Factored problems occur whenever agents collaborate while working on individual sub-tasks.

These features of the model make it tractable for an agent to evaluate a communication decision in terms of its expected effect on reward. Because the model is factored, an agent can compute its action policy independently of its teammates. Agents can use fast incremental algorithms (e.g. D*-Lite [48]) to plan through the state-action space since the transition function is deterministic. Fast, independent planning allows an agent to quickly forward simulate the behavior of its teammates. This leads to a key insight of this chapter: fast forward simulations make it tractable to estimate the effect of a message on team reward. Our first contribution then is a method that uses such forward simulations to evaluate messages directly in reward space.

Even once an agent has assigned a value to candidate messages, it still needs a mechanism for making communication decisions. Most existing methods (e.g. [4, 92, 97]) assume that the content of a message is static and consists of, for example, the agent's entire observation history. They estimate the value of this message, then weigh that value against a fixed communication cost that is part of the problem's reward structure. Such an approach answers the question of *when* the agent should communicate.

To respect practical bandwidth constraints, however, a robot must reason about *what* to communicate, a problem that has received little attention [77]. Consider for example a scenario in which a robot makes observations more quickly than the network would support sharing them. The robot then needs to decide which observations are worthwhile to send.

Our second contribution addresses this challenge. Namely, we apply a bandit-based combinatorial optimization algorithm [17] to select observations to communicate. This algorithm estimates the expected reward distribution associated with each observation by repeatedly forward simulating the outcome of messages containing that observation. It uses these estimates to build an approximately optimal message.

We call our complete method Optimizing Communication under Bandwidth Constraints (OCBC). We evaluate the performance of OCBC in a simulated multi-robot navigation problem. We compare it to a state-of-the-art approach [92] and show that OCBC achieves better task performance with less communication.

The key insight in this chapter is the path forward that we provide for making communication decisions in terms of their expected effect on reward. Exact, decision-theoretic communication with general multi-agent models (e.g. Dec-MDP or Dec-POMDP) is hope-

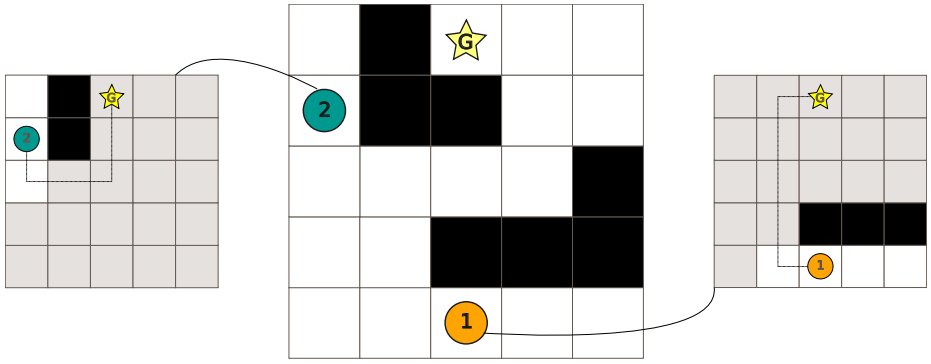

Figure 5.1: An example problem in which multiple robots independently navigate through an unknown environment. All robots try to reach the goal as quickly as possible; team reward depends on the total time taken by all robots. A robot observes the occupancy of adjacent cells as it moves through its environment. The robot can communicate these observations to its teammates to help them complete the task more quickly.

lessly intractable outside of toy problems. We argue that by deemphasizing uncertainty in the problem and using sampling to estimate potential communication values, we move the state-of-the-art forward in terms of the tractability of communication decision-making.

Our technical contributions in this chapter are:

- A mechanism for making communication decisions under bandwidth constraints in collaborative, multi-robot settings, using forward simulations to evaluate potential communication actions directly in terms of their expected effect on reward, and
- A bandit-based method that optimizes message content based on the results of these forward simulations.

5.2 Preliminaries

5.2.1 Model Formulation and Application

Consider the task of multiple robots navigating in an unknown environment as illustrated in Figure 5.1. The robots all move toward a goal destination, and they observe their environment as they travel through it. The robots can share these observations with their teammates to help them reach the destination more quickly.

We will use this example task to explain our model formulation.

Agent States

We denote the team of n agents as $I = \{1, \ldots, n\}$. At each time step $t \in \{1, \ldots, T\}$, agent i is in state $s_i(t) \in S$ and takes action $a_i(t) \in A$. In our example task, each robot's state is its current cell in the map, and its available actions are to move in any one of the cardinal directions (i.e. $A = \{\text{N}, \text{E}, \text{S}, \text{W}\}$).

Each agent has some goal state s_i^{GOAL} that it tries to reach, and each agent knows the desired joint goal state $\boldsymbol{s}^{\text{GOAL}} = \{s_1^{\text{GOAL}}, \ldots, s_n^{\text{GOAL}}\}$. Each agent i knows its own state s_i but cannot directly observe the joint state $\boldsymbol{s} = \{s_1, \ldots, s_n\}$. In our example task, this corresponds to the robot having a reliable localization system that provides its current location but not having a way to observe the locations of its teammates.

State Transition Function

The state transition function is deterministic (i.e. $P(s, a, s') \in \{0, 1\}$ for $s, s' \in S$ and $a \in A$), which corresponds to a robot having reliable closed-loop actuation. In this case, the success of an action depends only on whether the destination cell is occupied. Tasks like this have a spatial relationship between states and actions, giving the transition function two characteristics we exploit in this chapter.

First, a significant portion of the transition function is trivially zero-valued. Specifically, $P(s, a, s') = 0$ for any pair of states s, s' that do not correspond to adjacent cells or any actions a that would not move the robot between the two cells.

Second, many remaining elements of the transition function have a well-defined relationship. Specifically, all (non-trivial) elements transitioning to state s' are equal in value. We denote this value as $x_{s'} = P(s, a, s') \in \{0, 1\}$ for any state s and action a that would transition the robot to successor state s' if the cell at s' were unoccupied. This value is the same for any such state-action pair.

To better understand the spatial structure of the transition function and the significance of the value $x_{s'}$, consider the following 2×2 grid environment:

0	1
2	3

.

Transition function elements such as $P(0, \cdot, 3)$ are trivially zero-valued since no single action could move the robot from cell 0 to cell 3. Likewise, an element such as $P(0, \text{W}, 2)$ is trivially zero-valued since the action W cannot move the robot from cell 0 to cell 2. Of

the non-trivial elements of the transition function, all elements transitioning to a given state will have the same value. For example, we have $x_1 = P(0, \mathrm{E}, 1) = P(3, \mathrm{N}, 1) = 0$ and $x_2 = P(0, \mathrm{S}, 2) = P(3, \mathrm{W}, 2) = 1$. In other words, the elements of the transition function that would lead to cell 1 have value 0 since cell 1 is occupied, and those that would lead to cell 2 have value 1 since cell 2 is unoccupied. From this, observe that the value x_s corresponds to the occupancy of the cell at state s. For the remainder of the chapter, agents maintain the x_s values in place of the transition function. An agent can construct the full transition function from these values when needed.

Transition Beliefs

The agents do not know the true state transition function P. Equivalently, the agents do not know the true value of x_s for $s \in S$. Instead, each agent maintains a belief $b_i(x_s) \in \{0, 1\}$ for each x_s of the transition function. We denote the belief over all x_s as $b_i(x)$.

In our concrete example, the robots do not know the true map of the environment (i.e. P). The map is given by the occupancy value of its cells (i.e. x_s). The belief $b_i(x_s)$ corresponds to believed likelihood that a cell is occupied, which we assume is independent of any other $b_i(x_{s'})$. We illustrate such a belief in Figure 5.1.

Teammate Beliefs

Because an agent cannot directly observe the states or transition beliefs of other agents, it must represent these values with a belief distribution. Specifically, each agent i maintains a distribution $b_i(m_j)$ over possible models for each other agent j. An agent model $m_j = (s_j, s_j^{\text{GOAL}}, b_j(x))$ consists of a state, goal state, and transition belief. This type of modeling of other agents' mental states is an example of first-order Theory of Mind (ToM) [11].

In our example task, an agent model corresponds to a single estimate m_j of another robot's location and map. The belief $b_i(m_j)$ is a belief over all possible models, which we approximate with a particle filter (see Section 5.3.4). We denote the ego-agent's beliefs about all its teammates as $b_i(m)$.

Action Policies

All agents select actions using the same deterministic planning algorithm. This planner produces a policy $\pi_i : S \to A$ for agent i given its current state s_i, its goal state s_i^{GOAL}, and its transition belief $b_i(x)$.

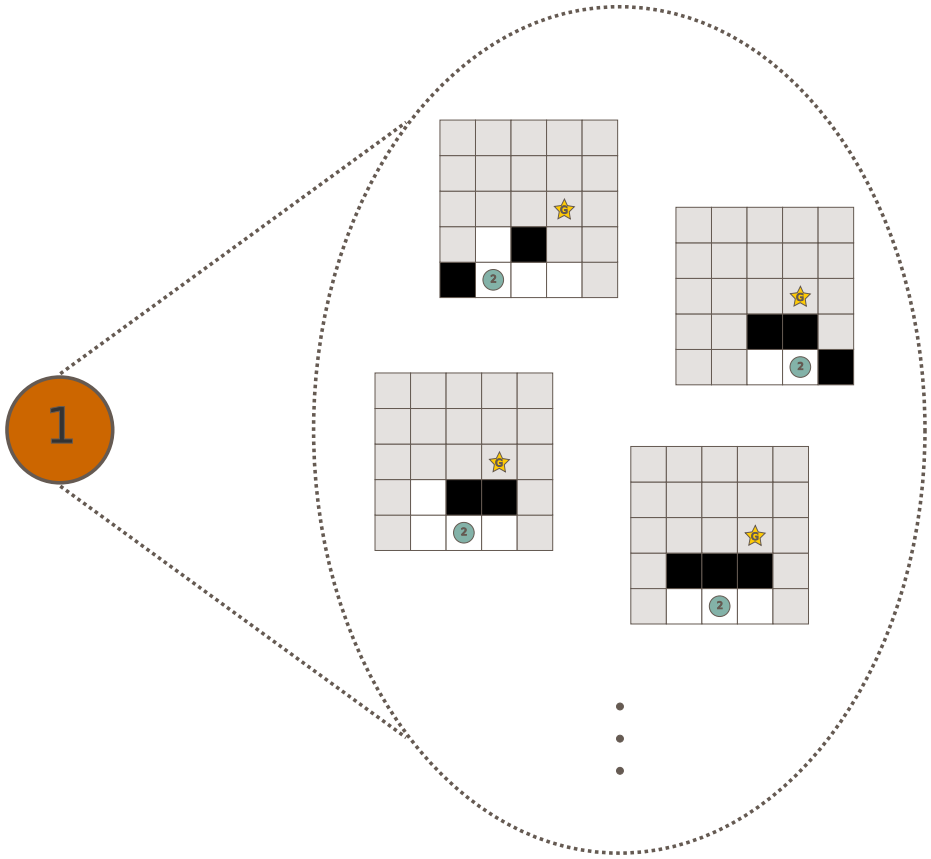

Figure 5.2: An illustration of Robot 1's belief over models of its teammate (Robot 2) in the multi-robot navigation task. These models represent possible locations and maps for Robot 2.

In our example task, robots compute the shortest path between their location and the goal location using their current map, assuming that unobserved cells are unoccupied [48].

Observations

An observation $\omega^{(s)} \in \{0, 1\}$ is sampled from the observation function $O(\omega^{(s)}|x_s)$. In our example task, this corresponds to an observation of the occupancy of the cell at state s. Though our formulation admits a stochastic observation function, we assume in our evaluation that O is deterministic, in which case $\omega^{(s)}$ reveals the true value of x_s.

At each time step t, agent i receives a set of observations $\boldsymbol{\omega}_i(t)$. This corresponds in our example task to a robot observing all its adjacent cells.

Communication

Agent i carries out a communication action $c_i(t) \in C$ at each time step t. This broadcasts observations $\boldsymbol{\omega}_{c_i}$ to all other agents along with the ego-agent's state s_i. This results in the other agents incorporating the associated observations into their beliefs.

Reward

We compute each agent's reward independently and then combine the individual rewards to yield the team reward (i.e. the reward function is factored). An agent's reward comes from two components. The action reward function $R_A : S \times A \to \mathbb{R}$ gives the reward an agent receives after taking an action from a particular state. The communication reward function $R_C : C \to \mathbb{R}$ assigns reward based on a communication action. The total reward for the team is given by

$$\begin{aligned} \gamma &= \gamma_A + \gamma_C \\ &= \sum_{i \in I} \gamma_{Ai} + \gamma_{Ci} \\ &= \sum_{i \in I} \sum_{t \in T} R_A\big(s_i(t), a_i(t)\big) + R_C\big(c_i(t)\big). \end{aligned} \tag{5.1}$$

Finding an appropriate weighting of multiple reward functions into a single overall reward function is a non-trivial exercise. However, this is not the focus of the material in this chapter. Rather, we assume the reward structure is defined for us as a part of the task. In

our evaluation, we will consider the weighting of action reward and communication reward as a free parameter and then sweep across it.

In our example task, we assign action reward -1 for each step the robots take until they reach the goal. We assign communication reward differently depending on the communication paradigm, which we discuss in Section 5.2.2.

Model Discussion

Overall, the previously described multi-agent model most closely resembles a Dec-MDP with a few key differences. First, the transition function P is considered to be deterministic, whereas in a Dec-MDP the transition function is stochastic. Second, the model considers a sub-set of multi-agent tasks in which the agents collaborate while carrying out individual sub-tasks, allowing the planning process to be factored. This differs from a Dec-MDP, which plans in the joint action space. Finally, our model assumes that the transition function is partially unknown to the agents, whereas in a Dec-MDP all agents know the transition function.

The first two differences previously outlined are simplifications of the more general Dec-MDP model. As with any such simplifications, this does limit the general applicability of the model. Furthermore, we are not suggesting that this is the best set of simplifications one could make or that they apply to the largest subset of multi-agent tasks. Rather, we argue that making these types of specializations is necessary for tractability in a domain as challenging as decision-theoretic communication planning. Future researchers might apply a similar approach with an entirely different set of specializations to adapt this research to other tasks. The ideas presented in the following sections are more generally applicable than the particular model we employ.

5.2.2 Communication Paradigms

Roth et al. [77] introduce the concept of communication paradigms, which are sets of rules governing how much communication is allowed or how much communication costs. They propose three such paradigms, which we summarize here. These paradigms give context to OCBC and related methods. In Section 5.4, we use these paradigms to structure our evaluation and facilitate comparison with existing work.

Fixed-Cost Communication

Most decision-theoretic methods (e.g. [15, 92, 97]) only focus on the question of *when* to communicate. This limits the set of possible communication actions C to only two members: share all information possible (e.g. the ego-agent's entire observation history) or do not communicate at all. The former action is assigned a fixed cost (i.e. $R_C(c) = \epsilon < 0$) while the latter has zero cost. We call this paradigm *Fixed-Cost Communication*.

Proportional-Cost Communication

To penalize excessive bandwidth consumption, communication cost should depend on message size. This is the *Proportional-Cost Communication* paradigm, in which the cost of a communication action is proportional to the length of the associated message (i.e. $R_C(c) \propto |\boldsymbol{\omega}_c|$). Roth et al. [77] use this paradigm to motivate the question of *what* to communicate.

Fixed-Bandwidth Communication

Multi-robot teams using a wireless network must share some fixed amount of bandwidth [7, 40, 81]. We denote the amount of bandwidth available to the team at each time step as β. The team allocates a portion of this bandwidth, $\beta_i(t)$, to each agent i at time step t through an offline round robin schedule or other similar mechanism. Each agent must then decide how to use this bandwidth allocation. We call such a paradigm *Fixed-Bandwidth Communication*.

In the *Fixed-Bandwidth* paradigm, there is no cost associated with a communication action (i.e. $R_C(\cdot) = 0$). The set of communication actions available to an agent, $C_i(t)$, contains all messages that fit within its bandwidth allocation at time step t. That is, $C_i(t) = \{c : |\boldsymbol{\omega}_c| \leq \beta_i(t)\}$.

As we present it in the following section, OCBC fits within the *Fixed-Bandwidth* paradigm. In Section 5.4, we will discuss how to modify OCBC to compare its performance to methods from the other two paradigms.

5.3 Approach

We now introduce OCBC and its supporting algorithms. Section 5.3.1 explains OCBC, which is a bandit-based approach to optimizing multi-robot communication under band-

width constraints. Section 5.3.2 shows how OCBC evaluates communication actions using forward simulation. These two sections correspond to our two primary contributions.

The remaining two sections discuss how we implement OCBC in the context of a sequential decision-making agent. Section 5.3.3 gives the main sequential decision-making method, and Section 5.3.4 deals with maintaining and updating agent beliefs within that method.

5.3.1 Optimizing Communication under Bandwidth Constraints

An agent typically has multiple observations it could communicate with its teammates. When the agent's bandwidth allocation is too small to share all of those observations, it must decide which observations to include. This requires estimating the reward associated with each observation, which we do by forward simulation (see Section 5.3.2). Because of the uncertainty in the ego-agent's beliefs, though, any one forward simulation only samples from the distribution of rewards for a given communication. In this way, the observations are like levers a multi-armed bandit could pull, and our goal is to efficiently identify the observations with the highest expected reward.

To this end, we apply the combinatorial successive accept-reject (CSAR) algorithm [17], which is a bandit-based method that performs combinatorial optimization under uncertain rewards. CSAR tries to find an optimal member of a specified decision class (e.g. all arm combinations of a certain size). CSAR stands for "Combinatorial Successive Accept Reject", a title that explains the algorithm's basic function. Given a set of K arms, CSAR makes K successive decisions to either accept or reject an arm. In between decisions, CSAR samples from the remaining arms to better estimate their associated reward distributions. CSAR makes the accept-reject decision on the arm for which it has the highest-confidence reward distribution estimate. By the end of this process, the set of accepted arms is an approximately optimal member of the decision class. See Chen et al. [17] for details on suboptimality bounds, etc.

Algorithm 5.1 shows how we apply the CSAR algorithm to the problem of optimizing communication under bandwidth constraints. The function OPTIMIZECOMMUNICATION takes in the ego-agent's current beliefs about the transition function $b_i(x)$ and its teammates $b_i(m)$ and decides which observations from the set $\boldsymbol{\omega}$ to communicate. The resulting message must satisfy the ego-agent's bandwidth constraint β_i.

The function has a fixed computational budget B (in terms of iterations of the main

Algorithm 5.1 Optimizing Communication under Bandwidth Constraints (OCBC)

For convenience, let β, $b_i(x)$, $b_i(m)$, $\hat{\Delta}_k$, and μ_k be global variables shared among all functions of Algorithms 5.1 to 5.5

1: **function** OPTIMIZECOMMUNICATION($b_i(x)$, $b_i(m)$, $\boldsymbol{\omega}$)
2: $K \leftarrow |\boldsymbol{\omega}|$ ▷ $\boldsymbol{\omega}$: the set of all observations
3: $\boldsymbol{\omega}^{\text{ACC}} \leftarrow \emptyset$ ▷ $\boldsymbol{\omega}^{\text{ACC}}$: Observations accepted into message
4: $\boldsymbol{\omega}^{\text{CAND}} \leftarrow \boldsymbol{\omega}$ ▷ $\boldsymbol{\omega}^{\text{CAND}}$: Remaining candidate observations
5: $\hat{\Delta}_k \leftarrow \emptyset,\ k \in \{1, \ldots, K\}$ ▷ $\hat{\Delta}_k$: Set of rewards observed empirically for observation k
6: $\mu_k \leftarrow 0,\ k \in \{1, \ldots, K\}$ ▷ μ_k: Mean of $\hat{\Delta}_k$
7: **for** $k \leftarrow 1, \ldots, K$ **do**
8: $B_k \leftarrow$ ALLOCATESAMPLEBUDGET(B, K, k) ▷ (Algorithm 5.2)
9: ESTIMATEREWARDDISTRIBUTIONS(B_k, $\boldsymbol{\omega}^{\text{CAND}}$) ▷ (Algorithm 5.3)
10: $\hat{\boldsymbol{\omega}}^* \leftarrow$ ASSEMBLE($\boldsymbol{\mu}$, $\boldsymbol{\omega}^{\text{ACC}}$, $\boldsymbol{\omega}^{\text{CAND}}$) ▷ (Algorithm 5.4) ▷ $\hat{\boldsymbol{\omega}}^*$: Optimal message estimate
11: $l \leftarrow$ SELECT($\boldsymbol{\omega}^{\text{ACC}}$, $\boldsymbol{\omega}^{\text{CAND}}$, $\hat{\boldsymbol{\omega}}^*$) ▷ (Algorithm 5.5) ▷ ω_l: observation to accept/reject
12: **if** $\omega_l \in \hat{\boldsymbol{\omega}}^*$ **then** ▷ Is selected observation part of optimal message estimate?
13: $\boldsymbol{\omega}^{\text{ACC}} \leftarrow \boldsymbol{\omega}^{\text{ACC}} \cup \{\omega_l\}$ ▷ If so, accept it into actual message
14: $\boldsymbol{\omega}^{\text{CAND}} \leftarrow \boldsymbol{\omega}^{\text{CAND}} \setminus \{\omega_l\}$ ▷ Remove observation from remaining candidates
15: **return** $\boldsymbol{\omega}^{\text{ACC}}$

loop) that it uses to evaluate observations and decide what to communicate. The CSAR algorithm dictates how much of this computational budget should be allocated to each of the K accept-reject decisions (Algorithm 5.2, see [17] for derivation). The algorithm uses this budget allocation B_k to refine its reward distribution estimates of the candidate arms.

ESTIMATEREWARDDISTRIBUTIONS (Algorithm 5.3) is responsible for this task. The function carries out B_k forward simulations on each of the candidate observations. We begin each of these forward simulations by sampling a set of agent models (Line 3) and a transition function from the ego-agent's beliefs (Lines 4 to 5). The function EVALUATECOMMUNICATION returns the reward obtained from communicating a candidate observation given that set of agent models and transition function (see Section 5.3.2). This reward is used to update the statistics associated with the observation (Lines 8 to 9).

Once the reward distributions have been updated, we assemble the estimated optimal message $\hat{\boldsymbol{\omega}}^*$ according to the current rewards and the bandwidth constraint (Line 10 of Algorithm 5.1). The function ASSEMBLE (Algorithm 5.4) is responsible for this task. It starts by adding all of the previously accepted observations into the message. It then re-

Algorithm 5.2 Allocating OCBC Sample Budget

See Algorithm 5.1 for note on shared global variables

function ALLOCATESAMPLEBUDGET(B, K, k) ▷ See [17] for derivation
  $\kappa(y) \triangleq \sum_{y=1}^{K} \frac{1}{y}$ ▷ K: Number of observations
  ▷ (i.e. number of decisions to make)
  $f(y) \triangleq \lceil \frac{B-K}{\kappa(y)(K-y+1)} \rceil, \;\; y > 0$ ▷ B: Total sample budget
  $f(0) \triangleq 0$
  return $f(k) - f(k-1)$ ▷ k: Number of accept-reject decisions already made

Algorithm 5.3 Estimating OCBC Reward Distributions

See Algorithm 5.1 for note on shared global variables

function ESTIMATEREWARDDISTRIBUTIONS(B_k, $\boldsymbol{\omega}^{\text{CAND}}$)
  for $1, \ldots, B_k$ **do** ▷ While within the sample budget B_k
    $\tilde{\boldsymbol{m}} \leftarrow \bigcup_{j \in I \setminus i}$ Sample $m_j \sim b_i(m_j)$ ▷ Sample model m_j for each agent j
    Sample $\tilde{x} \sim b_i(x)$ ▷ Sample map instance $\tilde{x}$
    $\tilde{P} \leftarrow$ CONSTRUCTTRANSITIONFUNCTION($\tilde{x}$)
    for $\omega_l \in \boldsymbol{\omega}^{\text{CAND}}$ **do** ▷ For each candidate observation ω_l
      $\hat{\delta} \leftarrow$ EVALUATECOMMUNICATION($\{\omega_l\}$, $\tilde{\boldsymbol{m}}$, $\tilde{P}$) ▷ (Algorithm 5.6)
      ▷ $\hat{\delta}$: Reward for ω_l given world $\tilde{P}$ and agents $\tilde{\boldsymbol{m}}$
      $\hat{\Delta}_l \leftarrow \hat{\Delta}_l \cup \{\hat{\delta}\}$
      $\mu_l \leftarrow \frac{1}{|\hat{\Delta}_l|} \sum_{\hat{\delta} \in \hat{\Delta}_l} \hat{\delta}$ ▷ μ_l: Mean of sampled rewards for ω_l

peatedly adds the best remaining observation to the message until the bandwidth constraint is reached or the candidate pool is exhausted.

We use the optimal message estimate $\hat{\boldsymbol{\omega}}^*$ to help us select the observation we will decide on at this iteration (Line 11 of Algorithm 5.1) using the function SELECT (Algorithm 5.5). This message estimate corresponds to a particular accept-reject decision for each of the remaining candidate elements. We then assemble optimal message estimates $\hat{\boldsymbol{\omega}}_l^*$ in which we force the alternative decision to be made for each element, ω_l (Lines 4 to 7). The difference in the reward expected to result from $\hat{\boldsymbol{\omega}}^*$ (the optimal message estimate) and $\hat{\boldsymbol{\omega}}_l^*$ (the optimal message estimate with the alternative decision about ω_l) is the suboptimality gap associated with ω_l (Line 9). The size of this suboptimality gap is a measure of the confidence of the associated reward distribution. We therefore select the observation with the largest suboptimality gap and decide whether to accept or reject it (Lines 12 to 14 of Algorithm 5.1).

The running time of Algorithm 5.1 depends on the overall sampling budget B (which is a free parameter) as well as parameters given by the problem instance. The function

Algorithm 5.4 Assembling OCBC Optimal Message Estimate

See Algorithm 5.1 for note on shared global variables

function ASSEMBLE($\boldsymbol{\mu}$, $\boldsymbol{\omega}^{\text{ACC}}$, $\boldsymbol{\omega}^{\text{CAND}}$)
  $\hat{\boldsymbol{\omega}}^* \leftarrow \boldsymbol{\omega}^{\text{ACC}}$ ▷ Add accepted observations to optimal message estimate $\hat{\boldsymbol{\omega}}^*$
  while $|\hat{\boldsymbol{\omega}}^*| < \beta_i$ **and** $|\boldsymbol{\omega}^{\text{CAND}}| > 0$ **do** ▷ While message fits bandwidth allocation β_i
    $l \leftarrow \arg\max_{l:\omega_l \in \boldsymbol{\omega}^{\text{CAND}}} \mu_l$ ▷ Select candidate ω_l with highest mean reward
    if $\mu_l \leq 0$ **then return** $\hat{\boldsymbol{\omega}}^*$
    $\hat{\boldsymbol{\omega}}^* \leftarrow \hat{\boldsymbol{\omega}}^* \cup \{\omega_l\}$ ▷ Add to optimal message estimate
    $\boldsymbol{\omega}^{\text{CAND}} \leftarrow \boldsymbol{\omega}^{\text{CAND}} \setminus \{\omega_l\}$ ▷ Remove from candidate observations
  return $\hat{\boldsymbol{\omega}}^*$

Algorithm 5.5 Selecting OCBC Observation for Accept-Reject Decision

See Algorithm 5.1 for note on shared global variables

function SELECT($\boldsymbol{\omega}^{\text{ACC}}$, $\boldsymbol{\omega}^{\text{CAND}}$, $\hat{\boldsymbol{\omega}}^*$)
  $\hat{\gamma}^* \leftarrow \sum_{l:\omega_l \in \hat{\boldsymbol{\omega}}^*} \mu_l$ ▷ $\hat{\gamma}^*$: Total reward for optimal message estimate
  for $\omega_l \in \boldsymbol{\omega}^{\text{CAND}}$ **do**
    if $\omega_l \in \hat{\boldsymbol{\omega}}^*$ **then** ▷ If ω_l is part of optimal message estimate
      $\hat{\boldsymbol{\omega}}_l^* \leftarrow$ ASSEMBLE($\boldsymbol{\mu}$, $\boldsymbol{\omega}^{\text{ACC}}$, $\boldsymbol{\omega}^{\text{CAND}} \setminus \{\omega_l\}$) ▷ Try excluding ω_l
    else
      $\hat{\boldsymbol{\omega}}_l^* \leftarrow$ ASSEMBLE($\boldsymbol{\mu}$, $\boldsymbol{\omega}^{\text{ACC}} \cup \omega_l$, $\boldsymbol{\omega}^{\text{CAND}} \setminus \{\omega_l\}$) ▷ Try including ω_l
    $\hat{\gamma}_l^* \leftarrow \sum_{l:\omega_l \in \hat{\boldsymbol{\omega}}_l^*} \mu_l$ ▷ $\hat{\gamma}_l^*$: Reward for message with alternate decision about ω_l
    $\psi_l \leftarrow \hat{\gamma}^* - \hat{\gamma}_l^*$ ▷ ψ_l: Reward lost for alternate decision (i.e. suboptimality gap)
  return $\arg\max_l \psi_l$ ▷ Select observation with largest suboptimality gap

OPTIMIZECOMMUNICATION carries out B evaluations to refine its estimates of the reward distributions associated with each candidate observation. In each of these evaluations, the reward distribution for each candidate observation is updated. At most K such updates occur, since K is the number of initial candidate observations, and the pool of candidates shrinks as each accept-reject decision is made. Updating the reward distribution for a candidate observation requires a call to the function EVALUATECOMMUNICATION, which carries out forward simulations for all $n-1$ teammate agents (see Section 5.3.2). Forward simulating to time horizon T requires a policy computation at each time step, the cost of which depends on the particular planning algorithm used. An implementation with a naive shortest path planner would have complexity $O(|S| + |S||A|)$ (i.e. the sum of the number of vertices and edges in the state-action graph), but an incremental replanning algorithm requires this computation only at the first time step and then can quickly update the policy thereafter [48]. Altogether, the worst-case complexity of OPTIMIZECOMMUNICATION is

Algorithm 5.6 Evaluating Communication Actions

function EVALUATECOMMUNICATION($\boldsymbol{\omega}_c$, $\tilde{\boldsymbol{m}}$, $\tilde{P}$)
  for $\tilde{m}_j \in \tilde{\boldsymbol{m}}$ **do** ▷ For all teammates
    $(s_j, s_j^{\text{GOAL}}, b_j(x)) \leftarrow \tilde{m}_j$ ▷ Teammate's state (s_j)
    ▷ Teammate's goal (s_j^{GOAL})
    ▷ Teammate's belief ($b_j(x)$)
    $\gamma_{A_j}^{\text{NoCom}} \leftarrow$ FORWARDSIMULATE($\tilde{m}_j$, $\tilde{P}$)
    ▷ $\gamma_{A_j}^{\text{NoCom}}$: Task reward without communication
    $b'_j(x) \leftarrow$ INCORPORATEOBSERVATIONS($b_j(x)$, $\boldsymbol{\omega}_c$)
    ▷ $b'_j(x)$: Belief after communication
    $\tilde{m}'_j \leftarrow (s_j, s_j^{\text{GOAL}}, b'(x))$ ▷ $\tilde{m}'_j$: Teammate model after communication
    $\gamma_{A_j}^{\text{Com}} \leftarrow$ FORWARDSIMULATE($\tilde{m}'_j$, $\tilde{P}$)
    ▷ $\gamma_{A_j}^{\text{Com}}$: Task reward with communication
  $\delta \leftarrow R_C(c) + \sum_j (\gamma_{A_j}^{\text{Com}} - \gamma_{A_j}^{\text{NoCom}})$
  ▷ δ: Total reward difference from communication
  return δ

$O(BKnT|S||A|)$, but careful implementation can reduce this.

One drawback of the CSAR algorithm is that it assumes independence between the reward distributions of arms. Such an assumption is necessary for CSAR's tractability, but it may harm performance in tasks where the reward distributions are not independent. For example, there may be scenarios in which communicating the occupancy of one cell is only useful when combined with communicating the occupancy of a second cell. We leave study of this drawback as future work.

5.3.2 Evaluating Communication Actions

Evaluating candidate communications in terms of their expected effect on reward is typically expensive. In this section, we provide our method for performing this evaluation efficiently and review the key aspects of our formulation that enable this approach.

General multi-agent models are unfactored, assuming that team reward is a function of joint actions. Such models require planning in the joint action space, which scales exponentially with the number of agents. In contrast, we consider only factored multi-agent problems, where agents plan independently of one another. The computational cost of forward simulating such problems then scales linearly with the number of agents.

Furthermore, we assume that the transition function is deterministic. We can transform such a deterministic transition function into a directed graph where nodes are states and

Algorithm 5.7 Forward Simulation

```
1:  function FORWARDSIMULATE(m̃_j, P̃)
2:      (s_j, s_j^GOAL, b_j(x)) ← m̃_j      ▷ Teammate's state (s_j), goal (s_j^GOAL), and belief (b_j(x))
3:      γ_A ← 0                              ▷ γ_A: Teammate's accumulated action reward
4:      for t, ..., T do
5:          ω ← OBSERVE(s_j, P̃)              ▷ Generate observation ω of sampled world
6:          b_j(x) ← INCORPORATEOBSERVATIONS(b_j(x), ω)
                                             ▷ Update teammate's belief b_j(x)
7:          π ← COMPUTEPOLICY(s_j, s_j^GOAL, b_j(x))        ▷ Plan for teammate
8:          a ← π(s_j)                       ▷ Select teammate action a
9:          s_j ← arg max_{s'} P̃(·|s_j, a)   ▷ Update teammate state s_j
10:         γ_A ← γ_A + R_A(s_j, a)          ▷ Augment action reward γ_A earned by teammate
11:     return γ_A
```

edges are actions. An agent can compute a policy by finding a shortest path through the state-action graph. We can further leverage incremental shortest-path planners to recompute the policy quickly at each step of the forward simulation [48].

Algorithm 5.6 specifies our method for evaluating candidate communications. The goal of this algorithm is to estimate the effect a proposed communication will have on team reward given sampled agent models and a sampled transition function (see Algorithm 5.1). To measure this effect, we first forward simulate the agent models to get the baseline reward that occurs without any communication (Line 4). Next, we incorporate the communicated observation into the agent models' beliefs and repeat the forward simulation (Lines 5 to 7). The difference in reward is an estimate of the value of the communication (Line 8).

The function FORWARDSIMULATE (Algorithm 5.7) estimates the reward earned by an agent model $\tilde{m}_j$ in a world given by transition function $\tilde{P}$. At each time step of the forward simulation, the modeled agent first observes its environment and incorporates that observation into its belief (Lines 5 to 6). The modeled agent then plans and executes an action and arrives in an updated state (Lines 7 to 9). The function returns the cumulative action reward earned by the modeled agent (Line 11).

5.3.3 Making Sequential Communication Decisions with OCBC

In Algorithm 5.8, we show how we use OCBC within a sequential decision-making agent. This algorithm provides context for using OCBC as well as a means for evaluating its performance (see Section 5.4).

Algorithm 5.8 Sequential Decision Algorithm

For convenience, let $s_i(\cdot)$, s_i^{GOAL}, $a_i(\cdot)$, $\boldsymbol{\omega}_i(\cdot)$, $c_i(\cdot)$ be global variables shared among all functions

function MAKESEQUENTIALDECISIONS( )
  $b_i(x), b_i(m) \leftarrow$ INITIALIZEBELIEFS( ) ▷ $b_i(x)$: Map belief ▷ $b_i(m)$: Teammate beliefs
  for $t \in \{1, \ldots, T\}$ **do**
    $\boldsymbol{\omega}_i(t) \leftarrow$ OBSERVE($s_i(t)$, P) ▷ $\boldsymbol{\omega}_i(t)$: Current observations
    $\boldsymbol{c}(t-1) \leftarrow$RECEIVECOMMUNICATIONS( ) ▷ $\boldsymbol{c}(t-1)$: Previous communications
    $b_i(x), b_i(m) \leftarrow$ UPDATEBELIEFS($b_i(x)$, $b_i(m)$)
    $\pi, c_i(t) \leftarrow$ PLAN($b_i(x)$, $b_i(m)$) ▷ π: Planned policy ▷ $c_i(t)$: Planned communication
    $s_i(t+1) \leftarrow$ ACT(π)

function INITIALIZEBELIEFS( )
  for $s \in S$ **do**
    $b_i(x_s) \leftarrow p_s$ ▷ Assign prior to each map cell
  for $j \in I \setminus \{i\}$ **do**
    $m_j \leftarrow (s_j, s_j^{\text{GOAL}}, b_j(x))$
    $b_i(m_j) \leftarrow \bigcup_{1,\ldots,M}\{m_j\}$ ▷ Add M copies of teammate model to particle filter
  $b_i(m) \leftarrow \bigcup_{j \in I\setminus\{i\}} b_i(m_j)$
  return $b_i(x), b_i(m)$

function UPDATEBELIEFS($b_i(x)$, $b_i(m)$)
  $\boldsymbol{\omega}_{\text{UPDATE}} \leftarrow \boldsymbol{\omega}_i(t) \cup \boldsymbol{c}(t-1)$ ▷ Aggregate direct and communicated observations
  $b_i'(x) \leftarrow$ INCORPORATEOBSERVATIONS($b_i(x)$, $\boldsymbol{\omega}_{\text{UPDATE}}$) ▷ (Algorithm 5.9)
  $b_i'(m) \leftarrow$ UPDATETEAMBELIEFS($b_i(m)$, $\boldsymbol{\omega}_{\text{UPDATE}}$) ▷ (Algorithm 5.10)
  return $b_i'(x), b_i'(m)$

function PLAN($b_i(x)$, $b_i(m)$)
  $\pi \leftarrow$ COMPUTEPOLICY($s_i(t)$, s_i^{GOAL}, $b_i(x)$) ▷ Plan e.g. with [48]
  $\boldsymbol{\omega}_i^{\text{ALL}} \leftarrow \bigcup_{\tau\in\{1,\ldots,t\}} \boldsymbol{\omega}_i(\tau)$ ▷ Aggregate all previous observations
  $c_i(t) \leftarrow$ **OPTIMIZECOMMUNICATION**($b_i(x)$, $b_i(m)$, $\boldsymbol{\omega}_i^{\text{ALL}}$)
  return $\pi, c_i(t)$

function ACT(π)
  $a_i(t) \leftarrow \pi(s_i(t))$ ▷ Select action from computed policy
  $s_i(t+1) \leftarrow \arg\max_{s'} P(\cdot|s_i(t), a_i(t))$ ▷ Transition to new state
  COMMUNICATE($c_i(t)$)
  return $s_i(t+1)$

Initialization

The algorithm begins by initializing the ego-agent's beliefs via the function INITIALIZEBELIEFS (Lines 9 to 16). The agent starts with some prior p_s for each transition function value $b_i(x_s)$ (Line 11). The ego-agent also initializes its beliefs about each of its teammates (Lines 12 to 15). In our evaluation, we assume a strong prior on this belief, that is, we assume that the ego-agent knows the initial state and transition belief of each of its teammates. Such a prior could come from an offline sharing step before the task begins. We implement the belief over agent models as a set of particles (see Section 5.3.4), so the initial particle set contains M copies of the agent model m_j, where M is a user-specified parameter.

Sensing

At each time step t, the ego-agent first senses its environment, making observations $\boldsymbol{\omega}_i(t)$ drawn from the observation function $O(\omega^{(s)}|x_s)$ (Line 4). Additionally, the ego agent receives the communications $\boldsymbol{c}(t-1)$ transmitted by its teammates at the previous time step (Line 5).

Updating Beliefs

Next, the agent updates its beliefs based on the result of the sensing step (Line 6). The ego-agent aggregates all the observations it made directly or received via communication (Line 18). It incorporates these observations into its transition belief via the function INCORPORATEOBSERVATIONS (see Algorithm 5.9). The ego-agent then updates its beliefs about its teammates via the function UPDATETEAMBELIEFS (see Section 5.3.4 and Algorithm 5.10).

Planning

The ego-agent computes a policy π based on its current state $s_i(t)$ and its transition belief $b_i(x)$ (Line 23). Then, it selects a (possibly empty) set of observations to communicate using the function OPTIMIZECOMMUNICATION, the implementation of which is our primary contribution (see Algorithm 5.1).

Algorithm 5.9 Incorporating Observations into a Transition Belief

```
1: function IncorporateObservations(b(x), ω)
2:     for ω^(s) ∈ ω do
3:         b'(x_s) ← ηO(ω^(s)|x_s)b(x_s)        ▷ Bayesian update to belief element
4:     return b'(x)
```

Acting

Finally, the agent executes action $a_i(t)$ based on the policy π and arrives in state $s_i(t+1)$ (Lines 28 to 29). The agent then broadcasts its selected set of observations to all its teammates (Line 30).

5.3.4 Updating Agent Beliefs

Transition Beliefs

Algorithm 5.9 shows how we update an agent's transition belief based on a set of observations. Recall that we assume the beliefs about transition elements are independent of one another. Therefore, observation $\omega^{(s)}$ only affects $b(x_s)$. We use Bayes rule to compute the new value of the belief for each affected element (Line 3).

Teammate Beliefs

Algorithm 5.10 details our method of updating the ego-agent's beliefs about its teammate models. The algorithm takes in the existing beliefs $b_i(m)$ as well as a set of new observations $\boldsymbol{\omega}_{\text{UPDATE}}$. It performs a particle filter update on the belief $b_i(m_j)$ about each teammate j (Lines 3 to 14).

First, each particle $m^{(k)}$ is evolved forward one step (Lines 4 to 12). Recall that each particle $m^{(k)}$ is an agent model that contains a state, goal state, and transition belief. We then plan and execute an action for the agent model (Lines 5 to 6). To simulate the next time step for the agent model, we sample from the model's transition element beliefs and construct a transition function $\tilde{P}$ (Lines 7 to 8). We use $\tilde{P}$ to find the new state s' and generate an observation $\boldsymbol{\omega}$ (Lines 9 to 10). We then incorporate this observation into the particle's transition belief to get $b'(x)$ (Line 11). At this point, we assemble s', $s_{m_k}^{\text{GOAL}}$, and $b'(x)$ into an evolved particle m'_k. We assign the particle weight w_k based on the likelihood of the actual observations $\boldsymbol{\omega}_{\text{UPDATE}}$ given the updated model (Line 13). We then re-sample

Algorithm 5.10 Updating Team Beliefs

1: **function** UPDATETEAMBELIEFS($b_i(m)$, $\boldsymbol{\omega}_{\text{UPDATE}}$)
2: **for** $b_i(m_j) \in b_i(m)$ **do** ▷ For each teammate j
3: **for** $m^{(k)} \in b_i(m_j)$ **do** ▷ For all particles (models) of that teammate
4: $\big(s, s^{\text{GOAL}}, b(x)\big) \leftarrow m^{(k)}$
5: $\pi \leftarrow$ COMPUTEPOLICY(s, s^{GOAL}, $b(x)$)
6: $a \leftarrow \pi(s)$
7: Sample $\tilde{x} \sim b(x)$ ▷ Sample map from ego-agent's belief
8: $\tilde{P} \leftarrow$ CONSTRUCTTRANSITIONFUNCTION($\tilde{x}$)
9: $s' \leftarrow \arg\max_{s'} \tilde{P}(\cdot|s, a)$ ▷ Update state based on sampled map
10: $\boldsymbol{\omega} \leftarrow$ OBSERVE(s', $\tilde{P}$) ▷ Simulate observations based on sampled map
11: $b'(x) \leftarrow$ INCORPORATEOBSERVATIONS($b(x)$, $\boldsymbol{\omega}$)
12: $m'_k \leftarrow (s', s^{\text{GOAL}}, b'(x))$
13: $w_k \leftarrow \prod_{\omega^{(s)} \in \boldsymbol{\omega}_{\text{UPDATE}}} \mathbb{P}\big(\omega^{(s)} | b'(x_{s'})\big)$ ▷ Assign particle weight
14: $b'_i(m_j) \leftarrow \bigcup_{1,\dots,M}$ Sample m'_k with probability $\propto w_k$ ▷ Re-sample particles
15: **return** $b'_i(m)$

the particles according to these weights to get the updated set $b'_i(m_j)$ (Line 14).

5.4 Evaluation

In this section, we evaluate our method on the multi-robot navigation task introduced in Section 5.2.1. We structure our evaluation around the three communication paradigms introduced in Section 5.2.2. As we have presented it so far, OCBC is a part of the *Fixed-Bandwidth* paradigm. The closest existing work [92] is in the *Fixed-Cost* paradigm. We therefore evaluate variants of OCBC for the *Fixed-Cost* and *Proportional-Cost* paradigms that allow us to compare it to this existing method.

We first detail the experimental setup (Section 5.4.1). Then, we compare OCBC to ConTaCT [92] in the *Fixed-Cost* and *Proportional-Cost* paradigms (Section 5.4.2). Finally, we compare OCBC to a randomized baseline method in the *Fixed-Bandwidth* paradigm (Section 5.4.3).

5.4.1 Experimental Setup

In the experiments of Section 5.4.2, we use five agents in each simulation. In Section 5.4.3, we vary the number of agents between two and ten.

We begin each simulated trial by generating a 10×10 gridmap. The occupancy of each grid cell is determined by a Bernoulli trial with probability 0.3. Such a randomized method can yield maps without feasible paths between certain cells. We therefore test that the resulting gridmap is fully connected; that is, any given cell in the map must be reachable from every other cell. If the gridmap is not fully connected, we replace it with a new randomly generated map until the condition is met.

Once the map is generated, we select a shared goal cell and agent start cells uniformly at random from among the unoccupied cells. Because of the random nature of this process, some generated problem instances are not interesting tests of communication. For example, agents that begin on opposite sides of a map may not benefit at all from sharing observations with one another. We can detect such cases by conducting two initial simulations: in the first, agents automatically share all observations, and in the second, agents share nothing. If both simulations yield the same result, we reject the trial as uninteresting and generate a new gridmap, a new goal location, and new agent locations.

Each simulated trial returns the reward, $\gamma(\chi) = \gamma_A(\chi) + \gamma_C(\chi)$, earned by the team using method χ. We are interested in measuring the effect of communication on task performance. In this case, we measure task performance by action reward, which is given by $\gamma_A(\chi)$. To measure the effect of communication on task performance, we repeat the trial without any inter-robot communication to obtain the value $\gamma(\text{NoCom}) = \gamma_A(\text{NoCom}) + \gamma_C(\text{NoCom})$. We then compute the difference in action reward earned during the two trials, that is,

$$\delta(\chi) = \gamma_A(\chi) - \gamma_A(\text{NoCom}). \tag{5.2}$$

We normalize this value by dividing by the corresponding value for the method AllCom, in which agents automatically share all observations. This normalized value is given by

$$\hat{\delta}(\chi) = \frac{\delta(\chi)}{\delta(\text{AllCom})}. \tag{5.3}$$

In other words, $\hat{\delta}(\chi)$ measures how much the communication of method χ helps task performance relative to the full communication baseline method.

For each method, we run the same set of 1000 trials. Each data point represents the mean of $\hat{\delta}(\chi)$ for these trials, and the error bars represent one standard error on the mean. In our experiments, we set OCBC computational budget at $B = 1000$ and maintain beliefs

consisting of $M = 50$ particles.

5.4.2 Fixed-Cost and Proportional-Cost Communication

To compare OCBC to existing work [92], we present two variants *OCBC, Fixed-Cost* and *OCBC, Proportional-Cost*.

In *OCBC, Proportional-Cost*, the ego-agent has no limit on its available bandwidth (i.e. $\beta_i \to \infty$), but each communicated observation incurs some cost (i.e. $R_C(\cdot) > 0$). This causes the function ASSEMBLE of Algorithm 5.4 to return all observations with positive expected reward regardless of how large the resulting message is. Although *OCBC, Proportional-Cost* is a variant of OCBC, it still includes both of our primary contributions (i.e. optimizing message content and evaluating messages with forward simulation).

By contrast, *OCBC, Fixed-Cost* only includes the latter contribution (i.e. evaluating messages with forward simulation). It uses the same sampling method as OCBC, but only evaluates messages containing all available observations.

We compare the performance of these methods to that of *ConTaCT*, a method introduced by Unhelkar and Shah [92]. Our method has three key distinctions from *ConTaCT*. First, the *ConTaCT* ego-agent maintains only a single estimate of the transition function and of its teammates, whereas OCBC maintains belief distributions over these values (see Section 5.3.4). Second, *ConTaCT* estimates the no-communication reward by computing the expected reward of an agent's previously declared policy in the ego-agent's updated transition function estimate. This fixed-policy evaluation does not consider the future observations and associated re-planning of other agents, which OCBC models through forward simulation (see Section 5.3.2). Finally, *ConTaCT* reasons only about *when* to communicate, whereas OCBC also considers *what* to communicate, which allows OCBC to operate under bandwidth constraints (see Section 5.3.1).

ConTaCT belongs to the *Fixed-Cost* paradigm and thus reasons only about *when* to communicate. The primary distinction between *OCBC, Fixed-Cost* and *ConTaCT* is our forward simulation method. Unlike *ConTaCT*, our method considers the future observations and re-planning of other agents as well as the uncertainty in the ego-agent's beliefs.

To compare the cost-based methods, we vary the modeled cost of communication in the range $0.01 \leq R_C(\cdot) \leq 20$ and measure the resulting task performance and proportion of observations communicated. Figure 5.3 shows the results of this experiment with the measured number of communications on the horizontal axis and the measured task per-

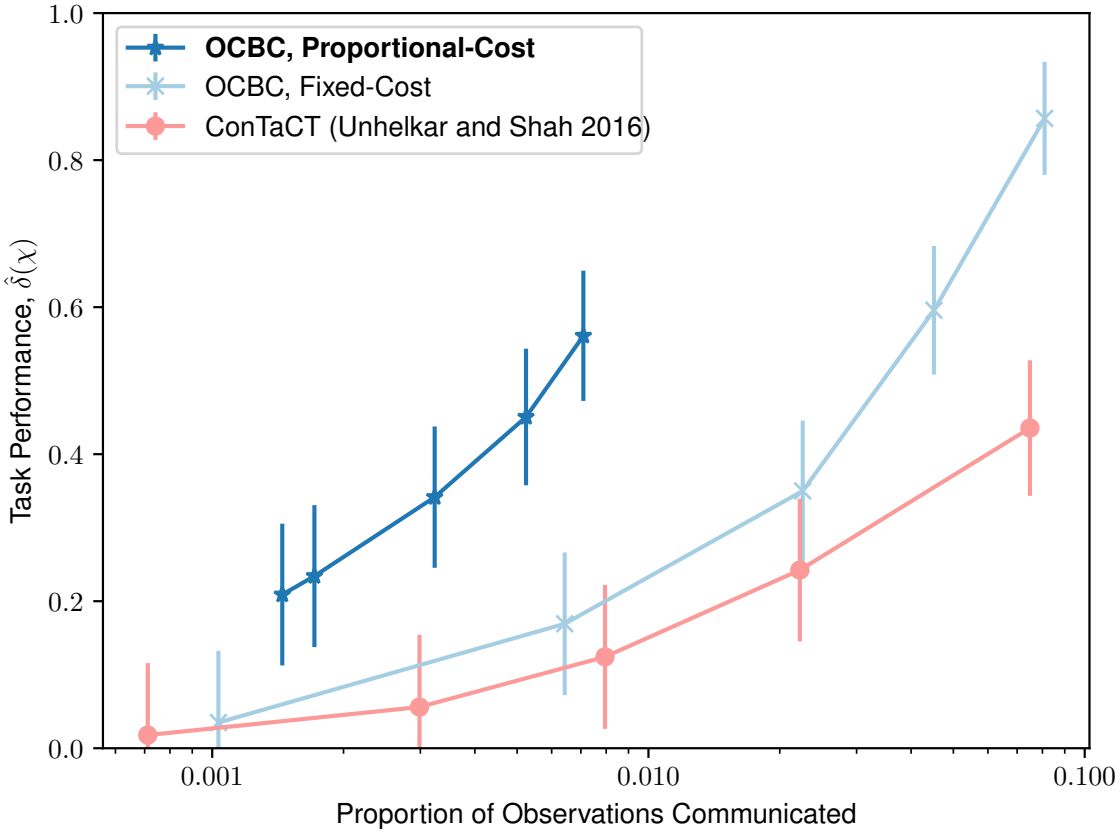

Figure 5.3: Effect of communication on task performance as a function of the proportion of observations communicated. To achieve the different levels of communication, we vary the modeled cost of communication within each method. As the cost of communication decreases, the methods communicate more often and thus achieve higher task performance. Compared to previous work (*ConTaCT*), our full method (*OCBC, Proportional-Cost*) yields higher task performance for a given level of communication. Furthermore, to achieve the same level of task performance, our full method communicates more than an order of magnitude less.

formance on the vertical axis. Note that the controlled parameter in the experiment is the communication cost, and both the number of communications and task performance are measured values. When the modeled cost of communication is high, all methods communicate little, leading to low task performance. As the modeled cost parameter decreases, the amount of communication and task performance both increase.

Our partial method, *OCBC, Fixed-Cost*, performs similarly to *ConTaCT* at low levels of communication, but outperforms *ConTaCT* as the amount of communication increases. At the top right portion of the figure, *OCBC, Fixed-Cost* communicates less than 10 percent of its observations while achieving approximately 85 percent of the task performance of the method that communicates all observations.

Our full method, *OCBC, Proportional-Cost*, outperforms *ConTaCT* across the entire tested range of communication levels. For a given level of communication, *OCBC, Proportional-Cost* achieves higher task performance compared to *ConTaCT*. Furthermore, to achieve the same level of task performance, our method uses more than an order of magnitude fewer communications. At the extreme end of its range, *OCBC, Proportional-Cost* achieves approximately 55 percent of the task performance of the full communication method with less than 1 percent of the communications.

5.4.3 Fixed-Bandwidth Communication

We now evaluate OCBC performance in the *Fixed-Bandwidth* paradigm. We fix the bandwidth available to the robots β and vary the number of robots sharing that bandwidth. The robots use a round-robin scheduling method, with each robot i having bandwidth allocation $\beta_i(t)$ at time step t. As the number of agents sharing the bandwidth increases, the bandwidth allocation per agent decreases. We evaluate OCBC against a baseline method that randomly selects observations.

Figure 5.4 shows the effect of communication on task performance for variable-size robot teams. The total bandwidth available in all experiments is $\beta = 4$, meaning that the team can communicate up to 4 observations at each time step.

We first point out the general trend present in the methods: as more robots share the same amount of bandwidth, task performance decreases. This may be surprising at first since the same number of observations are still transmitted at each time step. However, the value of communications varies, and the round-robin bandwidth allocation cannot account for this. As a part of a large team sharing a small amount of bandwidth, a robot that makes

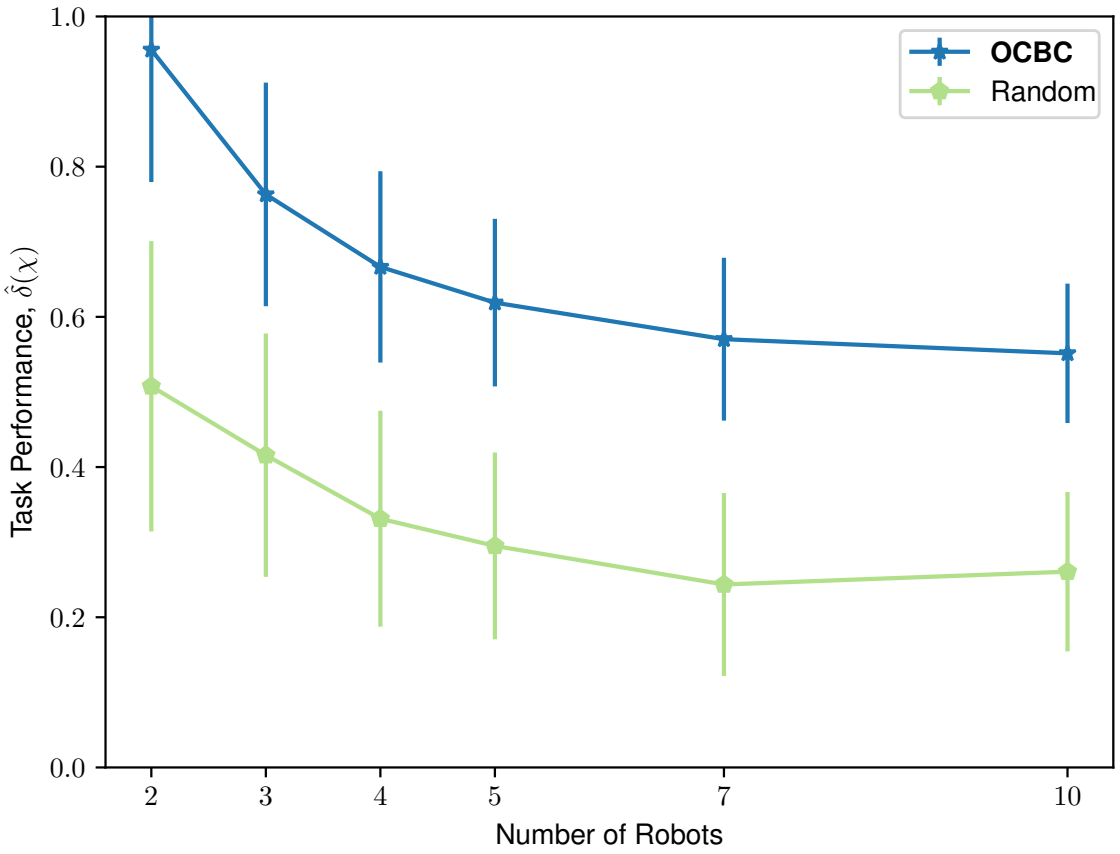

Figure 5.4: Effect of communication on task performance as a function of the number of simulated robots sharing a fixed amount of bandwidth. As team size increases, each robot has less opportunities to communicate, leading to decreased performance. Our method (*OCBC*), achieves better task performance compared to a randomized baseline method.

an interesting observation may have to wait for its turn to share that observation.

We observe that across the range of team sizes, our method outperforms the *Random* baseline method. Furthermore, our method achieves normalized task performance close to 1 in the 2-robot experiment. This means that, in that experiment, our method has similar task performance to the baseline method sharing all observations without a bandwidth constraint. Our method achieves this performance while communicating at least 50 percent fewer observations.

We also point out that this experiment features simulations with up to 10 agents in an environment with 100 states and 4 available actions. This is one of the largest-scale experiments to date in the communication decision-making literature (c.f. [4, 92, 97]), demonstrating the scalability of our method.

5.5 Summary

In this chapter, we proposed OCBC, an approach to Optimizing Communication under Bandwidth Constraints. OCBC uses forward simulation to evaluate possible communication actions and incorporates those evaluations into a bandit-based combinatorial optimization algorithm that computes an approximately optimal set of observations to communicate. OCBC is designed for collaborative multi-agent problems with a deterministic but unknown transition function, which includes tasks where multiple robots operate in previously unexplored terrain. We showed that OCBC outperforms its closest existing competitor at a simulated multi-robot navigation task, achieving higher task performance while communicating more than an order of magnitude less.

CHAPTER 6

Conclusions and Future Directions

6.1 Conclusions

A team of multiple robots collaborating to accomplish a task can be immensely powerful. Communication is the key to unleashing this power, allowing the team to become something that is greater than the sum of its individual parts. However, robust and reliable communication remains highly challenging in practice. As such, it is critical that roboticists place emphasis on the further study of communication methods for multi-robot systems, as we have in this dissertation.

In particular, we encourage the *full-stack* approach to improving multi-robot communication illustrated in Figure 6.1. A traditional network stack consists of multiple layers, each of which is responsible for a particular aspect of communication. In a robotics context, one can think of the top-most (application) layer as deciding *what* a robot communicates while the lower layers determine *how* that communication actually occurs. We advocate for research that addresses both of these questions, investigating improvements across all layers of the network stack. Our own research in this dissertation has followed this approach, with techniques at both the application (Chapter 5) and transport layers (Chapters 3 and 4).

A full-stack approach is important for multiple reasons. The lower layers of a traditional network stack (e.g. IEEE 802.11 with TCP/IP) have been designed to work in a range of conditions to support traffic over the global Internet. However, mobile multi-robot systems impose demands on their network infrastructure that differ from those of the general Internet. It is therefore important to look for ways to improve upon general-purpose network protocols with solutions suitable for mobile ad hoc networks. Chapter 4 provides an example of this idea, replacing a standard protocol (TCP) with one tailored to the unique challenges of mobile multi-robot systems (HBAEC).

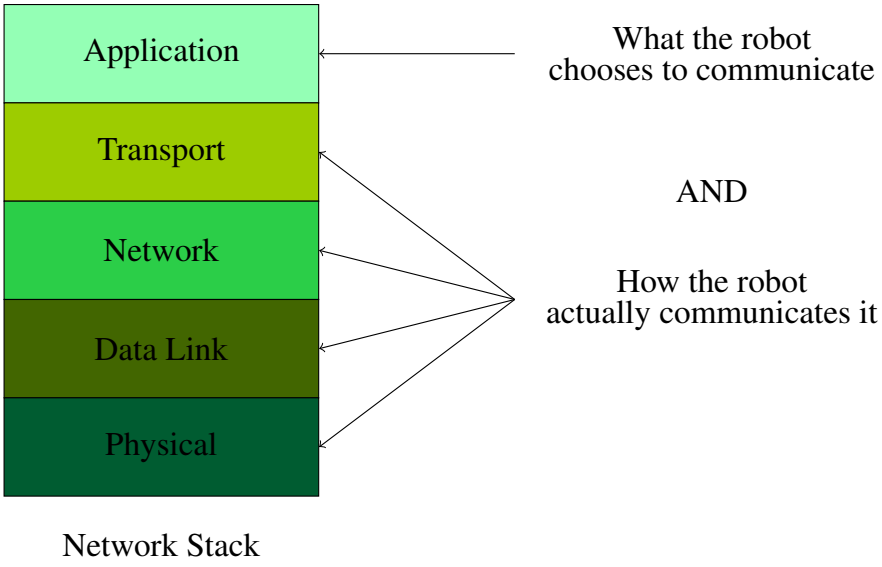

Figure 6.1: Illustration of a *full-stack* approach to improving multi-robot communication

Second, it is important that robots be able to reason at the application layer about what information they should pass over the network. Regardless of what improvements are made at lower layers of the stack to support communication, there will always be limits on how much data can be passed over a wireless network. Hardware may advance and provide greater bandwidth capabilities, but robot sensors will likewise increase the amount of data they take in. It will therefore remain important for robots to make decisions about what information to communicate with their teammates. General solutions to multi-agent communication decision-making are highly intractable and will likely remain so given the great difficulty of the problem. We encourage roboticists to consider how to reign in this complexity to offer scalable solutions for more specific problems, as we did in Chapter 5.

As we have argued so far, the full-stack approach is beneficial by providing improved capabilities at individual layers. However, a final benefit of this approach comes when roboticists look for opportunities to build cross-layer solutions. Traditional networks consider each layer to be self-contained, a modular approach that allows the modern Internet to connect billions of heterogeneous devices. In many cases, though, the roboticist has the unique opportunity to work on a closed system with complete control over the entire network stack. A higher-level application can provide its Quality of Service (QoS) needs to lower-layer protocols, or the lower layers can determine the state of the network and provide that information to the higher-level application to aid in communication decisions. Chapter 3 serves as an example of this approach, where latency tolerance (a characteristic

specified by the application layer) allows a transport layer protocol to operate more reliably and efficiently.

Finally, we argue that adaptivity is key in any approach to improving multi-robot communication. Network conditions are continually changing as robots move through their environment, leading to variable levels of packet loss and throughput. A communication protocol must be able to adapt to these changing conditions, providing the reliable performance needed to support the objectives of the team.

6.2 Contributions

This dissertation included the following contributions.

In Chapter 3, we described a mechanism by which robot systems can exchange tolerance to latency for improvements in the probability of successful packet delivery. We introduced an algorithm that determines the minimum amount of redundancy to transmit to keep the probability of unrecoverable packet loss sufficiently small, an algorithm that depends on a technique to estimate the distribution of possible instantaneous packet loss rates from historical packet loss measurements. We showed that the overall system successfully exchanges latency tolerance for improved delivery performance on a real-world mobile robotic platform.

In Chapter 4, we generalized the optimization technique of Chapter 3 for use with erasure codes with probabilistic decoding characteristics, which are themselves suited to encoding large quantities of data. We presented a technique that uses kernel density estimation to estimate the distribution of instantaneous packet loss based on historical packet loss measurements. We described a repeatable evaluation that uses packet traces collected from a real-world robotic network to drive simulations.

In Chapter 5, we provided a mechanism for making communication decisions under bandwidth constraints in collaborative, multi-robot settings. This mechanism uses forward simulations to evaluate potential communication actions directly in terms of their expected effect on reward. We described a bandit-based method that optimizes message content based on the results of these forward simulations.

6.3 Future Directions

In this section, we identify ways in which future researchers might extend or improve upon the methods of this dissertation. This list is by no means exhaustive; rather, it is meant to serve as a useful starting point for someone looking to carry these ideas further.

Broadcasting and Adaptive Erasure Coding (AEC)

The AEC methods of Chapters 3 and 4 are primarily intended for communication between a source node and a single destination node. However, a node in a robotic network may wish to communicate with many destination nodes at once. It is therefore desirable to extend the AEC methods to handle multiple destinations. Consider a redefinition of the optimization objective to be the minimization of encoding strength such that the probability of decoding failure at any destination node was sufficiently small. This extension would be trivial in the case that all destination nodes were a single hop from the source node. However, the problem becomes highly non-trivial if some destination nodes are multiple hops away and other destination nodes route the data to them. This would require knowledge of the network topology to be incorporated into the encoding strength optimization process. Such a problem bears some similarity to network coding [19].

Extending OCBC to Other Communication Settings

As presented in Chapter 5, the Optimizing Communication under Bandwidth Constraints (OCBC) algorithm assumes that inter-agent communication is broadcast-based and reliable. That is, the algorithm assumes that once the ego-agent decides to communicate a particular message, all other agents will receive that message and incorporate its contents into their belief. This assumption is common in the communication decision-making literature, but it does not accord with the unreliable nature of the networks of mobile multi-robot systems. It is therefore desirable to extend OCBC to relax the reliable communication assumption. The formulation of OCBC provides a straightforward way to do so. Because OCBC relies on sampling-based methods, one could incorporate the expected packet loss rate (e.g. as measured with the approach of Section 4.3) into the sampling procedure. More specifically, the particle filter method of Section 5.3.4 could include particles representing scenarios in which the modeled teammate did not receive the transmitted message. These

lost-packet particles would be identical to the scenario in which the ego-agent did not communicate with the teammate at all.

Additionally, the broadcast communication setting of OCBC could also be generalized to a unicast setting. OCBC is intended for factored multi-robot problems, where coordination is not a concern. Therefore, communicating information with individual robots is viable. For example, if the evaluation process finds that a particular observation is useful for one teammate but no others, that observation could be unicast to the teammate, likely imposing a lower cost on the network [7].

Both extensions, while straightforward to implement, present rich opportunities for empirical evaluation.

Exploiting the Sequential Nature of OCBC to Re-Use Computation

OCBC is intended to be used in sequential decision-making problems (see Section 5.3.3). The current formulation begins the process of evaluating communication decisions anew at each time step. However, it is likely that the evaluations at different time steps are dependent. For example, when the ego-agent evaluates a message to be uninteresting at one time step, that message is unlikely to be useful at the next. A mechanism for re-using some of the computation of previous time steps could greatly reduce computational burden of OCBC, allowing it to be used in larger-scale problems.

Incorporating Queries into OCBC

As presented in Chapter 5, OCBC reasons only about decisions to share observations with teammates. Such decisions depend on the models OCBC maintains of other teammates and their beliefs about the world. It is apparent that the accuracy of the evaluation of communication decisions depends greatly on the accuracy of teammate models. If the teammate models become too diffuse and uncertain, the evaluation process may be compromised.

A solution to this problem might be to introduce the capability of querying into OCBC. This would be similar to the method of Best et al. [6], in which the ego-agent requests information from teammates once its internal models of them become sufficiently uncertain.

Directed Sampling in OCBC

OCBC relies on sampling to evaluate communication decisions. Specifically, OCBC samples from the ego-agent's beliefs about the models of teammates and the transition function. OCBC draws samples with weights corresponding to the ego-agent's beliefs. Samples that are interesting from a communication decision-making perspective are distributed sparsely throughout this sample space. Rather than spending significant portions of the computational budget on uninteresting portions of the sample space, it might make sense to try to direct the sampling process. The idea here would be to search through the sample space for interesting samples, similar to the method Mehta et al. [57] introduced to find high-cost samples in robot navigation. Such an approach might lead to efficient communication decisions on tight computational budgets.

BIBLIOGRAPHY

[1] Paramasiven Appavoo, Ebram Kamal William, Mun Choon Chan, and Mobashir Mohammad. Indriya2: A heterogeneous wireless sensor network (WSN) testbed. In *Proceedings of the EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2018.

[2] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. Radio link quality estimation in wireless sensor networks. *ACM Transactions on Sensor Networks*, 8(4):1–33, 2012.

[3] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, December 2004.

[4] Raphen Becker, Alan Carlin, Victor Lesser, and Shlomo Zilberstein. Analyzing myopic approaches for multi-agent communication. *Computational Intelligence*, 25 (1), 2009.

[5] Graeme Best, Oliver M Cliff, Timothy Patten, Ramgopal R Mettu, and Robert Fitch. Dec-MCTS: Decentralized planning for multi-robot active perception. *International Journal of Robotics Research*, 1, 2018.

[6] Graeme Best, Michael Forrai, Ramgopal Mettu, and Robert Fitch. Planning-aware communication for decentralised multi-robot coordination. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018.

[7] Giuseppe Bianchi. IEEE 802.11-saturation throughput analysis. *IEEE Communications Letters*, 2(12):318–320, 1998.

[8] John Charles Bicket. *Bit-rate selection in wireless networks*. PhD thesis, Massachusetts Institute of Technology, 2005.

[9] Mike Bishop. Hypertext Transfer Protocol (HTTP) over QUIC. Internet-draft, IETF Secretariat, October 2018. URL http://www.ietf.org/internet-drafts/draft-ietf-quic-http-16.txt.

[10] Scott Burleigh, Adrian Hooke, Leigh Torgerson, Kevin Fall, Vint Cerf, Bob Durst, Keith Scott, and Howard Weiss. Delay-tolerant networking: an approach to interplanetary Internet. *IEEE Communications Magazine*, 41(6):128–136, June 2003.

[11] Jesse Butterfield, Odest Chadwicke Jenkins, David M Sobel, and Jonas Schwertfeger. Modeling aspects of theory of mind with Markov random fields. *International Journal of Social Robotics*, 1(1):41–51, 2009.

[12] John Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. *ACM SIGCOMM Computer Communication Review*, 28(4):56–67, 1998.

[13] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1):427–438, 2013.

[14] Alan Carlin and Shlomo Zilberstein. Myopic and non-myopic communication under partial observability. In *Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 331–338. Institute of Electrical and Electronics Engineers, 2009.

[15] Alan Carlin and Shlomo Zilberstein. Value of communication in decentralized POMDPs. In *Proceedings of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, pages 16–21, 2009.

[16] Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Sanadidi, and Ren Wang. TCP Westwood: End-to-end congestion control for wired/wireless networks. *Wireless Networks*, 8(5):467–479, 2002.

[17] Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen. Combinatorial pure exploration of multi-armed bandits. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 379–387, 2014.

[18] Tzi-Cker Chiueh, Rupa Krishnan, Pradipta De, and Jui-Hao Chiang. A networked robot system for wireless network emulation. In *Proceedings of the International Conference on Robot Communication and Coordination*, 2007.

[19] Philip Chou and Yunnan Wu. Network coding for the internet and wireless networks. *IEEE Signal Processing Magazine*, 24(5):77–85, 2007.

[20] David Clark. The design philosophy of the DARPA Internet protocols. *ACM SIGCOMM Computer Communication Review*, 18(4):106–114, 1988.

[21] Bastian Degener, Sandor Fekete, Barbara Kempkes, and Friedhelm Meyer Auf Der Heide. A survey on relay placement with runtime and approximation guarantees. *Computer Science Review*, 5(1):57–68, 2011.

[22] Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal Ananda. Indriya: A low-cost, 3D wireless sensor network testbed. In *EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, pages 302–316. Springer, 2011.

[23] Jonathan Fink. *Communication for Teams of Networked Robots*. PhD thesis, University of Pennsylvania, 2011.

[24] Jonathan Fink, Nathan Michael, Alex Kushleyev, and Vijay Kumar. Experimental characterization of radio signal propagation in indoor environments with application to estimation and control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2834–2839. Institute of Electrical and Electronics Engineers, October 2009.

[25] Eduardo Feo Flushing, Michal Kudelski, Luca Gambardella, and Gianni Di Caro. Spatial prediction of wireless links and its application to the path control of mobile robots. In *Proceedings of the IEEE International Symposium on Industrial Embedded Systems*, pages 218–227. Institute of Electrical and Electronics Engineers, 2014.

[26] Jakob Foerster, Yannis Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 2137–2145, 2016.

[27] Matthew Giamou, Kasra Khosoussi, and Jonathan P How. Talk resource-efficiently to me: Optimal communication planning for distributed slam front-ends. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018.

[28] Stephanie Gil. *Adaptive communication networks for heterogeneous teams of robots*. PhD thesis, Massachusetts Institute of Technology, 2014.

[29] Stephanie Gil, Swarun Kumar, Dina Katabi, and Daniela Rus. Adaptive communication in multi-robot systems using directionality of signal strength. *International Journal of Robotics Research*, 34(7):946–968, 2015.

[30] Claudia Goldman and Shlomo Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.

[31] Alejandro Gonzalez-Ruiz, Alireza Ghaffarkhah, and Yasamin Mostofi. A comprehensive overview and characterization of wireless channels for networked robotic and control systems. *Journal of Robotics*, (19):1–19, 2011.

[32] Luigi A. Grieco and Saverio Mascolo. Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. *ACM SIGCOMM Computer Communication Review*, 34(2):25–38, April 2004.

[33] Esten Ingar Grøtli and Tor Arne Johansen. Motion-and communication-planning of unmanned aerial vehicles in delay tolerant network using mixed-integer linear programming. *Modeling, Identification and Control*, 2016.

[34] Yunhong Gu and Robert Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks*, 51(7):1777–1799, 2006.

[35] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[36] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. *ACM SIGCOMM Computer Communication Review*, 41(4), 2010.

[37] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun-chan Lan, Ya Xu, Wei Ye, Deborah Estrin, and Ramesh Govindan. Effects of detail in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 3–11, 2001.

[38] Stephen Hemminger. Network emulation with NetEm. In *Linux Conf AU*, pages 18–23, 2005.

[39] Guido Hiertz, Dee Denteneer, Sebastian Max, Rakesh Taori, Javier Cardona, Lars Berlemann, and Bernhard Walke. IEEE 802.11s: The WLAN mesh standard. *IEEE Transactions on Wireless Communications*, 17(1):104–111, 2010.

[40] Guido R. Hiertz, Sebastian Max, Zhao Rui, Dee Denteneer, and Lars Berlemann. Principles of IEEE 802.11s. In *Proceedings of the International Conference on Computer Communications and Networks*, 2007.

[41] Geoffrey Hollinger, Sunav Choudhary, Parastoo Qarabaqi, Christopher Murphy, Urbashi Mitra, Gaurav Sukhatme, Milica Stojanovic, Hanumant Singh, and Franz Hover. Communication protocols for underwater data collection using a robotic sensor network. In *Proceedings of the IEEE GLOBECOM Workshop*, 2011.

[42] Mong-ying A. Hsieh, Anthony Cowley, Vijay Kumar, and Camillo Taylor. Maintaining network connectivity and performance in robot teams. *Journal of Field Robotics*, 25(1-2):111–131, 2008.

[43] Po-Chang Huang, Kuo-Chih Chu, Hsiang-Fu Lo, Wei-Tsong Lee, and Tin-Yu Wu. A novel adaptive FEC and interleaving architecture for H.264/SVC wireless video transmission. In *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 989–992, September 2009.

[44] Abdallah Kassir. *Communication Efficiency in Information Gathering through Dynamic Information Flow*. PhD thesis, University of Sydney, 2014.

[45] Abdallah Kassir, Robert Fitch, and Salah Sukkarieh. Communication-aware information gathering with dynamic information flow. *International Journal of Robotics Research*, 34(2):173–200, 2015.

[46] Abdallah Kassir, Robert Fitch, and Salah Sukkarieh. Communication-efficient motion coordination and data fusion in information gathering teams. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5258–5265. Institute of Electrical and Electronics Engineers, 2016.

[47] Wolfgang Kiess and Martin Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5(3):324–339, 2007.

[48] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[49] Michal Kudelski, Luca Gambardella, and Gianni Di Caro. RoboNetSim: An integrated framework for multi-robot and network simulation. *Robotics and Autonomous Systems*, 61(5):483–496, 2013.

[50] Michal Kudelski, Luca Gambardella, and Gianni Di Caro. A mobility-controlled link quality learning protocol for multi-robot coordination tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5024–5031. Institute of Electrical and Electronics Engineers, May 2014.

[51] Christopher J. Lowrance and Adrian P. Lauf. Link estimation in robot networks for multi-radio control and range extension. *Statistical Science*, 85(2):245–275, 2017.

[52] Christopher J. Lowrance, Adrian P. Lauf, and Mehmed Kantardzic. A fuzzy-based machine learning model for robot prediction of link quality. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1–8. Institute of Electrical and Electronics Engineers, 2016.

[53] David MacKay. Fountain codes. *IEE Proceedings - Communications*, 152(6):1062, 2005.

[54] Ryan J Marcotte and Edwin Olson. Adaptive forward error correction with adjustable-latency QoS for robotic networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5283–5288. Institute of Electrical and Electronics Engineers, 2016.

[55] Ryan J Marcotte, Xipeng Wang, and Edwin Olson. AprilFEC: Real-time channel estimation and adaptive forward error correction. In *Proceedings of the RSS Workshop on Robot Communication in the Wild*, 2017.

[56] Petar Maymounkov. Online codes. Technical report, New York University, 2002.

[57] Dhanvin Mehta, Gonzalo Ferrer, and Edwin Olson. Backprop-MPDM: Faster risk-aware policy evaluation through efficient gradient optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1740–1746. Institute of Electrical and Electronics Engineers, 2018.

[58] Nathan Michael, Michael Zavlanos, Vijay Kumar, and George Pappas. Maintaining connectivity in mobile robot networks. In *Experimental Robotics*. Springer Berlin Heidelberg, 2009.

[59] Andreas Molisch. *Wireless Communications*. John Wiley & Sons, 2 edition, 2011.

[60] Yasamin Mostofi, Mehrzad Malmirchegini, and Alireza Ghaffarkhah. Estimation of communication signal strength in robotic networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[61] Brian Noble, Mahadev Satyanarayanan, Giao Nguyen, and Randy Katz. Trace-based mobile network emulation. In *ACM SIGCOMM Computer Communication Review*, volume 27, pages 51–61. ACM, 1997.

[62] Edwin Olson, Johannes Strom, Ryan Morton, Andrew Richardson, Pradeep Ranganathan, Robert Goeddel, Mihai Bulic, Jacob Crossman, and Bob Marinier. Progress towards multi-robot reconnaissance and the MAGIC 2010 competition. *Journal of Field Robotics*, 29(5):762–792, September 2012.

[63] Edwin Olson, Johannes Strom, Robert Goeddel, Ryan Morton, Pradeep Ranganathan, and Andrew Richardson. Exploration and mapping with autonomous robot teams. *Communications of the ACM*, 2013.

[64] William Wesley Peterson and Daniel T Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, 1961.

[65] David Pynadath and Milind Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, June 2002.

[66] Jalaluddin Qureshi, Chuan Heng Foh, and Jianfei Cai. Primer and recent developments on fountain codes. *Recent Advances in Communications and Networking Technology*, 2(1):2–11, 2013.

[67] Irving Reed and Xuemin Chen. *Error-control coding for data networks*, volume 508. Springer Science & Business Media, 1999.

[68] Irving Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of Industrial and Applied Mathematics*, 1960.

[69] Wei Ren, Randal W Beard, and Ella M Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, 2007.

[70] Iain E Richardson. *The H 264 advanced video compression standard*. John Wiley & Sons, 2011.

[71] George Riley and Thomas Henderson. The ns-3 network simulator. In *Modeling and Tools for Network Simulation*, pages 15–34. Springer, 2010.

[72] Vincent Roca and Christoph Neumann. Design, evaluation and comparison of four large block FEC codecs, LDPC, LDGM, LDGM staircase and LDGM triangle, plus a Reed-Solomon small block FEC codec. Technical report, INRIA, 2006.

[73] Martin Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 2007.

[74] Jim Roskind. QUIC (Quick UDP Internet Connections): Multiplexed stream transport over UDP. Technical report, Google, 2013.

[75] Maayan Roth. *Execution-time communication decisions for coordination of multi-agent teams*. PhD thesis, Carnegie Mellon University, 2007.

[76] Maayan Roth, Reid Simmons, and Manuela Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, 2005.

[77] Maayan Roth, Reid Simmons, and Manuela Veloso. What to communicate? execution-time decision in multi-agent pomdps. In *Distributed Autonomous Robotic Systems*, pages 177–186. Springer Japan, 2006.

[78] Murat Senel, Krishna Chintalapudi, Dhananjay Lal, Abtin Keshavarzian, and Edward J. Coyle. A kalman filter based link quality estimation scheme for wireless sensor networks. In *Proceedings of the IEEE Global Telecommunications Conference*, pages 875–880. Institute of Electrical and Electronics Engineers, 2007.

[79] Amin Shokrollahi and Michael Luby. Raptor codes. *Foundations and Trends in Communications and Information Theory*, 6:213–322, 2011.

[80] Amin Shokrollahi, Soren Lassen, and Richard Karp. Systems and processes for decoding chain reaction codes through inactivation, 2005. US Patent 6,856,263.

[81] Brooke Shrader and Anthony Ephremides. Random access broadcast: Stability and throughput analysis. *IEEE Transactions on Information Theory*, 53(8):2915–2921, 2007.

[82] Aloizio P Silva, Scott Burleigh, Celso M Hirata, and Katia Obraczka. A survey on congestion control for delay and disruption tolerant networks. *Ad Hoc Networks*, 25: 480–494, 2015.

[83] Bernard Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, 1986.

[84] Thrasyvoulos Spyropoulos, Rao Naveed Rais, Thierry Turletti, Katia Obraczka, and Athanasios Vasilakos. Routing for Disruption Tolerant Networks: Taxonomy and Design. *Wireless Networks*, 16(8):2349–2370, 2010.

[85] Johannes Strom and Edwin Olson. Multi-sensor ATTenuation Estimation (MATTE): Signal-strength prediction for teams of robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[86] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 2244–2252, 2016.

[87] Ian Swett. QUIC FEC v1. Technical report, Google, 2016.

[88] Michael Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[89] Danilo Tardioli, Alejandro R. Mosteo, Luis Riazuelo, Jose L. Villarroel, and Luis Montano. Enforcing networking connectivity in robot team missions. *International Journal of Robotics Research*, 2010.

[90] Onur Tekdas, Wei Yang, and Volkan Isler. Robotic routers: Algorithms and implementation. *International Journal of Robotics Research*, 29(1):110–126, 2010.

[91] Ajay Tirumala, Feng Qin, Jon Dugan, and Jim Ferguson. iPerf: TCP/UDP bandwidth measurement tool, 2005.

[92] Vaibhav Unhelkar and Julie Shah. ConTaCT: Deciding to communicate during time-critical collaborative tasks in unknown, deterministic domains. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 2544–2550, 2016.

[93] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. Erasure-coding based routing for opportunistic networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*, pages 229–236. ACM, 2005.

[94] Simon Williamson, Enrico Gerding, and Nick Jennings. A principled information valuation for communications during multi-agent coordination. In *Proceedings of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, 2008.

[95] Simon Williamson, Enrico Gerding, and Nicholas Jennings. Reward shaping for valuing communications during multi-agent coordination. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, 2009.

[96] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, pages 459–471, April 2013.

[97] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175(2):487–511, 2011.

[98] Maya Yajnik, Sue Moon, Jim Kurose, and Don Towsley. Measurement and modelling of the temporal dependence in packet loss. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies*, 1999.

[99] Yuan Yan and Yasamin Mostofi. Co-optimization of communication and motion planning of a robotic operation under resource constraints and in fading environments. *IEEE Transactions on Wireless Communications*, 12(4):1562–1572, 2013.

[100] Michael M Zavlanos and George J Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, 2008.

[101] Artur Zolich, David Palma, Kimmo Kansanen, Kay Fjørtoft, João Sousa, Karl H Johansson, Yuming Jiang, Hefeng Dong, and Tor A Johansen. Survey on communication and networks for autonomous marine systems. *Journal of Intelligent and Robotic Systems*, pages 1–25, 2018.