UNKNOWN PARTICIPANTS, CHANGING TOPOLOGIES, AND FAULTY DEVICES: NEW BOUNDS FOR DISTRIBUTED ALGORITHMS IN CHALLENGING WIRELESS NETWORK MODELS

A Dissertation submitted to the Faculty of the Graduate School of Arts and Sciences of Georgetown University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

By

Brian Alexander Weaver, B.S.

Washington, DC
January 9, 2023
Copyright © 2023 by Brian Alexander Weaver
All Rights Reserved
In this thesis, we study several distributed problems in complex algorithm models designed to represent certain challenges associated with wireless networks. We begin by studying the contention resolution problem in a model we introduce in which the exact size of the network is unknown, but a probability distribution over the possible sizes is provided as input to the algorithm. We study this problem with and without collision detection, in both cases proving upper and lower bounds for the time complexity of the problem, revealing an interesting dependence on the Shannon entropy of the provided distribution.

We then turn our attention to the one-shot gossip and capacity problems. We study both of these problems in the mobile telephone model (MTM), a recently introduced peer-to-peer model designed to capture the constraints and capabilities of wireless mesh networks created by existing smartphone communication technologies. We once again prove upper and lower bounds for the complexities of these problems. We then extend the MTM by introducing a new model which eliminates the assumption of synchronized rounds, proving an upper bound for the one-shot gossip problem in this harder asynchronous model.

Finally, we explore two classic graph structuring problems, the maximal independent set (MIS) and maximal matching (MM) problems, in variations of the well-studied broadcast-Congest model which allow for the possibility of different types of device failures. We first introduce a model which includes crash failures. This model divides an algorithm execution into two stages. During the first stage, preparation for the crashes can
be made. In the second stage, which begins after the crashes occur, the algorithm attempts to compute a valid structure with respect to the surviving subgraph. In this model, we show upper and lower bounds for how quickly an MIS and MM can be repaired in the second stage, given some amount of precomputation performed in the first. We then introduce a model which allows for a bounded number of byzantine failures. In this model, we define what it means to correctly compute an MIS and provide upper and lower bounds on the time required to do so with respect to the number of failures.

**Index words:** Distributed algorithms, wireless networks, contention resolution, one-shot gossip, capacity, maximal independent set, maximal matching
ACKNOWLEDGMENTS

As I reach the end of my doctoral program, there are a few people that I would like to thank in particular.

First and foremost, I would like to thank my advisors, Dr. Cal Newport and Dr. Nitin Vaidya, for their gracious support during my time at Georgetown. It has truly been a privilege to have been mentored by such kind and talented people for the past few years, and I will miss working with them both.

I would also like to thank my committee members, Dr. Bala Kalyanasundaram and Dr. Justin Thaler. I am very grateful for the time they have given to be a part of my committee and their help in improving my thesis.

Last but not least I would like to thank my wife, Ashley. Not only did she inspire me to pursue this path for which I am grateful, throughout it she has offered me every conceivable means of support. There is little I could imagine doing without her, but certainly none of this would have been possible if not for her.
FUNDING

This work was funded in part by NSF CCF Awards 1733842 and 1733872 (AiTF: Collaborative Research: Algorithms for Smartphone Peer-to-Peer Networks).
# Table of Contents

**Chapter**

1 Introduction ................................................................. 1  
   1.1 Challenges of Wireless Networks ................................. 3  
      1.1.1 Increased Fault Frequency ................................. 3  
      1.1.2 Complex Communication Topologies ....................... 3  
      1.1.3 Imperfect Knowledge ........................................ 4  
   1.2 Methodology .......................................................... 4  
      1.2.1 Leveraging Imperfect Knowledge in the Data Link Layer 5  
      1.2.2 Using Peer-to-Peer Communication in the Transport Layer 6  
      1.2.3 Improving Fault Tolerance in the Application Layer ... 8  
   1.3 General Preliminaries and Notation .............................. 10  
      1.3.1 Mathematical Preliminaries ............................... 10  
      1.3.2 Graph Preliminaries ........................................ 18  

2 The Data Link Layer ....................................................... 22  
   2.1 Existing Work on Contention Resolution ....................... 25  
   2.2 Summary of New Results ............................................ 27  
      2.2.1 Lower Bounds ............................................... 28  
      2.2.2 Upper Bounds ............................................... 30  
   2.3 Model and Problem Definition .................................... 32  
   2.4 Lower Bounds for the Contention Resolution Problem with Predictions 35  
      2.4.1 Lower Bound for No Collision Detection .................. 36  
      2.4.2 Lower Bound for Collision Detection ..................... 47  
   2.5 Upper Bounds for the Contention Resolution Problem with Predictions 57  
      2.5.1 Upper Bound for No Collision Detection .................. 58  
      2.5.2 Upper Bound for Collision Detection ..................... 61  

3 The Transport Layer ....................................................... 65  
   3.1 Existing Work on Gossip in Peer-to-Peer Networks ............. 68  
   3.2 Summary of New Results ............................................ 72  
      3.2.1 Synchronous One-Shot Gossip .............................. 72  
      3.2.2 Asynchronous One-Shot Gossip ................................ 73  
      3.2.3 All-to-All Capacity ....................................... 74  
   3.3 Synchronous One-Shot Gossip in Peer-to-Peer Networks ........ 76  
      3.3.1 Model and Problem Definition ............................. 76
A wireless network is simply defined as any network of devices where the devices communicate with one another via radio broadcast. This broad definition encompasses many common types of communication methods: such as broadcast radio, Wi-Fi, and Bluetooth technologies. Therefore, this classification can be used to describe an absolutely enormous number of networks, many of which are extremely prevalent in our daily lives and have been for quite some time. Due to their continued ubiquity, these networks have unsurprisingly been a central focus of many fields within computer science since their inception. In this regard the field of distributed algorithm theory is certainly no exception. Work on this topic from the distributed algorithms community dates back nearly 40 years with works like those of Chlamtac and Kutten [18] and Luby [80].

Distributed algorithms researchers study wireless networks by designing and analyzing solutions to abstract problems which approximate the real challenges these networks face (e.g., sending a message from one device to another). The goal of this line of work is to unlock powerful strategies for wireless network protocols which achieve robust theoretical guarantees. However, achieving sufficient mathematical rigor comes at a cost, as it requires us to tame the complexities of an actual wireless network. Theorists accomplish this by applying simplifying assumptions to represent networks using an abstract algorithm model which formalizes properties such as the network topology and the communication capabilities of the devices. Once established, this formal model can be used to achieve algorithmic results with strong theoretical properties.
Using this paradigm, researchers have successfully studied many fundamental distributed problems for the past several decades. However, while many of these problems remain as relevant to wireless networks today as they were forty years ago, the same cannot be said about many of the models which the existing literature uses to study them. Namely, as we will see throughout this thesis, most of these models fail to account for certain key challenges that are posed by many wireless networks today. This divide between theory and practice is due to a combination of the rapid advancement of wireless technologies (e.g., wide-spread support for peer-to-peer networking) as well the comparatively slow process of proving rigorous mathematical results. This divergence has caused many classic results from distributed algorithm theory designed for overly simplified algorithm models to lose their relevancy to the wireless network setting.

It is the goal of this thesis to reduce this gap between wireless networks and distributed algorithms theory by studying fundamental distributed problems in more complex wireless networking models which better resemble true wireless networks. In many situations, this will force us to design new algorithm models in order to account for the unique challenges posed by wireless networks which are often neglected in existing distributed algorithm literature. We will then use these more complicated abstractions to study problems from the field of distributed algorithms theory which we believe have important applications in real wireless networking protocols. For each of the distributed problems we study, we seek to fully characterize its complexity in these challenging models by proving both upper bounds and lower bounds.
1.1 CHALLENGES OF WIRELESS NETWORKS

To frame our discussion, we begin by taking a closer look at several specific challenges posed by modern wireless networks that are not sufficiently modeled or studied in existing distributed algorithms research.

1.1.1 INCREASED FAULT FREQUENCY

Wireless networks are often composed of a large number of small devices, which afford enhanced portability at the cost of decreased robustness. These devices can be disrupted for any number of reasons such as battery depletion, physical damage, and environmental or radio interference. Not only can devices suffer from external disruption, they may also simply disappear from the network due to moving out of range as these devices often rely on local communication technologies which use on short-range radio frequencies.

1.1.2 COMPLEX COMMUNICATION TOPOLOGIES

While wireless networks have historically followed a somewhat centralized paradigm (e.g., where devices connect either to a cell tower or a nearby wired access point, etc.) there has been an increasing shift recently to build these networks using local peer-to-peer (P2P) communication technologies. The reason is that these technologies offer many advantages, namely that they are more robust to the vulnerability and bandwidth constraints which are well-known in traditional centralized infrastructure.

These communication protocols are not without their disadvantages, however, and indeed are generally weaker than many other means of wireless communication. Namely, these technologies restrict devices to only connecting with nearby peers and, in many cases, only a small subset of these available peers. This induces low-degree, multi-hop network communication topologies which can change over time as devices move in and
out of range with one another. These complex and dynamic topologies can significantly complicate the task of disseminating large amounts of information throughout a network.

1.1.3 Imperfect Knowledge

As we alluded to above, wireless networks can undergo unpredictable changes in the graph topology and are subject to strict communication bottlenecks. An additional complication this creates is that it may be difficult for devices to possess any perfect knowledge about relevant network conditions.

Most existing distributed algorithms research accounts for this by adopting the opposite extreme, where no knowledge of the network is attainable a priori. However, this assumption is also simplifying, as it is often the case that there is some degree of partial knowledge which can be learned by the network participants. This poses the intriguing question of how to meaningfully exploit this imperfect knowledge.

1.2 Methodology

In this work, we aim to systematically improve existing distributed algorithm research by analyzing the complexity of well-studied distributed problems in new wireless networking models which capture the challenges identified in the preceding section. As the existing body of distributed algorithms research is extensive, there are many important problems which we could choose to consider in this thesis. In order to identify relevant problems, we therefore consult the Open Systems Interconnection (OSI) model, a universal architecture that categorizes the functions of a communications system into a hierarchy of seven discrete layers (see Table 1.1).

Using this framework, we select three layers to focus on in this work: the data link, transport, and application layers. For each layer, we consider its functionality as described
<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
</tr>
<tr>
<td>6</td>
<td>Presentation</td>
</tr>
<tr>
<td>5</td>
<td>Session</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
</tr>
<tr>
<td>3</td>
<td>Network</td>
</tr>
<tr>
<td>2</td>
<td>Data Link</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
</tr>
<tr>
<td></td>
<td>Interfacing with end-user applications</td>
</tr>
<tr>
<td></td>
<td>Defining schema for data transmission</td>
</tr>
<tr>
<td></td>
<td>Enabling logical interhost connections</td>
</tr>
<tr>
<td></td>
<td>Ensuring reliable message delivery</td>
</tr>
<tr>
<td></td>
<td>Internetwork packet routing</td>
</tr>
<tr>
<td></td>
<td>Contention resolution and error correction</td>
</tr>
<tr>
<td></td>
<td>Data transmission over physical media</td>
</tr>
</tbody>
</table>

Table 1.1: The Open Systems Interconnection (OSI) model. Categorizes the different functions of the wireless network stack.

by the OSI standard and use this to identify problems from the field of distributed algorithms theory that are relevant to this layer. Then, for each problem, we extend existing algorithm models by introducing a complication motivated by one of the challenges described above.

Below we introduce each of the three layers which will be the focus of this thesis, outlining their core functionalities. For each, we further identify which of the theoretical challenges from Section 1.1 we will consider for that layer, and provide some intuition for the solutions presented in Sections 2, 3, and 4.

1.2.1 Leveraging Imperfect Knowledge in the Data Link Layer

We begin in Chapter 2 by studying the data link layer, which is tasked with a number of low-level functions which include controlling multi-node access to a single shared media (namely, a shared physical communication link). The analogue of this problem in distributed algorithms research is contention resolution. In this problem, an unknown subset of participants from a set of $n$ total participants are connected to a shared channel. Each
round, participants choose whether or not to broadcast a message over this shared channel and the problem is solved in the first round where exactly one participant broadcasts.

As is common in typical distributed algorithms research, most work on the contention resolution problem assumes very limited prior knowledge of the network. In the case of contention resolution, this is in part because perfect knowledge of certain network properties can trivialize the problem. Namely, knowing the number of participants almost exactly (within a constant factor) can reduce the expected time required to solve the problem to only $O(1)$ rounds in expectation (which of course is trivially optimal to solve the problem). However, as mentioned, both of these extremes are unlikely to faithfully represent a realistic wireless network in which some imperfect estimation of this parameter may be attainable by its participants.

For this reason, we extend the existing theory behind the contention resolution problem by designing a model in Section 2.3 which captures this notion of imperfect knowledge of the network size. In this model, we incorporate this knowledge by providing the algorithm as input a probability distribution over the possible network sizes. We then use this model to characterize the impact of this knowledge by proving both lower and upper bounds for the contention resolution problem. First, in Section 2.4 we prove fundamental limits on the time complexity of solving contention resolution given a probabilistic estimate of the network size. Then in Section 2.5, we design algorithms which are able to take advantage of this additional information to more quickly solve the problem.

1.2.2 Using Peer-to-Peer Communication in the Transport Layer

The transport layer, which we explore in Chapter 3, enables end-to-end communication across a network. In distributed algorithms theory, gossip provides a good representative problem for this layer. In this setting, a subset of the network participants begin with some
number of tokens. The goal of the problem is then to somehow distribute these tokens throughout the network.

Arguably, the most relevant communication model for studying gossip in wireless settings are those that explicitly capture peer-to-peer (P2P) connections among a topology that is a prior unknown, and perhaps even changing over time. As we elaborate in Section 3.1, the most common such models are often called "telephone models," as they allow each participant to "make a phone call" and initiate at most one connection per round. This constraint is meant to represent a core restriction of P2P communication which allows each device to only support a constant number of concurrent connections. However, most classic P2P models still allow participants to accept an unbounded number of simultaneous connections which is not possible in certain types of P2P communication technologies, namely those available on commodity smartphones [35]. To study this setting, the recent work of Ghaffari and Newport [57] introduced a P2P networking model called the mobile telephone model (MTM) which restricts participants to a constant number of total connections each round. Although this restriction makes analyzing these smartphone P2P networks much more difficult, some preliminary results for the gossip problem in this setting have been proven (e.g., [85]).

In this thesis, we significantly extend these existing results in several ways. First, in Section 3.3 we study the simplest variation of the gossip problem, known as one-shot gossip (where a finite number of packets must be gossiped), in the MTM. In Section 3.3.2 we prove the first lower bound for this problem and in Section 3.3.3 we provide an improved upper bound over that achieved in [85]. Next in Section 3.4 we extend the MTM itself by removing the assumption of synchronized rounds. This model, which we introduce in Section 3.4.1, is called the asynchronous MTM (aMTM) and it provides a much better abstraction for the capabilities of smartphone P2P networking libraries where it is unreasonable to assume synchronization between devices. This consequently, however, makes
algorithms for this setting far more difficult to analyze. That being said, in Section 3.4.2 we
overcome these challenges to adapt our synchronous one-shot gossip algorithm from Sec-
tion 3.3.3 to the challenging asynchronous setting. Finally, in Section 3.5 we return to the
synchronous MTM to study a more challenging version of the gossip problem called all-to-
all capacity. In this problem, the network participants must instead continuously spread an
infinite stream of packets at the fastest rate possible (called throughput). For this problem,
in Section 3.5.2, we prove a new upper on the fastest throughput achievable by any algo-

rithm in the MTM. We then provide an algorithm in Section 3.5.3 which achieves this
optimal throughput, discussing its implications for the one-shot gossip problem.

1.2.3 Improving Fault Tolerance in the Application Layer

Finally, in Chapter 4 we study the application layer, the highest layer of the OSI architec-
ture. This layer supports several high-level functions, including subroutines that are used
by many different applications. One particular class of subroutines relevant to distributed
algorithms theory is the distributed construction of graph structures, which act as important
backbones for application layer protocols. Here, we focus on two of the most fundamental
graph structures: maximal independent sets (MIS) and maximal matchings (MM).

As we discuss in Section 4.1, the MIS and MM problems are two of the most well-
studied problems in distributed algorithms theory. However, compared to the extensive lit-
erature surrounding these problems, little work has explored these graph structures outside
of classic distributed computation models such as the the LOCAL and CONGEST models.
In this existing work, the emphasis is on how fast these structures can be computed under
mostly ideal conditions (e.g., a stable network with honest participants). Therefore not
much is known about the effect of different types of device failures on these graph struc-
tures, which as we mention in Section 1.1 can occur quite frequently in wireless networks.
In this work we therefore introduce two challenging computation models which are designed to model two common types of device failures. In Section 4.3 we explore the impact of crash failures, where a bounded number of arbitrary participants can suddenly leave the network. In Section 4.3.1 we introduce our crash failure model which separates an algorithm execution into two distinct stages which occur before and after the participants crash. In this model, our interest lies in how fast we can repair a graph structure in the second stage given sufficient time to perform some degree of precomputation in the first stage. In Section 4.3.2 we then use this model to prove new upper bounds for repairing maximal independent sets in maximal matchings in rooted trees, planar graphs, and bipartite graphs. In Section 4.3.3 we provide upper bounds for repairing these structures in the broader class of bounded independence graphs. Our final upper bounds for this model are presented in Sections 4.3.4 and 4.3.5 where we provide algorithms for general graphs for repairing maximal independent sets and maximal matchings, respectively. Finally in Section 4.3.6 we provide lower bounds for these problems, bounding the fastest repair time theoretically possible in this model.

In Section 4.4 we then consider a more pessimistic type of failures: byzantine failures. Here, participants can behave dishonestly in order to actively hinder computation. For this challenging setting, in Section 4.4.1 we first modify the definitions of the classic broadcast-CONGEST model and the MIS problem to account for the presence of byzantine failures. Then in Section 4.4.2 we explore algorithm upper bounds for this byzantine MIS problem, showing that it is indeed possible to compute an MIS given a bounded number of dishonest participants. Finally in Section 4.4.3 we prove lower bounds for this setting, showing the minimum time complexity for this problem given a particular bound on the number of byzantine failures.
1.3 General Preliminaries and Notation

Many of the analytical techniques in this thesis leverage preliminaries which we use only once to prove a core theorem. When this is the case, these existing results are introduced within the analytical scope for which they are relevant. In this section, however, we introduce some more general preliminary notation and results which we will reference throughout this thesis.

1.3.1 Mathematical Preliminaries

While much of our analysis that follows defines more nuanced notation, a few numerical conventions will be assumed throughout the entirety of this work. First, we use \( \mathbb{R} \) to denote the set of all real numbers, \( \mathbb{Z} \) to denote the set of all integers, and \( \mathbb{N} \) to represent the natural numbers (i.e., the set positive integers). Next, let the set \( R(x) = \{1, \ldots, \lceil x \rceil \} \) for \( x \geq 1 \) be the integer range (inclusive) from 1 to \( \lceil x \rceil \). Furthermore, assume that all logarithms are base 2 unless stated otherwise (i.e., \( \log x = \log_2 x \)).

We say that an event occurs "with high probability (w.h.p.) in \( x \)" for some value \( x \), to mean that the event occurs with probability at least \( 1 - \frac{c}{x^d} \) for constants \( c, d \geq 1 \). On the other hand, if an event occurs with probability at most \( \frac{c}{x^d} \) for constants \( c, d \geq 1 \), we say the event occurs "with low probability in \( x \)". We next introduce several useful tools and definitions from the field of probability theory, beginning with the following presentation of Markov’s inequality.

Theorem 1.3.1 (Markov’s Inequality [42]). For a nonnegative random variable \( X \) with expectation \( \mathbb{E}[X] \) and constant \( a \) such that \( a > 0 \), it follows that

\[ \Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}. \]  

(1.1)
Equivalently,

\[ \Pr[X < a] \geq 1 - \frac{E[X]}{a}. \] (1.2)

Markov’s inequality is a useful tool as it holds for any nonnegative random variable, \( X \), whose expectation is known (regardless of the distribution of \( X \)). However, for random variables with particular distributions, much stronger probabilistic bounds exist. For instance, many of the variables we consider in this thesis represent the sums of *negatively associated* random variables. We define this property next.

**Definition 1.3.2** (Negative Association [67]). A set of random variables \( X_1, X_2, \ldots, X_n \) is said to be **negatively associated** if for any subset \( S \subseteq \mathbb{R}^n \) we have that

\[ \Pr\left[ \bigwedge_{i \in S} X_i = 1 \right] \leq \prod_{i \in S} \Pr[X_i = 1]. \] (1.3)

The concept negative association is used to describe random events where some subset of events occurring cannot increase the probability that an event outside of that subset occurs. Note that if the random variables are independent, then the two sides of Line 1.3 are equal, implying negative association. Helpfully, sums of negatively associated random variables can be shown to only deviate from their expected values with small probability. This is formalized in the following presentation of the generalized Chernoff bound.

**Theorem 1.3.3** (Chernoff Bound [37]). Let \( X_1, \ldots, X_n \) be a series of negatively associated random variables such that \( X_i \in [0, 1] \) where \( X = \sum_{i=1}^n X_i \) has expectation \( E[X] \). For any \( \varepsilon > 0 \),

\[ \Pr[X \leq (1 - \varepsilon) \cdot E[X]] \leq \left( \frac{e^{-\varepsilon}}{(1 - \varepsilon)^{1-\varepsilon}} \right)^{E[X]}. \] (1.4)
Since this bound can be somewhat unwieldy, we next derive a simpler version to use in this thesis.

**Theorem 1.3.4** (Chernoff Bound). Let \( X_1, \ldots, X_n \) be a series of negatively associated random variables such that \( X_i \in [0, 1] \) where \( X = \sum_{i=1}^{n} X_i \) has expectation \( \mathbb{E}[X] \geq \mu \). For any \( \varepsilon \in (0, 1) \),

\[
\Pr[X \leq (1 - \varepsilon) \cdot \mu] \leq e^{-\varepsilon^2 \mu / 2}. \tag{1.5}
\]

Equivalently,

\[
\Pr[X > (1 - \varepsilon) \cdot \mu] \geq 1 - e^{-\varepsilon^2 \mu / 2}. \tag{1.6}
\]

*Proof.* Fix \( \varepsilon \in (0, 1) \) and \( \mu \leq \mathbb{E}[X] \) and note that due to the latter, it follows

\[
\Pr[X \leq (1 - \varepsilon) \cdot \mu] \leq \Pr[X \leq (1 - \varepsilon) \cdot \mathbb{E}[X]]. \tag{1.7}
\]

Therefore, since \( \varepsilon > 0 \), by Theorem 1.3.3 we have that

\[
\Pr[X \leq (1 - \varepsilon) \cdot \mu] \leq \Pr[X \leq (1 - \varepsilon) \cdot \mathbb{E}[X]] \tag{1.8}
\]

\[
\leq \left( \frac{e^{-\varepsilon}}{(1 - \varepsilon)^{1-\varepsilon}} \right)^{\mathbb{E}[X]} \tag{1.9}
\]

\[
= \left( \frac{e^{-\varepsilon}}{e^{(1-\varepsilon)\ln(1-\varepsilon)}} \right)^{\mathbb{E}[X]} \tag{1.10}
\]

\[
= \exp(-\varepsilon - (1 - \varepsilon) \ln (1 - \varepsilon))^{\mathbb{E}[X]} \tag{1.11}
\]

\[
= \exp(\mathbb{E}[X] \cdot (-\varepsilon - (1 - \varepsilon) \ln (1 - \varepsilon))). \tag{1.12}
\]
To simplify further, we next use a fact from [99] which states that for all \( x \in (-1, 0] \) it follows,

\[
\ln (1 + x) \geq \frac{x}{2} \cdot \frac{2 + x}{1 + x}.
\] (1.13)

Since \( \varepsilon \in (0, 1) \), clearly \( -\varepsilon \in (-1, 0) \subset (-1, 0] \). Therefore, substituting \( x = -\varepsilon \) in the inequality on Line 1.13 gives us that,

\[
\ln (1 - \varepsilon) \geq -\frac{\varepsilon}{2} \cdot \frac{2 - \varepsilon}{1 - \varepsilon}.
\] (1.14)

Therefore, returning to the inequality on Line 1.12, we have,

\[
\exp (\mathbb{E}[X] \cdot (\varepsilon - (1 - \varepsilon) \ln (1 - \varepsilon))) \leq \exp \left( \mathbb{E}[X] \cdot \left( -\varepsilon - (1 - \varepsilon) \cdot \left( \frac{\varepsilon}{2} \cdot \frac{2 - \varepsilon}{1 - \varepsilon} \right) \right) \right) = \exp (\mathbb{E}[X] \cdot \left( -\varepsilon + (-\frac{\varepsilon}{2} \cdot (2 - \varepsilon)) \right)) = \exp (\mathbb{E}[X] \cdot (-\varepsilon^2 / 2)).
\] (1.15) (1.16) (1.17) (1.18)

Lastly, since \( e^{-\varepsilon^2 / 2} \) is less than 1 (since \( \varepsilon > 0 \)) and \( \mathbb{E}[X] \geq \mu \), it follows that

\[
\exp (\mathbb{E}[X] \cdot (-\varepsilon^2 / 2)) \leq \exp (\mu \cdot (-\varepsilon^2 / 2)) \Rightarrow \Pr[X \leq (1 - \varepsilon) \cdot \mu] \leq e^{-\varepsilon^2 \mu / 2}.
\] (1.19) (1.20)

Furthermore,

\[
\Pr[X > (1 - \varepsilon) \cdot \mu] = 1 - \Pr[X \leq (1 - \varepsilon) \cdot \mu] \geq 1 - e^{-\varepsilon^2 \mu / 2},
\] (1.21) (1.22)
thus completing both statements of the lemma.

On its own, the Chernoff bound is a useful tool for demonstrating that a sum of negatively associated binary random variables is unlikely to deviate greatly from its mean. However, in some cases, it can be further used to describe the sum of arbitrary binary random variables, provided that their probabilities are bounded in some way. We formalize this notion, known as stochastic dominance, in the following lemma.

**Lemma 1.3.5.** Let $X_1, \ldots, X_t$ be a sequence of $t > 0$ arbitrary indicator random variables and let $X = \sum_{i=1}^{t} X_i$ be their sum. Fix some probability $p \in [0, 1]$ such that for all $i \in R(t)$,

$$\Pr[X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \geq p \tag{1.23}$$

for every assignment $x_1, \ldots, x_{i-1} \in \{0, 1\}^{i-1}$ for which $\Pr[X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] > 0$. It follows that,

$$\Pr[X \geq pt/2] > 1 - e^{-pt/8}. \tag{1.24}$$

**Proof.** Fix the sequence $X_1, \ldots, X_t$ and probability $p \in [0, 1]$ as specified by the lemma statement. Now, define the sequence of $t$ independent indicator random variables $\hat{X}_i, \ldots, \hat{X}_t$ where for all $i \in R(t)$, $\hat{X}_i = 1$ with probability $p$ and $\hat{X}_i = 0$ otherwise. Note that this means for all $i \in R(t)$,

$$\Pr[X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \geq p \tag{1.25}$$

$$= \Pr[\hat{X}_i = 1]. \tag{1.26}$$
Let \( \hat{X} = \sum_{i=1}^{t} \hat{X}_i \) and recall that \( X = \sum_{i=1}^{t} X_i \). From Lemma 1.8.7 of [36], we then have that \( X \) stochastically dominates \( \hat{X} \), which, by Definition 1.8.1 of [36], means that for any \( x \in \mathbb{R} \),

\[
\Pr[X \geq x] \geq \Pr[\hat{X} \geq x]. \tag{1.27}
\]

Since \( \hat{X} \), by definition, is a sum of indicator random variables where \( \mathbb{E}[\hat{X}] = pt \), we can apply the Chernoff bound from Theorem 1.3.4 for \( \mu = pt \). For \( \varepsilon = 1/2 \), this gives that,

\[
\Pr[\hat{X} > \mu/2] = \Pr[\hat{X} > pt/2] \geq 1 - e^{-\varepsilon \mu/2} \tag{1.28}
\]

\[
> 1 - e^{-p t/2} \tag{1.29}
\]

\[
= 1 - e^{-(1/2)^2 pt/2} \tag{1.30}
\]

\[
= 1 - e^{-pt/8}. \tag{1.31}
\]

Combining Lines 1.26 and 1.31 for \( x = pt/2 \) then gives that,

\[
\Pr[X \geq pt/2] \geq \Pr[\hat{X} \geq pt/2] \tag{1.32}
\]

\[
\geq \Pr[\hat{X} > pt/2] \tag{1.33}
\]

\[
> 1 - e^{-pt/8}. \tag{1.34}
\]

Therefore, we have that \( \Pr[X \geq pt/2] > 1 - e^{-pt/8} \), as required. \( \square \)

Our analysis will also leverage several inequalities which can be derived as a product of Bernoulli’s inequality (e.g., see [12] for reference) which we introduce next.

**Theorem 1.3.6** (Bernoulli’s Inequality [12]). *For all \( n \geq 1 \) and \( r \geq -1 \), it follows that*

\[
(1 + r)^n \geq 1 + rn.
\]
We next present a convenient manipulation of Theorem 1.3.6.

**Lemma 1.3.7.** For all \( x \in \mathbb{R} \),

\[
e^x \geq 1 + x.
\]  

(1.35)

**Proof.** We begin with the fact from [82] that for any \( x \in \mathbb{R} \),

\[
e^x = \lim_{n \to \infty} \left( 1 + \frac{x}{n} \right)^n.
\]  

(1.36)

Invoking Bernoulli’s inequality from Theorem 1.3.6 by substituting \( r = x/n \) then gives \((1 + x/n)^n \geq 1 + x\). Combining this inequality with the fact from Line 1.36 gives us

\[
e^x = \lim_{n \to \infty} \left( 1 + \frac{x}{n} \right)^n 
\geq 1 + x.
\]  

(1.37)

(1.38)

Therefore \( e^x \geq 1 + x \) for any \( x \in \mathbb{R} \). \( \square \)

Next, we derive a lower bound on the value \( 1 - x \) for \( x \in [0, 1/2] \).

**Lemma 1.3.8.** For all \( x \in (0, 1/2] \),

\[1 - x \geq 4^{-x}. \]  

(1.39)

**Proof.** Fix \( x \in (0, 1/2] \) and let \( n = \frac{1}{2x} \), noting that this implies that \( n \geq 1 \). Furthermore, let \( r = -\frac{1}{2n} \) and note that this means \( r \geq -1/2 > -1 \). Therefore, we can invoke Bernoulli’s
inequality from Theorem 1.3.6 for these values of $n$ and $r$ to derive that,

\[(1 + r)^n \geq 1 + rn \tag{1.40}\]
\[\Rightarrow \left(1 - \frac{1}{2n}\right)^n \geq 1 - \frac{1}{2^n} \tag{1.41}\]
\[= \frac{1}{2} \tag{1.42}\]
\[\Rightarrow 1 - \frac{1}{2^n} \geq 2^{-1/n} \tag{1.43}\]
\[\Rightarrow 1 - x \geq 2^{-2x}. \tag{1.44}\]

Therefore, we have that $1 - x \geq 2^{-2x} = 4^{-x}$ and the lemma is satisfied. \hfill \Box

Finally, we derive a few useful mathematical facts.

**Lemma 1.3.9.** For all $x, y \geq 2$, it follows that $xy \geq x + y$.

**Proof.** This follows simply due to the fact that for all $x, y \geq 2$ we have that

\[xy \geq 2 \max\{x, y\} \tag{1.45}\]
\[\geq x + y. \tag{1.46}\]

Therefore, the lemma is satisfied. \hfill \Box

**Lemma 1.3.10.** For all $x, y \geq 2$, it follows that $\lceil x \rceil \cdot \lceil y \rceil < \lceil 2xy \rceil + 1$.

**Proof.** Fix any $x, y \geq 2$. By the definition of the ceiling function,

\[\lceil x \rceil \cdot \lceil y \rceil < (x + 1)(y + 1) \tag{1.47}\]
\[= xy + x + y + 1. \tag{1.48}\]
Since Lemma 1.3.9 gives that $x + y \leq xy$ for $x, y \geq 2$, we have

\[
xy + x + y + 1 \leq xy + xy + 1 = 2xy + 1 \leq [2xy] + 1
\]

Therefore, $[x] \cdot [y] < [2xy] + 1$. \qed

1.3.2 Graph Preliminaries

As this thesis is concerned with algorithms for networks, we will in all cases use some form of graph notation. Again, this notation is extended at many points for particular proofs, but the core setup will always be relevant. Namely, we consider a network of devices where the network is represented by an undirected graph $G = (V, E)$. Fixing $G$, each device is represented by a node $u \in V$, typically using $n = |V|$ to denote the size of the network.\(^1\) If two devices, $u$ and $v$, are able to communicate in the network, we represent this capability with an undirected edge $\{u, v\} \in E$. For any $u \in V$, let $N(u) = \{v \in V \mid \{u, v\} \in E\}$ denote $u$’s exclusive neighborhood in $G$ (i.e., the nodes with which $u$ is connected to via an edge). This is in contrast to $N^+(u) = \{u\} \cap N(u)$ which we call $u$’s inclusive neighborhood (as it includes $u$ itself). Let $d(u) = |N(u)|$ denote $u$’s degree in $G$, as well as let $\Delta \geq \max_{u \in V} d(u)$ be an upper bound on the maximum degree of any node in $V$. Furthermore, we use $D$ to denote the diameter of $G$, the longest distance between any node pair $u, v \in V$. For a subgraph $G'$ of $G$, let $\Delta(G')$ denote the maximum degree of $G'$ and $D(G')$ denote the diameter of $G'$ (i.e., without considering any nodes or edges not in $G'$).

\(^1\)The only exception is in Chapter 2 where instead $k = |V|$ and $n$ is used to denote an upper bound on $k$. 
For a given subset of nodes $S \subseteq V$, let $\partial S = \{ v \mid v \in V \setminus S, N(v) \cap S \neq \emptyset \}$ denote what we will call the *boundary* of $S$. We now define two useful metrics of graph connectivity: vertex expansion and conductance.

**Definition 1.3.11 (Vertex Expansion).** For a graph $G = (V, E)$ where $n = |V|$, let $\alpha(S) = \frac{|\partial S|}{|S|}$ for a node subset $S \subseteq V$. The vertex expansion, $\alpha$, of $G$ is defined as,

$$
\alpha = \min_{S \subseteq V \mid |S| \in \mathbb{R} \cdot (n/2)} \alpha(S).
$$

(1.52)

Let $B(S)$ represent the bipartite graph with bipartitions $(S, V \setminus S)$ (i.e., with edge set $\{ \{u, v\} \in E \mid u \in S, v \in V \setminus S \}$) and let $\nu(B(S))$ represent the size of the maximum matching over $B(S)$.

**Definition 1.3.12 (Conductance).** For a graph $G = (V, E)$ where $n = |V|$ and node subset $S \subseteq V$, let $\text{vol}(S) = \sum_{u \in S} d(u)$ and let $E_{S, V \setminus S}$ denote the edge set of $B(S)$. The conductance, $\phi$, of $G$ is defined as

$$
\phi = \min_{S \subseteq V \mid 0 \leq \text{vol}(S) \leq \text{vol}(V)/2} \frac{|E_{S, V \setminus S}|}{\text{vol}(S)}.
$$

(1.53)

Our analysis also leverages the following lemma from [57] which makes a connection between the size of the maximum matching over $B(S)$ for any node subset $S \subseteq V$ and the vertex expansion of $G$.

**Lemma 1.3.13 (from [57]).** Let $\gamma = \min_{S \subseteq V \mid |S| \in \mathbb{R} \cdot (n/2)} \nu(B(S))/|S|$. It follows that $\gamma \geq \alpha/4$.

Let $G(V \setminus S)$, for $S \subseteq V$, denote the graph defined when we remove from $G$ the nodes in $S$ and their adjacent edges (i.e., the subgraph of $G$ induced by $V \setminus S$). For a fixed $k \in \mathbb{N}$, we say $G$ has a *$k$-tree* if there exists a spanning tree in $G$ with maximum degree $k$. Finally, let $\Delta_{mdst}(G)$ be the smallest $k$ such that $G$ has a $k$-tree. That is, $\Delta_{mdst}(G)$ describes the maximum

19
degree of the *minimum degree spanning tree* (MDST) in \( G \). Next, let \( c(G \setminus S) \) signify the number of connected components after removing nodes in \( S \) from graph \( G \).

Several of our results build on a graph metric called *toughness*, introduced by Chvátal [21] in the context of studying Hamiltonian paths. It is defined as follows:

**Definition 1.3.14** (Toughness). An undirected graph \( G = (V,E) \) has toughness \( t(G) \) if \( t(G) \) is the largest number \( t \) such that for every \( S \subseteq V : \) if \( c(G \setminus S) > 1 \), then \( |S| \geq t \cdot c(G \setminus S) \).

Intuitively, to have toughness \( t \) means that you need to remove \( t \) nodes for every component you hope to create. Win [103] formalized this by establishing a link between toughness and \( k \)-trees:

**Theorem 1.3.15** (from [103]). For any \( k \geq 3 \), if \( t(G) \geq \frac{1}{k-2} \), then \( G \) has a \( k \)-tree.

Win’s theorem captures the intuition that a small toughness indicates a small number of strategic node removals can generate a large number of components. This in turn implies the existence of a spanning tree containing some high degree nodes (i.e., the nodes whose removal creates many components). We formalize this intuition with the following straightforward corollary of Win’s theorem:

**Theorem 1.3.16.** Fix an undirected graph \( G = (V,E) \) and degree \( k \geq 3 \). If \( \Delta_{mdst}(G) > k \), then there exists a non-empty subset of nodes \( S \subset V \) such that \( c(G \setminus S) > (k - 2) \cdot |S| \).

**Proof.** Since \( \Delta_{mdst}(G) > k \), the contrapositive of Theorem 1.3.15 implies that \( t(G) < 1/(k - 2) \). By the definition of toughness, there exists \( S \subset V \) such that \( |S| = t(G) \cdot c(G \setminus S) \).

For this set, \( c(G \setminus S) = |S|/t(G) > (k - 2)|S| \). \( \square \)

Finally, we present a classic definition and result from the analysis of Alon, Babai, and Itai [1] which we draw on at several points in this thesis. The first is the definition of what are called *good* nodes and edges.
Definition 1.3.17 (from [1]). For a graph $G = (V, E)$ let $N(u)$ and $d(u)$ be the neighborhood (exclusive) and degree functions of node $u \in V$ with respect to $G$. A node $u$ is bad w.r.t. $G$ if $|\{v \in N(u) \mid d(v) > d(u)\}| \geq (2/3) \cdot d(u)$. Otherwise, $u$ is good w.r.t. $G$. An edge $\{u, v\} \in E$ bad w.r.t. $G$ if both $u$ and $v$ are bad w.r.t. $G$. Otherwise $\{u, v\}$ is good w.r.t. $G$.

Put another way, a node $u$ is good with respect to a particular graph $G$ if at least $1/3$ of its neighbors have a degree that is at most $u$’s degree. The authors of [1] then use this definition to show the following useful result regarding good edges which we will use multiple times throughout this thesis.

Theorem 1.3.18 (from [1]). For a graph $G = (V, E)$, at least half of the edges $E$ are good with respect to $G$. 

21
In this chapter, we extend existing distributed algorithm theory relevant to the design of contention resolution protocols, which are critical to the functionality of the data link layer of wireless networks.

The data link layer is one of the lowest layers of the OSI model, second only to that which describes the physical media used to send raw bits. It is comprised of two sublayers, the logical link control (LLC) and media access control (MAC) sublayers, which together facilitate communication between directly connected, adjacent hosts. The LLC sublayer allows for hosts to multithread different protocols over a single communication link, while the MAC sublayer enables multiple hosts to share the link. Each communication link is subject to the physical constraints of the underlying wireless technology, and as such can only provide a finite capacity. Therefore, it is the responsibility of the data link to negotiate the use of these links by allocating this capacity between different processes and different hosts. Any mismanagement of a shared link can lead not only to wasted transmission energy, but also partial or total loss of information. Therefore, reducing contention over the link is of critical importance to the protocols of this layer.

In distributed algorithms research, the theoretical analogue to this challenge of negotiating the use of a shared communication link is called the contention resolution problem. In general, this problem consists of a network of $k$ nodes which are assumed to be connected via a shared channel or link with a finite capacity. To solve the problem, one of these nodes must use the shared link to transmit a message to all other participants. However, due to
the link’s finite capacity, the contention resolution problem dictates that if too many nodes transmit simultaneously, then some of the transmissions may be lost. Therefore, the network participants must compete over access to the limited shared communication media in order to solve the problem. Most commonly, these models represent time synchronously by structuring an algorithm execution into synchronized rounds with integer labels. Each round, a participant can choose to either transmit on the shared channel, or stay silent. In one class of data link layer models, called radio networks, if two or more participants transmit within the same round then all of the transmissions in that round are lost. This is called a collision, an event which captures the main restriction when considering data link layer protocols. The contention resolution problem is considered solved in the first round in which exactly one participant broadcasts alone.

As we discuss in Section 2.1, the contention resolution problem is a classic, well-studied problem. Therefore, there is a large body of existing work which considers this problem in many models which capture the finite capacity constraint essential to studying the data link layer. These abstractions primarily differ based on how they treat the collision events mentioned already. For example, in some models participants may be made aware that a collision has occurred, while in others the participants cannot distinguish a collision from silence on the channel. However, despite their differences, all existing models for this problem share one key restricting assumption: which is that they assume very little prior knowledge of the network size. Typically this is limited to knowing an upper bound, $n \geq k$, on the possible number of participants, $k$. That is, no existing model for this problem assumes that participants possess some sort of estimate for the actual number of participants in the network, $k$, leaving a gap in the current literature.

Therefore, in this thesis we study the effect of partial knowledge (often called advice) on the complexity of these solutions. Specifically, we ask the question of whether or not the many well-known upper and lower bounds for this problem can be strengthened if the
participants are given some advance knowledge of the network size. Furthermore, how can we quantify the dependence between the quality of the advice and its impact on the time complexity of the contention resolution problem? The notion of algorithms with advice has gained traction in the past few years, with recent work considering the scenario where a machine learning model is deployed to gather probabilistic estimates of network properties. One such example is the work of Mitzenmacher and Vassilvitskii [83] which studies the classic "ski rental problem". In this study, the authors show that given some stochastically structured knowledge about the environment, very efficient strategies can be developed based on the accuracy of the advice. More generally, the impact of advice has been studied for a variety of different distributed problems [48, 41, 40, 14].

In this chapter, we extend this line of work regarding algorithms with advice to the contention resolution problem. In Section 2.1 we start by discussing the rich literature behind this fundamental problem, allowing us in Section 2.2 to summarize the new results we achieve in detail. The formal presentation of our results begins in Section 2.3 where we develop a model which allows participants prior access to some imperfect knowledge of the number of participants. We represent this knowledge as a probability distribution over the possible network sizes, which may be different than the true distribution. The challenge is then to use our model to characterize the impact of this knowledge. First, in Section 2.4 we provide lower bounds for the contention resolution problem in this model which depend on the Shannon entropy of the true distribution and the Kullback-Leibler divergence between the true distribution and the one provided to the participants. In Section 2.5 we then provide upper bounds for the same setting, expressed with respect to the same statistical quantities. Both sets of results include bounds for the version of the contention resolution problem with collision detection and the version without. Note that all of the results in this chapter first appeared in [62].
Before proceeding any further with the results we achieve, however, there are many existing results regarding the contention resolution problem in more standard models which we must first consider.

2.1 Existing Work on Contention Resolution

In this section, we formally explore the existing work surrounding the contention resolution problem. As mentioned, the contention resolution problem is well-studied in a number of models which capture the finite capacity constraint which defines data link layer protocols. Historically, abstractions for this setting consisted of several devices connected via a single channel (representing a shared communication link) [18, 102]. Once again, typically time is assumed to proceed in synchronized rounds with integer labels. In each round, each device chooses to transmit on the shared channel or stay silent.

While this general structure applies to many of the models for this layer, a family of models known as radio networks differ based on how they handle collisions. The term collision describes the event that more than a single node transmits on the channel in a given round, which in all cases leads to some degree of information loss (e.g., no transmissions that round are received). Where these models differ primarily, is whether or not the devices can detect if a collision has occurred. This ability in the literature is called collision detection and can have a dramatic impact on the complexity of many problems in this setting. The significance of collisions in these models captures the importance of the data link layer’s core responsibility to properly allocate the capacity of a shared communication link between multiple processes and hosts.

The wake-up problem in single-channel radio networks has been extensively studied since the model was introduced by the work of Chlamtac and Kutten [18] almost 40 years
ago. Without collision detection, the classic decay strategy proposed by Bar-Yehuda, Goldreich, and Itai [5] achieves a time complexity of $O(\log n)$ expected time. For uniform algorithms in this setting (where all nodes broadcast with same probability each round), the lower bound of $\Omega\left(\frac{\log n}{\log \log n}\right)$ expected time was proven by Jurdziński and Stachowiak [71]. This bound was later improved to $\Omega(\log n)$ expected time by Farach-Colton, Fernandes, and Mosteiro [43] and generalized by Newport [87] to apply to non-uniform algorithms. When collision detection is available, the seminal work of Willard [102] demonstrates that $\Theta(\log \log n)$ expected time is both sufficient and necessary.

On an entirely separate research front, some models use communication mechanisms based on more sophisticated representations of the physical properties of radio signals. For example, in the SINR-style communication model, the work of Fineman et al. [45] demonstrates a contention resolution algorithm which takes $O(\log n + \log R)$ with high probability (w.h.p.) in $n$, where $R$ is the ratio between the longest and shortest communication distances in the network. Similar to the models we have discussed so far, these models represent time synchronously and each round nodes can transmit or listen. However, these abstractions use formulas such as the signal to interference and noise ratio (SINR) equation to dictate whether or not a device receives a particular transmission. Such formulas weigh the power used to generate the transmission and the proximity of the sender against the interference caused by the transmissions of other nearby devices.

More generally, contention resolution has been studied in many model settings beyond that of radio network models. The work of Censor-Hillel et al. [17], uses novel random coding techniques to solve a variety of symmetry breaking problems in additive networks, each collision is represented as the sum of the conflicting transmissions. In multi-channel networks with collision detection, the wake-up algorithm of Fineman, Newport, and Wang [46] takes $O(\log_C n + \log \log n \log \log \log n)$ w.h.p. in $n$, where $C$ is the number of channels. This bound is nearly tight (to within a factor of $\log \log \log n$) of the corresponding
\( \Omega(\log n + \log \log n) \) lower bound proved by Newport [87]. There has also been much research into what are called beeping networks, which restrict transmissions to unary signals called "beeps" [3, 39, 38].

2.2 Summary of New Results

There is little doubt that the contention resolution problem is well-studied under many different assumptions, but one glaring gap in the existing body of research is the emphasis on worst-case performance in the presence of no knowledge about the network topology. Therefore as we have mentioned, in this thesis we study the impact of partial knowledge (sometimes called advice) on the time complexity of these solutions.

To represent the concept of advice, in Section 2.3, we augment the standard contention resolution problem to provide each algorithm as input the definition of a random variable \( X \) defined over the possible participant set sizes in the range \( R(n) \) (recall once again that \( n \) is a known upper bound on the actual number of participants \( k \)). The goal is to use the distribution defining \( X \) to speed up contention resolution if possible. (For concision, we sometimes say the algorithm is provided a random variable over network sizes as input. In these instances, we mean that the full distribution defining the variable is provided.)

However, as we alluded to in Section 1.2, we do not need the exact network size, \( k \), to solve contention resolution fast. An estimate within a constant factor of \( k \) is sufficient. Given a network size random variable \( X \) describing \( k \), let \( c(X) \) be the condensed version of \( X \) that aggregates the probability mass over \([\log n]\) value ranges of geometrically increasing size. Formally, for each \( i \in R(\log n) \), define

\[
\Pr[c(X) = i] = \sum_{2^{i-1} < j \leq 2^i} \Pr[X = j]
\] (2.1)
Note this definition assumes \( k > 1 \).\(^1\) Again, intuitively, knowing a value \( i \in R(\log n) \) such that \( k = \Theta(2^i) \), is sufficient. We therefore express the bounds that follow for a given \( X \) with respect to \( c(X) \).

### 2.2.1 Lower Bounds

We begin in Section 2.4 by proving two lower bounds for this new model. To achieve the strongest possible result, we assume that the variable \( X \) is accurate, in the sense that the network will actually determine the number of participants according to \( X \). We also assume *uniform* algorithms in which all players use the same transmission probability in each round from a fixed schedule.\(^2\)

Intuitively, if \( c(X) \) places all of its probability mass on a single network size range, then we are in the perfect prediction setting and can solve the problem in \( O(1) \) rounds. On the other extreme, if \( c(X) \) describes a uniform distribution, we are likely unable to do much better than the worst-case bounds. To describe these cases we need a property of these distributions that succinctly captures their predictive power. The natural candidate here is entropy—and this is indeed what we end up deploying in our results.

We begin in Section 2.4.1 by considering contention resolution with no collision detection. We prove that for a given network size random variable \( X \), contention resolution requires \( \Omega(\frac{2^{H(c(X))}}{\log \log n}) \) rounds in expectation, where \( H \) denotes Shannon entropy. For maximum values of \( H(c(X)) \), this reduces to \( \Omega\left(\frac{\log n}{\log \log n}\right) \) rounds, exactly matching the original uniform algorithm lower bound from [71]—a result that is interesting in its own right, as it shows yet another approach to proving limits to this fundamental problem.

---

\(^1\)Note that although our definition excludes the possibility that \( k = 1 \), this edge case is easily handled. For example, we could modify any algorithm to have all nodes broadcast with probability 1 in the first round. This would add at most one round to the time complexity of any algorithm and solve the problem in the first round when \( k = 1 \).

\(^2\)It is important to point out that until recent work [87], most existing lower bounds for contention resolution assumed uniform algorithms.
The core idea driving this bound is a connection between contention resolution and coding. Using a reduction argument involving an intermediate game we call range finding, we formalize the intuition that solving contention resolution requires an algorithm to try a transmission probability relatively close to the optimal probability. We can therefore use a contention resolution algorithm designed for $X$ to help construct a code for a given symbol source $c(X)$. Let $s \in R(\log n)$ be the symbol to be transmitted. We can then consider the execution of our algorithm in a network of size $2^s$, and identify the first round $r$ such that the algorithm attempts a transmission probability near $2^{-s}$, and send $r$ to the receiver. Because the receiver knows the same algorithm, it can use the probability scheduled for round $r$ to determine $s$. If the algorithm terminates fast in expectation with respect to $X$, then the expected code length of our code for $c(X)$ is small. Shannon’s Source Coding Theorem (see Section 2.3), however, tells us that the average code length for $c(X)$ is lower bounded by $H(c(X))$. Because, roughly speaking, the code length is computed as the logarithm of the round complexity, we get a bound of roughly $2^{c(H(X))}$ rounds.

In Section 2.4.2, we leverage a similar connection between contention resolution and information theory to prove that, given collision detection, uniform algorithms require $\Omega(H(c(X)))$ rounds in expectation with respect to a network size variable $X$. For the maximum possible entropy of $c(X)$, this matches the original $\Omega(\log \log n)$ lower bound proven by Willard using non-information theoretic techniques [102].

A uniform algorithm in the collision detection setting can be understood as a function that maps a binary collision history$^3$ to the uniform probability that all players try in the current round given the current collision history. We show how to construct a code for $c(X)$ for such an algorithm by directly sending the shortest collision history that corresponds to a

$^3$In the collision detection model, in every round, either every player detects a collision or no player detects a collision. We can therefore encode this history for the $r$ rounds of an execution as a binary string $b_1b_2\ldots b_r$, where $b_i = 0$ if there was no collision in round $i$, and $b_i = 1$ if there was a collision in $i$. 

29
<table>
<thead>
<tr>
<th>No Collision Detection</th>
<th>Lower Bounds (*)</th>
<th>Upper Bounds (*)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\Omega\left(\frac{2^{H(c(X))}}{\log \log n}\right))</td>
<td>(O\left(2^{2H(c(X))}\right))</td>
</tr>
<tr>
<td>Collision Detection</td>
<td>(\frac{1}{16}H(c(X)) - O(\log \log \log \log n))</td>
<td>(O(H^2(c(X))))</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of contention resolution results achieved in this thesis. Here we summarize our results for contention resolution with network size predictions. \(H(c(X))\) is the entropy of the condensed probability distribution over the possible network sizes.

(*): Lower bounds are expressed as the expected number of rounds while upper bounds are expressed with respect the number of rounds with at least constant probability.

Probability well-matched to the current symbol. (As in the no collision detection case, there is actually an intermediate step involved here where our contention resolution algorithm reduces to a simpler game called range finding, and it is, in fact, this range finding solution from which we generate our code.) In this case, the round complexity is the same as the code length. It follows that \(H(c(X))\) must bound the expected value of the former.

2.2.2 Upper Bounds

We next turn our attention to contention resolution upper bounds that take a network size random variable as input and attempt to come as close as possible to matching the relevant lower bounds. We start in Section 2.5.1 with an algorithm for the no collision detection setting. Let \(X\) be the actual distribution from which the network size will be drawn. Let \(Y\) be the distribution the algorithm is given as input (which may be different than \(X\)). We analyze a natural strategy: sort the values in \(R(\log n)\) in decreasing order of likelihood given \(c(Y)\); visit these values in turn, for each such \(i\) transmitting with probability \(2^{-i}\).
We prove that with constant probability, this strategy succeeds in $O(2^T)$ rounds, where:

$$T = 2H(c(X)) + 2D_{KL}(c(X) \parallel c(Y)),$$

(2.2)

and $D_{KL}$ denotes the Kullback–Leibler divergence between the two distributions (see Section 2.3). When $Y = X$, this simplifies to $O(2^{2H(c(X))})$ which matches the same general form of our lower bound, but includes an extra constant factor in the exponent. We conjecture that something like this extra factor may be fundamental in the analysis of this natural strategy. We support this conjecture by noting a straightforward application of a cryptography result due to Pliaim [92] which implies that this strategy requires $\omega(2^{2H(c(X))})$ rounds.  

An important characteristic of the Kullback–Leibler divergence is that it is if each probability in $Y$ is off by at most a bounded constant fraction from the real probability in $X$, $D_{KL}(c(X) \parallel c(Y)) = O(1)$. This establishes that our algorithm does not require a precise prediction to be useful, and its efficiency increases smoothly along with prediction quality.

In Section 2.5.2, we consider algorithms for the collision detection setting. Given network size variable $Y$, the algorithm first constructs an optimal code $f$ with respect to source $c(Y)$. It then considers all codes from shortest to longest in length. For each length $\ell$, it considers all symbols mapped to codes of this length. Ordering these symbols from smallest to largest, it deploys the collision detector-driven binary search strategy introduced by Willard [102] to explore if any of these symbols correspond to the correct network size.

---

4To be more precise, the result in [92] shows that for every constant $\alpha \geq 1$, there is a random variable $X$ such that this strategy requires more than $\alpha 2^{H(c(X))}$ rounds to succeed with constant probability. Therefore, there can be no constant (e.g., $\alpha_0$) for which an algorithm can be shown to take at most $\alpha_0 2^{H(c(X))}$ for every random variable $X$ (since we can pick any constant $\alpha_1 > \alpha_0$ and find the random variable $Y$ for which more than $\alpha_1 2^{H(c(Y))}$ rounds are required). Formalizing this result would require quantification of the reality that an exact guess is not needed to solve the problem. This crude application, however, serves its purpose of bolstering our conjecture that $O(2^{H(c(X))})$ rounds are insufficient.
We show that this algorithm solves the problem in \( O((H(c(X)) + D_{KL}(c(X) \parallel c(Y)))^2) \) rounds with constant probability, which reduces to \( O(H^2(c(X))) \) rounds when the probabilities in \( Y \) are all within a bounded constant factor of the real probabilities in \( X \). As before, our upper bound shares the same general form as the lower bound (in this case, a result expressed as a polynomial of \( H(c(X)) \)), but is not exactly tight. Closing these final gaps between the upper and lower bounds for small divergence is not obvious, but likely tractable; and therefore left as intriguing future work. We also note that for clarity, we focused only on one-shot attempts to resolve contention that succeed with constant probability. Achieving good bounds on expected time also represents important future work.

2.3 Model and Problem Definition

We now introduce some additional notation. Fix a network size \( n \). Let \( X \) be the discrete random variable that determines the number of participants in each instance of contention resolution. In more detail, \( X \) takes its values from \( R(n) \), that occur with probabilities \( p_1, p_2, \ldots, p_n \), respectively. In our setting, the size \( k \) of the participant set is determined by \( X \), leaving the adversary only to determine which \( k \) nodes participate. Notice, however, when considering uniform algorithms the identity of the participants is not consequential as transmission behavior is determined exclusively by the algorithm and the collision history.

In our algorithms, we sometimes talk about this random variable \( X \), or perhaps an estimate \( Y \) of this random variable, being provided as input to the algorithm. This is shorthand for the more accurate statement that the underlying distribution defining the variable is provided as input. That is, the algorithm is given for each network size, the probability that this size is drawn.

In our analysis, we will find it useful to reference a condensed version of \( X \), we call \( c(X) \), which aggregates the probability mass spread over \( n \) possible network sizes into \( \lfloor \log n \rfloor \).
geometric size ranges. We associate each range \( i \in R(\log n) \) with the values in the interval \((2^{i-1}, 2^i]\). That is, \( i = 1 \) is associated with just the value 2, \( i = 2 \) is associated with the range 3 to 4, \( i = 3 \) is associated with 5 to 8, and so on. Implicit in this definition is the assumption that the network size is always of size at least 2 as there is no contention to resolve in a network of size less than 2.\(^5\) The random variable \( c(X) \) takes its values from \( R(\log n) \). For each \( i \in R(\log n) \), let \( q_i = \Pr[c(X) = i] \), defined as follows:

\[
q_i = \sum_{2^{i-1} < j \leq 2^i} p_j
\]

Next, our analysis uses the following version of Jensen’s inequality for concave functions:

**Theorem 2.3.1** (Jensen’s Inequality [69]). If \( p_1, \ldots, p_n \) are real numbers where \( p_i \geq 0 \) for all \( i \in R(n) \) and \( \sum_{i \in R(n)} p_i = 1 \), and \( f \) is a real continuous function that is concave, then

\[
f \left( \sum_{i=1}^{n} p_i x_i \right) \geq \sum_{i=1}^{n} p_i f(x_i). \tag{2.4}
\]

It also builds on the lower bound result from Shannon’s famed Source Coding Theorem, which concerns the efficiency of codes on noiseless channels:

**Theorem 2.3.2** (Shannon’s Source Coding Theorem [97]). Let \( X \) be a random variable taking values in some finite alphabet \( \Sigma \). Let \( f \) be a uniquely decodable code from this alphabet to \( \{0, 1\} \). Let \( S \) be the random variable that describes the length of codeword

\(^5\)This assumption can hold without loss of generality as all algorithms can eliminate the \( k \leq c \) possibility for any constant \( c \in \mathbb{N} \) in an additional \( c = O(1) \) early rounds in which round \( i \in R(c) \) all players transmit with probability \( 1/i \).
\[ H(X) \leq \mathbb{E}[S]. \] (2.5)

Our upper bound analysis considers the case in which the network size distribution \( Y \) provided as input to our algorithms does not exactly match the actual distribution \( X \) from which the network size will be drawn. Drawing from standard statistics, we can use the Kullback-Leibler divergence between the two distributions, denoted \( D_{KL}(X \| Y) \), to quantify their differences. We then leverage the following well-known information theory result which bounds the decrease in coding performance, with respect to \( D_{KL}(X \| Y) \), when you construct an optimal code for \( Y \) that you then combine with symbol source \( X \) (for more on Kullback-Leibler divergence and the coding bound, see the excellent review in [26]):

**Theorem 2.3.3.** Let \( X \) and \( Y \) be random variables taking values in some finite alphabet \( \Sigma \). Let \( f \) be an optimal, uniquely decodable code from \( Y \) to \( \{0, 1\} \). Let \( S \) be the random variable for the length of codeword \( f(X) \). Let \( H \) be the entropy function and \( D_{KL} \) be the Kullback-Leibler divergence of two distributions. It follows:

\[ H(X) + D_{KL}(X \| Y) \leq \mathbb{E}[S] \leq H(X) + D_{KL}(X \| Y) + 1. \] (2.6)

Further, we note that \( D_{KL}(X \| X) = 0 \) for any random variable \( X \).

Lastly, note once more that our focus in this thesis is on uniform contention resolution algorithms, in which participants rely on predetermined probabilities. With no collision detection, a uniform algorithm can be interpreted as a sequence of probabilities, \( \delta_1, \delta_2, \delta_3, \ldots \), such that in round \( i \), all participants broadcast with probability \( \delta_i \). With collision detection, a uniform algorithm can be interpreted as a function \( f \) from collision histories to broadcast probabilities. For a given round \( r \), let the binary string \( B = b_1, b_2, \ldots, b_{r-1} \)
describe the collision history through the first $r-1$ rounds of the given execution (i.e., $b_i = 0$ indicates the channel was silent in round $i$, while $b_i = 1$ indicates there was a collision). In round $r$, all players transmit with the same uniform probability $f(b_r)$. Uniformity is a common assumption in the study of contention resolution. Many previous lower bounds assume this property (e.g., [102, 71, 26]).

2.4 Lower Bounds for the Contention Resolution Problem with Predictions

We now assess the impact of network size predictions on the contention resolution problem from a lower bound perspective. Namely, we seek to quantify how fast is theoretically possible to solve the contention resolution problem given a probability distribution over the possible network sizes. First, we introduce a useful preliminary result used by both bounds in this section, known as Rice codes [104].

Both of our lower bounds involve constructing a coding scheme which in part requires sending an integer value as a variable-length, uniquely decodable bit string. To accomplish this, we leverage a particular instantiation of the Rice coding mechanism as a subroutine. Namely, fix any positive integers $z, k \in \mathbb{N}$ and note that $z$ can be expressed as $z = 2^kq + r + 1$ where $q = \lfloor (z - 1)/2^k \rfloor$ and $r = z - 2^kq - 1$. The Rice code $R_k(z)$ of $z$ leverages this fact by encoding $z$ into two parts: a variable-length, unary representation of $q$ (q zeroes followed by a one), and a fixed-length, binary representation of $r$ using $k$ bits (since $r < 2^k$). Therefore if $|R_k(z)|$ is the length of the Rice code of $z$, note that $|R_k(z)| = q + k + 1$. Rice codes are a special case of Golomb codes (see [104] for an overview) and are easily shown to be uniquely decodable. For a discrete random variable $Z$ which takes on a positive integer value, let $R_k(Z)$ represent the random variable denoting the Rice code $R_k(z)$ of a random value $z$ selected according to $Z$. Let $|R_k(Z)|$ denote the random variable corresponding to the length of this random code. For convenience, let $R(Z) = R_{\lceil \log E[Z] \rceil}(Z)$. 

35
Lemma 2.4.1. For discrete random variable $Z$ it follows that $E[|\mathcal{R}(Z)|] < \log E[Z] + 3$.

Proof. Let $k = \lceil \log E[Z] \rceil$. For any $z \in \mathbb{N}$, $|\mathcal{R}_k(z)| = q + k + 1$ where $q = \lfloor (z + 1)/2^k \rfloor$. Therefore,

$$|\mathcal{R}_k(z)| = \left\lfloor \frac{z - 1}{2^k} \right\rfloor + k + 1 = \left\lfloor \frac{z - 1}{2 \log E[Z]} \right\rfloor + \lceil \log E[Z] \rceil + 1. \quad (2.7)$$

Next since $\log E[Z] \leq \lceil \log E[Z] \rceil < \log E[Z] + 1$ and $\lfloor x \rfloor \leq x$ for any $x \in \mathbb{R}$,

$$\left\lfloor \frac{z - 1}{2 \log E[Z]} \right\rfloor + \lceil \log E[Z] \rceil + 1 < \left\lfloor \frac{z - 1}{2 \log E[Z]} \right\rfloor + \log E[Z] + 2 \quad (2.9)$$

$$\leq \frac{z - 1}{2 \log E[Z]} + \log E[Z] + 2 \quad (2.10)$$

$$= \frac{z - 1}{E[Z]} + \log E[Z] + 2. \quad (2.11)$$

Therefore,

$$E[|\mathcal{R}(Z)|] < E\left[\frac{z - 1}{E[Z]} + \log E[Z] + 2\right] \quad (2.12)$$

$$= \frac{E[Z] - 1}{E[Z]} + \log E[Z] + 2 \quad (2.13)$$

$$< 1 + \log E[Z] + 2 \quad (2.14)$$

$$= \log E[Z] + 3, \quad (2.15)$$

satisfying the lemma. \qed

2.4.1 Lower Bound for No Collision Detection

In this section, we prove a lower bound for the time complexity for the no collision detection setting. Our goal is to prove the following lower bound that connects contention res-
olution with a known network size distribution to the entropy of the condensed version of that distribution:

**Theorem 2.4.2.** Fix a uniform algorithm A for a network with size upper bounded by \( n \geq 16 \). Let \( t_X(n) \) be the expected round complexity for A to solve contention resolution on a channel with no collision detection and the number of participants determined by random variable \( X \). It follows:

\[
t_X(n) = \Omega\left( \frac{2^{H(c(X))}}{\log \log n} \right).
\]  

Our proof strategy deploys two steps. We begin by defining an abstract combinatorial-style problem called range finding (RF), which we can more directly and clearly connect to entropy. We then show how to transform a contention resolution solution into range finding solution with a related time complexity.

**Sequence Range Finding**

The range finding problem is parameterized with a network size \( n \) and range expressed as a function \( f(n) \) of this size. A range finding strategy can take the form of a sequence of values from \( R(\log n) \), or a binary tree with its nodes labelled with values from \( R(\log n) \). Here we define the version defined with respect to a sequence, as this is the version needed for our proof of the above theorem. The binary tree variation will be used when we later consider contention resolution with collision detection.

We say a sequence \( S = v_1, v_2, \ldots \) *solves* the \((n, f(n))\)-range finding problem in \( t \) steps for a given target \( v \in R(\log n) \), if \( S[t] \) is the first position in \( S \) such that \( |S[t] - v| \leq f(n) \), where \( S[t] = v_t \) is the \( t \)th element of \( S \). To handle probabilistic selections of targets from
a known distribution, fix some random variable $Y$ that takes values from $R(\log n)$. For each $i \in R(\log n)$, let $p'_i = \Pr[Y = i]$. We say $S$ solves $(n, f(n))$-range finding in expected time $t$ with respect to $Y$, if $t$ is the expected step at which $A$ solves the problem when the target value, $v \in R(\log n)$, is determined by $Y$.

**Bounding Sequence Range Finding Using Entropy**

Assume $S$ is a sequence that solves $(n, f(n))$-range finding in expected time $T$ with respect to some distribution $Y$ over $R(\log n)$. We can use $S$ to design a code for source $Y$ with an efficiency determined by $f(n)$. We leverage this connection to prove the following about the connection between range finding and entropy for a range $f(n) = O(\log \log n)$ that will prove useful for our subsequent attempts to connect contention resolution to range finding:

**Lemma 2.4.3.** Let $S$ be a sequence that solves $(n, \alpha \log \log n)$-range finding for some constant $\alpha \geq 1$ and network size $n > 1$. Assume that the range is determined by random variable $Y$. Let $Z$ be the random variable describing the complexity of $S$. It follows:

$$E[Z] = \Omega\left(\frac{2^{H(Y)}}{\alpha \log \log n}\right),$$

(2.17)

where $H$ is the entropy function.

**Proof.** Fix some $n, \alpha, S$ and $Y$ as specified by the lemma statement. We can use $S$ to design a code for transmitting symbols from $R(\log n)$ over a noiseless channel as follows:

- Initialize the sender and receiver with sequence $S$.

- To communicate a value $x \in R(\log n)$, the sender transmits $(R_{\lceil E[Z] \rceil}(r), |d|, b)$ to the receiver, where:
  - $r$ is the first round such that $S[r]$ is within $\alpha \log \log n$ of $x$ (i.e., the first round to solve $(n, \alpha \log \log n)$-range finding for $x$),

38
- \( d = x - v_r \) is the distance of \( S[r] \) from \( x \), and
- \( b = \begin{cases} 
0 & \text{if } d < 0 \\
1 & \text{if } d \geq 0 
\end{cases} \) indicates the sign of \( d \).

- The receiver can then locally calculate \( x = S[r] + d \).

Recall that \( R_k(r) \) (detailed in the beginning of Section 2.4) refers to the Rice encoding of \( r \) parameterized by some \( k \in \mathbb{N} \). Let us call the above scheme target-distance coding. We now relate the expected code length with this scheme with the expected complexity of range finding with \( A \). Consider the performance of the target-distance coding scheme based on these values. For a given \( x \in R(\log n) \), that is solved by \( S \) by round \( r \), the code length of this scheme is upper bounded by \( |R_{\log E[Z]}(r)| + \lceil \log (\alpha \log \log n + 1) \rceil + 1 \). Since \( r \) is chosen according to \( Z \), we therefore have that the expected code length is upper bounded as:

\[
E[|R_{\log E[Z]}(Z)|] + \lceil \log (\alpha \log \log n + 1) \rceil + 1 = E[|R(Z)|] + \lceil \log (\alpha \log \log n + 1) \rceil + 1
\]

\[\leq E[|R(Z)|] + \lceil \log (\alpha \log \log n) \rceil + 2 \quad (2.18)\]

Again, recall that we define \( R(Z) = R_{\log E[Z]}(Z) \) for convenience. By applying the result from Lemma 2.4.1 regarding our Rice coding scheme, we can then further refine this bound as follows:

\[
E[|R(Z)|] + \lceil \log (\alpha \log \log n) \rceil + 2 \leq \log E[Z] + 3 + \lceil \log (\alpha \log \log n) \rceil + 2
\]

\[= \log E[Z] + \lceil \log (\alpha \log \log n) \rceil + 5. \quad (2.21)\]

Shannon’s Source Coding Theorem (Theorem 2.3.2) tells us that the average code length of this scheme is lower bounded by the entropy of \( Y \). It follows that \( H(Y) \leq \log E[Z] + \)
\[ \lceil \log (\alpha \log \log n) \rceil + 5, \text{ which implies:} \]

\[ 2^{H(Y)} \leq 2^{\log \mathbb{E}[Z] + \lceil \log (\alpha \log \log n) \rceil + 5} \]  
\[ \Rightarrow 2^{H(Y)} \leq 2^{\log \mathbb{E}[Z] 2^{\lceil \log (\alpha \log \log n) \rceil + 5}} \]  
\[ \Rightarrow 2^{H(Y)} \leq \mathbb{E}[Z] \cdot 64\alpha \log \log n \]  
\[ \Rightarrow \mathbb{E}[Z] \geq \frac{2^{H(Y)}}{64\alpha \log \log n} \]  
\[ \Rightarrow \mathbb{E}[Z] = \Omega \left( \frac{2^{H(Y)}}{\alpha \log \log n} \right), \]

as claimed by the lemma. \(\square\)

**Solving Range Finding with Contention Resolution**

Here we transform a solution to contention resolution to a sequence that solves range finding with a similar expected complexity. Contention resolution is a more general problem than range finding, so care is needed to tame its possible unexpected behaviors.

We begin by defining an algorithmic process for transforming a uniform algorithm \(A\) into a range finding sequence \(S_A\). We then analyze the properties of \(S_A\).

We now analyze the quality of the range finding solution produced by our RF-construction algorithm. We begin a useful helper lemma that formalizes the intuitive notion that a contention resolution algorithm is unlikely to succeed if its probability is too far from the optimal value for the participant count.

**Lemma 2.4.4.** Assume in a given round of a uniform contention resolution algorithm that the \(1 < k \leq n\) participants each transmit with probability \(p\) such that \(p < \frac{1}{k\beta \log n}\) or \(p > \frac{\beta \log n}{k}\), for some constant \(\beta \geq 6\). If \(n \geq 3\), it follows that the probability exactly one participant transmits is strictly less than \(\frac{1}{4 \log n}\).
Algorithm 1 Range Finding Construction

Input Uniform contention resolution algorithm \( A = \delta_1, \delta_2, \ldots \)

Output Range finding sequence \( S_A \)

1: \( S_A \leftarrow \emptyset \)
2: \( i \leftarrow 1 \)
3: \( j \leftarrow 0 \)
4: while true do
5: Append \( \lceil \log (1/A[i]) \rceil \) to end of \( S_A \)
6: Append \( 2^j \) to end of \( S_A \)
7: \( j \leftarrow j + 1 \)
8: if \( j > \lceil \log n \rceil \) then
9: \( j \leftarrow 0 \)
10: \( i \leftarrow i + 1 \)
return \( S_A \)

Proof. Fix \( \beta \geq 6 \). If \( k \) nodes transmit with probability \( p \), the number of nodes which transmit is governed by the binomial distribution \( B(k, p) \) and so the probability that a single node transmits is \( \Pr[B(k, p) = 1] = kp(1 - p)^{k-1} \). Therefore, if \( p < \frac{1}{k\beta \log n} \) then

\[
\Pr[B(k, p) = 1] < \frac{k}{k\beta \log n} \cdot (1 - p)^{k-1} < \frac{1}{\beta \log n},
\]

satisfying our lemma for any \( \beta \geq 6 \). Secondly, if it is instead the case that \( p > \beta \log n / k \), it follows that:

\[
\Pr[B(k, p) = 1] < kp \left( 1 - \frac{\beta \log n}{k} \right)^{k-1} = kp \left( 1 - \frac{\beta \log n}{k} \right)^{\left(\frac{k}{k(k-1)}\right)}. \]
Lemma 1.3.7 then gives that for \( x = -\beta \log n / k \) that

\[
k p \left( 1 - \frac{\beta \log n}{k} \right)^{(k/k)(k-1)} \leq \frac{k p}{e^{(\beta \log n) / (k-1)/k}}
\]

(2.31)

\[
\leq \frac{k p}{e^{(\beta \log n) / 2}}
\]

(2.32)

\[
\leq \frac{k p}{2^{(\beta \log n) / 2}}
\]

(2.33)

\[
= \frac{k p}{n^{\beta/2}}
\]

(2.34)

\[
\leq \frac{k}{n^{\beta/2}}
\]

(2.35)

\[
\leq 1/n^2
\]

(2.36)

Where the last line is true for any \( \beta \geq 6 \) and \( 2 \leq k \leq n \), assumed by the lemma. Therefore, since \( 1/n^2 < \frac{1}{4 \log n} \) for all \( n \geq 3 \) the lemma is satisfied in both cases.  

\[ \Box \]

We can now prove our primary lemma which connects the performance of a contention resolution algorithm to the range finding solution it induces.

**Lemma 2.4.5.** Let \( A \) be a uniform contention resolution algorithm defined for a network with size upper bounded by \( n \geq 16 \) that solves the problem in \( t_X(n) \) expected rounds when the size is determined by random variable, \( X \). Let \( S_A \) be the sequence returned by the RF-construction run on \( A \). There exists a constant \( \alpha \geq 1 \) such that \( S_A \) solves \( (n, \alpha \log \log n) \)-range finding in expected time at most \( 4t_X(n) \) with respect to \( c(X) \).

**Proof.** The expected time complexity \( t_X(n) \) of \( A \) is calculated with respect to both the probabilistic source of network size \( X \) and the bits used by \( A \). Here we introduce the notation required to formalize and manipulate this expectation equation. Let \( S \) be the sample space of possible random bits generated by the participants running \( A \). We can also enumerate the values in \( S \) as \( s_1, s_2, \ldots, s_\ell \). As a reminder, we have \( p_k = \Pr[X = k] \). Let us also introduce \( p'_s \), for \( s \in S \), to be the probability that \( s \) are the random bits generated by participants in
A. For participant size \( k \in R(n) \) and bits \( s \in S \), let \( q(k, s) \) be the number of rounds until \( A \) solves contention resolution with \( k \) participants using bits \( s \).\(^6\) And finally, for each such \( k \) let \( \mu_k = \sum_{i=1}^{z} p'_s q(k, s_i) \) be the expected round complexity of \( A \) when we fix the participant size to \( k \). We note:

\[
t_X(n) = \sum_{k=1}^{n} \sum_{s \in S} p_k \cdot p'_s \cdot q(k, s) \tag{2.37}
\]

\[
= p_1 \left( p'_{s_1} q(1, s_1) + \ldots + p'_{s_z} q(1, s_z) \right) + \ldots + p_n \left( p'_{s_1} q(n, s_1) + \ldots + p'_{s_z} q(n, s_z) \right) \tag{2.38}
\]

\[
= p_1 \mu_1 + p_2 \mu_2 + \ldots + p_n \mu_n \tag{2.39}
\]

We now connect this equation to our calculation of the expected time complexity of \( S_A \) with respect to \( c(X) \). To do so, we first consider a variation of range finding where instead of determining a range with \( c(X) \), we determine a size \( k \) with \( X \), and then consider the problem solved when \( S_A \) arrives at a range sufficiently close to the corresponding range \([\log k]\). Let \( r_k \), for size \( k \), be the round where \( S_A \) solves range finding for the range corresponding to \( k \). We can compute the expected complexity of this variation as:

\[
p_1 r_1 + p_2 r_2 + \ldots + p_n r_n \tag{2.40}
\]

Notice, however, that for \( i \) and \( j \) corresponding to the same range, \( r_i = r_j \), meaning we can aggregate the probabilities associated with range in \( R(\log n) \), and get exactly \( E[Z] \), defined with respect to \( c(X) \) (where here \( E[Z] \) is the expected time complexity of \( S_A \) as dictated by \( c(X) \)). The above equation, in other words, is an elaborated form of \( E[Z] \). This

\(^6\)A technicality: this understanding of contention resolution complexity assumes that the ids of the participants do not matter, only their number. This is clearly the case for uniform algorithms of the type studied here, in which participants broadcast according to a fixed schedule and do not make use of their ids in determining their behavior. The other technicality is that we do not assume shared randomness.
elaboration is useful because it simplifies our connection of \( E[Z] \) to \( t_X(n) \), the expected complexity of \( A \), calculated above.

In particular, we will next argue that for each \( i \), \( r_i \leq 4\mu_i \). To do so, we consider two cases for a given \( \mu_i \):

- **Case 1: \( \mu_i \leq \log n \):** Here we deploy a key operational property of \( A \): if its expected complexity for a given participant size is small, it must feature a good probability for that participant size early on. Formally, assume for contradiction that the first \( 2\mu_i \) probabilities in \( A \) fall outside the range from \( 1/(\beta \cdot \log n \cdot i) \) to \( (\beta \log n)/i \) for a fixed \( \beta \geq 6 \). By Lemma 2.4.4, the probability of success in each of these rounds is strictly less than \( 1/(4 \log n) \). Because \( \mu_i \leq \log n \), a union bound establishes that the probability that at least one of these rounds succeeds is strictly less than 1/2. It would follow that the probability that \( A \) succeeds in the first \( 2\mu_i \) rounds with network size \( i \) is less than 1/2, contradicting the assumption that \( \mu_i \) is the expected time complexity in this context.

It follows then that there is a probability \( p^* \) between the values of \( 1/(\beta \cdot \log n \cdot i) \) to \( (\beta \log n)/i \) in the first \( 2\mu_i \) rounds of \( A \). In RF-construction, this probability becomes guess \( x = \lceil \log (1/p^*) \rceil \). Therefore:

\[
x \geq \left\lfloor \log \left( \frac{i}{\beta \log n} \right) \right\rfloor \quad (2.41)
\]

\[
= \lceil \log i - \log (\beta \log n) \rceil \quad (2.42)
\]

\[
= \lceil \log i - (\log \log n + \log \beta) \rceil. \quad (2.43)
\]
Furthermore, since we assume \( n \geq 16 \) and \( \beta \geq 6 \), we have that both \( \log \log n \geq 2 \) and \( \log \beta \geq 2 \). Therefore, by Lemma 1.3.9, it follows that

\[
[\log i - (\log \log n + \log \beta)] \geq [\log i - \log \beta \log \log n].
\]  

(2.44)

Bounding the other direction, and applying a similar derivation, we get:

\[
x \leq [\log (\beta \cdot \log n \cdot i)]
\]  

(2.45)

\[
\leq [\log i + \log \beta \log \log n].
\]  

(2.46)

It follows that, for \( \alpha \geq \log (\beta) \), the slot in \( S_A \) corresponding to \( p^* \) solves the \((n, \alpha \log \log n)\)-range finding. Because the construction of \( S_A \) interleaves values between those corresponding to the probabilities in \( A \), the position of \( p^* \) in \( S_A \) could be up to a factor of 2 larger than its position in \( A \). Therefore, it shows up within the first \( 4\mu_i \) positions in \( S_A \), satisfying our claim.

- **Case 2: \( \mu_i > \log n \):** This is the easier case. By the construction of \( S_A \), during the first \( 2 \cdot [\log n] \) rounds we interleave values corresponding to all ranges. Therefore, by definition,

\[
r_i \leq 2 \cdot [\log n]
\]  

(2.47)

\[
< 2 \cdot (\log n + 1).
\]  

(2.48)
Since we assume $n \geq 16$ and therefore $\log n \geq 4 > 1$, it follows that

\[
2 \cdot (\log n + 1) < 2 \cdot (\log n + \log n) = 4 \log n < 4\mu_i \Rightarrow r_i \leq 4\mu_i.
\]

Pulling together the pieces, we have established that

\[
E[Z] = p_1 r_1 + p_2 r_2 + \ldots + p_n r_n \leq p_1 (4\mu_1) + p_2 (4\mu_2) + \ldots + p_n (4\mu_n) = 4 (p_1 \mu_1 + p_2 \mu_2 + \ldots + p_n \mu_n) = 4t_X(n),
\]

as claimed by the lemma. \qed

We now have the required support to prove Theorem 2.4.2.

\textbf{Proof.} (of Theorem 2.4.2) Fix some uniform contention resolution algorithm $A$ that solves contention resolution in expected time $t_X(n)$ when run in a network of size $n$ with no collision detection and a participant size determined by $X$. By Lemma 2.4.5, there exists a constant $\alpha \geq 1$, such that the range finding sequence $S_A$ constructed by applying RF-construction on $A$, solves $(n, \alpha \log \log n)$-ranging finding in expected time $T \leq 4t_X(n)$ when ranges are drawn from $c(X)$. Applying Lemma 2.4.3 further tells us $T \geq \Omega\left(\frac{2^{H(c(X))}}{\alpha \log \log n}\right)$.

It then follows that $4t_X(n) \geq \Omega\left(\frac{2^{H(c(X))}}{\alpha \log \log n}\right) \Rightarrow t_X(n) = \Omega\left(\frac{2^{H(c(X))}}{\log \log n}\right)$. \qed
2.4.2 Lower Bound for Collision Detection

We now adapt the techniques used in the preceding section to achieve an entropy-based lower bound for the setting with collision detection. Our goal is to prove the following:

**Theorem 2.4.6.** Fix a uniform algorithm $A$ for a network with size upper bounded by $n \geq 4$. Let $t_X(n)$ be the expected round complexity for $A$ to solve contention resolution on a channel with collision detection and the number of participants determined by random variable $X$. It follows that

$$t_X(n) \geq \frac{1}{16} H(c(X)) - O(\log \log \log \log n).$$  \hfill (2.57)

For the maximum possible entropy $H(c(X)) = \log \log n$, this bound asymptotically matches the best known upper bound due to Willard [102], which requires $O(\log \log n)$ rounds. It also provides an arguably simpler and more intuitive approach than the original lower bound from [102], which deployed a more complex probabilistic counting argument to establish $\log \log n - O(1)$ rounds are necessary for uniform algorithms.

We note that the appearance of a quadruple logarithm is unusual, but straightforward to explain in our context. In the argument that follows we seek a probability within a factor of $\frac{1}{\log \log n}$ from the optimal probability for the current participant size. This is a log factor closer than in our argument for no collision detection, as the shorter executions can handle smaller error probabilities. Recall, within our condensed support $R(\log n)$, each $i \in R(\log n)$ is associated with probability $2^{-i}$. So if $2^{-i}$ is the optimal probability, then a range $j$ that is within a distance of $\log \log \log n$ from $i$ will yield a probability within a factor of $\frac{1}{\log \log n}$.
e.g.,

\[ 2^{-j} = 2^{-(i + \log \log \log n)} \]  
\[ = \frac{1}{2^{\log \log \log n}} \]  
\[ = \frac{1}{2^i \log \log n} \]

(2.58)  
(2.59)  
(2.60)

In the coding scheme used below, as in the no collision detection argument, a code contains a distance value from 0 to \( O(\log \log \log n) \), which requires \( O(\log \log \log \log n) \) bits. Our proof below follows the same structure as in the no collision detection case. We differ, however, in how we construct our range finding solution and bound such solutions from an information theory perspective.

**Tree Range Finding**

We now define the range finding problem with collision detection. Once again, the problem is parameterized with a network size \( n \) and range expressed as a function \( f(n) \) of this size. A solution to this version of the problem takes the form of a binary tree \( T \) where each node is labeled with a value from the range \( R(\log n) \). Each round \( r \), the value of a different node, \( v_r \), of \( T \) is considered depending on a target value \( v \).

For a target value \( v \), we define \( v_r \) recursively where \( v_1 \) is the root of \( T \). Namely, \( v_r \) is the node of \( T \) which is indicated by the binary path \( B = b_1, b_2, \ldots, b_{r-1} \) where \( b_i = 0 \) if \( v < v_i \) and \( b_i = 1 \) otherwise (a 0 bit means extend the path by descending to the left sub-tree and a 1 bit means descend to the right). We say a binary tree \( T \) solves the \((n, f(n))\)-range finding problem in \( t \) steps for a given target \( v \in R(\log n) \), if a node \( v_t \) (again, defined with respect to target value, \( v \)) is the first node in the sequence \( v_1, v_2, \ldots, v_t \) such that \( |v_t - v| \leq f(n) \).

Once again, to handle probabilistic selections of the target from a known distribution, fix a random variable \( Y \) that takes values from \( R(\log n) \). We say \( T \) solves \((n, f(n))\)-range finding
in expected time \( t \) with respect to \( Y \), if \( t \) is the expected step at which \( T \) solves the problem when \( v \in R(\log n) \) is determined by \( Y \).

**Bounding Tree Range Finding Using Entropy**

Assume \( T \) is a binary tree that solves \((n, \alpha \log \log \log n)\)-range finding in expected time \( T \) with respect to some distribution \( Y \) defined over \( R(\log n) \). By interpreting \( T \) as a code, we can deploy Shannon’s source coding theorem to arrive at the following bounds:

**Lemma 2.4.7.** For some constant \( \alpha \geq 1 \) and network size \( n > 1 \), let \( T \) be a labeled binary tree that solves \((n, \alpha \log \log \log n)\)-range finding. Assume the target range be determined by random variable \( Y \). Let \( Z \) be the random variable describing the complexity of solving range finding using \( T \) with respect to \( Y \). It follows:

\[
E[Z] \geq \frac{1}{2} H(Y) - O(\log \log \log \log n),
\]

(2.61)

where \( H \) is the entropy function.

**Proof.** Fix some \( n, \alpha, T \) and \( Y \) as specified by the theorem statement. We can use \( T \) to design a code for transmitting symbol \( x \in R(\log n) \) over a noiseless channel as follows:

- Initialize the sender and receiver with tree \( T \).
- Let \( \ell \) be the round in which \( T \) solves the problem with respect to target, \( x \in R(\log n) \).
- To communicate \( x \), the sender transmits \((R\lceil \log E[Z] \rceil(\ell), p, |d|, b)\) to the receiver, where:
  - \( p \) is a binary sequence describing a path from the root to \( v_\ell \) (defined with respect to \( T \) and \( x \)) within distance \( \alpha \log \log \log n \) of \( x \) (note that by definition \( \ell = |p| \)),
  - \( d = x - v_\ell \), and
\[ -b = \begin{cases} 
0 & \text{if } d < 0 \\
1 & \text{if } d \geq 0 
\end{cases} \]
indicates the sign of \(d\).

- The receiver can then locally calculate \(x\) by traversing to the node labeled \(v\) in \(T\) then adding \(d\).

We now relate the expected code length with this scheme with the expected complexity of range finding. For a given \(x \in R(\log n)\), that is solved by \(T\) in \(\ell\) steps by node \(v_\ell\) of \(T\), the code length of this scheme is upper bounded by \(|R_{\log E[Z]}(\ell)| + \ell + \lceil \log (\alpha \log \log \log n + 1) \rceil + 1\), where the quadruple log factor encodes the distance of \(v_\ell\) from \(x\). Noting that \(\lceil \log (\alpha \log \log \log n + 1) \rceil \leq \lceil \log (\alpha \log \log n) \rceil + 1\), since the value of \(\ell\) is dictated by random variable, \(Z\), the expected code length is then upper bounded as:

\[
\mathbb{E}[|R_{\log E[Z]}(Z)|] + \mathbb{E}[Z] + \lceil \log (\alpha \log \log n) \rceil + 2 \\
= \mathbb{E}[|R(Z)|] + \mathbb{E}[Z] + \lceil \log (\alpha \log \log n) \rceil + 2 
\]  

(2.62)  

(2.63)

Once again, by Lemma 2.4.1 we then have that:

\[
\mathbb{E}[|R(Z)|] + \mathbb{E}[Z] + \lceil \log (\alpha \log \log n) \rceil + 2 \\
\leq \log \mathbb{E}[Z] + 3 + \mathbb{E}[Z] + \lceil \log (\alpha \log \log n) \rceil + 2 \\
\leq 2\mathbb{E}[Z] + \lceil \log (\alpha \log \log n) \rceil + 5 
\]  

(2.64)  

(2.65)  

(2.66)

Shannon’s Source Coding Theorem (Theorem 2.3.2) tells us that the average code length of this scheme is lower bounded by the entropy of \(Y\). It follows:

\[
H(Y) \leq 2\mathbb{E}[Z] + \lceil \log (\alpha \log \log n) \rceil + 5 \\
\Rightarrow \mathbb{E}[Z] \geq \frac{1}{2}H(Y) - O(\log \log \log n),
\]

(2.67)  

(2.68)
as claimed by the theorem.

**Solving Range Finding with Contention Resolution**

Here we transform a solution to contention resolution with collision detection to a binary tree that solves range finding with a related expected complexity. We begin by defining an algorithmic process for transforming a uniform algorithm $A$ into a range finding tree $T_A$.

As a uniform contention resolution algorithm that assumes collision detection, $A$ is formalized as a function that maps the sequence of collisions and silences detected so far into a probability for all participants to use during the next round. A history of length $r$ can be captured by a bit sequence $b_1b_2\ldots b_r$, where $b_i = 0$ means silence was detected in round $i$ and $b_i = 1$ means a collision was detected.

We can interpret $A$ as a binary tree (with infinite depth) where each node is labeled with a probability. In particular, interpret each input string $s = b_1b_2\ldots b_r$ as specifying a particular node at depth $r$, reached in a $r$-step traversal starting from the root, where at step $i$ we descend the to left sub-tree if $b_i = 0$ and descend to the right sub-tree if $b_i = 1$. We label this node with probability $A(s)$; that is, the probability mapped to by the algorithm given the path specified by the input string, $s$.

Let $T_1$ be this binary tree labeled with probabilities. We next create tree $T_2$ by replacing each label $\ell$ in $T_1$ with its related range: $[\log (1/\ell)]$, as in our no collision detection construction. Let $T^*$ be the canonical binary tree of depth $[\log \log n]$ labeled with all the values in $R(\log n)$. To arrive at our final range finding solution $T_A$, we must "insert" $T^*$ into $T_2$ such that the root of $T^*$ is placed at depth $[2 \log \log n]$ in $T_2$. There are many equally useful ways to perform this insertion. Assume for now that we just follow the left-most path through $T_2$, and when arrive at node $v$ with depth $[2 \log \log n] - 1$, we simply remove $v$’s children and instead make the root of $T^*$ the only child of $v$. We refer to this resulting tree as $T_A$. 

51
We now analyze the quality of the range finding tree $T_A$ produced by our above procedure. We begin with a useful probability observation.

**Lemma 2.4.8.** Assume in a given round of a uniform contention resolution algorithm that the $1 < k \leq n$ participants each decide to transmit with a probability $p$ such that $p < \frac{1}{\beta \log \log n}$ or $p > \frac{\beta \log \log n}{k}$, for some constant $\beta \geq 8$. For $n \geq 4$, the probability exactly one participant transmits is strictly less than: $\frac{1}{4 \log \log n}$.

**Proof.** Fix any $\beta \geq 8$. In the case where $p < \frac{1}{k \beta \log \log n}$, a simple union bound over all $k$ nodes yields that the probability any node transmits is at most

\[
kp < \frac{k}{k \beta \log \log n} = \frac{1}{\beta \log \log n} \leq \frac{1}{4 \log \log n} \Rightarrow kp < \frac{1}{4 \log \log n}
\]

This satisfies the lemma statement for the case $p < \frac{1}{k \beta \log \log n}$.

Again recall that the number of nodes which transmit is given by the binomial distribution $B(k, p)$ where $E[B(k, p)] = kp$. Let $\mu = E[B(k, p)]$ denote this expectation and note
that since we assume $n \geq 4$, in the case that $p > \frac{\beta \log \log n}{k}$ we have,

\[ \mu = kp \]
\[ > k \cdot \frac{\beta \log \log n}{k} \]
\[ = \beta \log \log n \]
\[ \geq \beta \log \log (4) \]
\[ = \beta \]
\[ \Rightarrow 1 - \frac{1}{\mu} \geq 1 - \frac{1}{\beta} \]

Therefore, if we treat $B(k, p)$ as the sum of independent random variables with expectation at least $\mu$, we can invoke the Chernoff bound from Theorem 1.3.4 for $\varepsilon = 1 - \frac{1}{\mu}$ to give that,

\[ \Pr[B(k, p) = 1] \leq \Pr[B(k, p) \leq 1] \]
\[ = \Pr\left[B(k, p) \leq \left(1 - \left(1 - \frac{1}{\mu}\right)\right) \cdot \mu\right] \]
\[ \leq \exp\left(-\frac{1}{2} \cdot \left(1 - \frac{1}{\mu}\right)^2 \cdot \mu\right) \]
\[ \leq \exp\left(-\frac{1}{2} \cdot \left(1 - \frac{1}{\beta}\right)^2 \cdot \mu\right). \]

Where Line 2.82 uses the inequality derived on Line 2.78. Simplifying further,

\[ \exp\left(-\frac{1}{2} \cdot \left(1 - \frac{1}{\beta}\right)^2 \cdot \mu\right) < \exp\left(-\frac{1}{2} \cdot \left(1 - \frac{1}{\beta}\right)^2 \cdot \beta \log \log n\right) \]
\[ = \exp\left(-\frac{\beta}{2} \cdot \left(1 - \frac{1}{\beta}\right)^2 \cdot \log \log n\right) \]
\[ = \frac{1}{\log^{(\beta/2)(1-1/\beta)^2} n} \]
Since we assume $\beta \geq 8$, we can simplify the exponent as follows,

\[
\frac{\beta}{2} \left( 1 - \frac{1}{\beta} \right)^2 \geq \frac{8}{2} \left( 1 - \frac{1}{8} \right)^2
\]

\[
= 4 \cdot \frac{49}{64}
\]

\[
> 2.
\]

Therefore, for $\beta \geq 8$,

\[
\frac{1}{\log^{(\beta/2)(1-1/\beta)^2} n} < \frac{1}{\log^2 n}.
\]

Since $\log^2 n \geq 4 \log \log n$ for sufficiently large values of $n$ (e.g., for $n \geq 4$), we get that,

\[
\Pr[B(k, p) = 1] < \frac{1}{\log^2 n}
\]

\[
\leq \frac{1}{4 \log \log n}
\]

\[
\Rightarrow \Pr[B(k, p) = 1] < \frac{1}{4 \log \log n},
\]

thus satisfying the lemma for both cases. \(\square\)

We now make our main argument regarding $T_A$ as a solution to range-finding.

**Lemma 2.4.9.** Let $A$ be a uniform contention resolution algorithm defined for a network size upper bounded by size $n \geq 4$ with collision detection, that solves the problems in $t_X(n)$ rounds in expectation when the network size is determined by random variable $X$. Let $T_A$ be the range finding tree returned by applying our above procedure to $A$. There exists a constant $\alpha \geq 1$ such that $T_A$ solves $(n, \alpha \log \log \log n)$-range finding in expected time no more than $8t_X(n)$ with respect to $c(X)$. 

54
Proof. Redeploying the same argument as in the proof Lemma 2.4.5, we can express $t_X(n)$ in the following useful form:

$$p_1\mu_1 + p_2\mu_2 + \ldots + p_n\mu_n,$$

(2.93)

where $p_i = \Pr[X = i]$ and $\mu_i$ is the expected round complexity of $A$ given that $i$ is the network size. We use this form of $t_X(n)$ to help bound the expected complexity of $T_A$ with respect to $c(X)$. Also, as in the proof of Lemma 2.4.5, we can consider a variation of range finding where instead of directly determining a range with $c(X)$, we determine a size $k$ with $X$, and then transform this into the corresponding range $[\log k]$. As argued earlier, calculating the expected complexity of $T_A$ for this variation is mathematically equivalent to the complexity for the standard variation in which values are drawn from $c(X)$.

We compute the expected complexity of $T_A$ running this variation as:

$$p_1r_1 + p_2r_2 + \ldots + p_nr_n$$

(2.94)

where $r_i$ is the depth of the first value in tree $T_A$ to fall within a sufficient distance of the range $[\log i]$ corresponding to size $i$. We will next argue that for each $i$, $r_i \leq 2\mu_i$.

To do so, we consider two cases for a given $\mu_i$:

- Case 1: $\mu_i \leq \log \log n$: Consider $T_1$, our interpretation defined earlier of $A$ as a binary tree labeled with $A$’s broadcast probabilities. We argue here that within depth $2\mu_i$ of $T_1$ we can find a probability within a multiplicative factor of $\beta \log \log n$ of $1/i$, where the constant $\beta$ comes from Lemma 2.4.8.

To make this argument, assume for contradiction that no probability within depth $2\mu_i$ was within this range. By Lemma 2.4.8, given any execution of $A$, during any of the first $2\mu_i \leq 2\log \log n$ rounds of the execution, the probability of solving contention
resolution in that round is strictly less than \(1/(4 \log \log n)\). Therefore, by a union bound over these rounds, the probability of succeeding in the first \(2\mu_i\) rounds is itself strictly less than

\[
\frac{2\mu_i}{4 \log \log n} \leq \frac{2 \log \log n}{4 \log \log n}
\]

\[
= 1/2.
\]

(2.95)  
(2.96)

This contradicts the assumption that the expected round in which contention resolution is solved is \(\mu_i\).

We have established, therefore, that somewhere in the first \(2\mu_i\) levels of the tree is a probability \(p^*\) in the range \([\frac{1}{i} \cdot \frac{1}{\beta \log \log n}, \frac{1}{i} \cdot \frac{\beta}{\log \log n}]\). When constructing \(T_A\), this value will be replaced by \(x = \left\lceil \log \left(\frac{1}{p^*}\right) \right\rceil\), which for a sufficiently large constant \(\alpha\) (defined with respect to \(\beta\), is within distance \(\alpha \log \log \log n\) of the target \([\log (1/i)]\), solving range finding for this value. It follows that \(r_i \leq 2\mu_i\).

- **Case 2: \(\mu_i > \log \log n\):** This is the easier case. By the construction of \(T_A\), every range shows up somewhere between depth \([2 \log \log n]\) and \([2 \log \log n] + [\log \log n]\).\(^7\) Therefore, it follows that the depth at which the problem is solved for size \(i\) is at most

\[
[2 \log \log n] + [\log \log n] \leq [2 \log \log n + \log \log n] + 1
\]

\[
= [3 \log \log n] + 1
\]

\[
< 3 \cdot (\log \log n + 1) + 1
\]

\[
= 3 \log \log n + 4.
\]

(2.97)  
(2.98)  
(2.99)  
(2.100)

\(^7\)Since \(T^*\) was inserted at depth \([2 \log \log n]\) and has height at most \([\log \log n]\).
Since we assume \( n \geq 4 \) and therefore \( \log \log n \geq 1 \),

\[
3 \log \log n + 4 \leq 3 \log \log n + 4 \log \log n
\]

\[
= 7 \log \log n
\]

\[
< 7\mu_i
\]

\[
< 8\mu_i.
\]

Combined, these two cases establish the for a sufficiently large constant \( \alpha \), the expected complexity of \((n, \alpha \log \log \log n)\)-range finding with \( T_A \) and \( c(X) \) is within a factor of 8 of the expected complexity of \( A \) with \( X \), as claimed by the lemma.

We can now prove Theorem 2.4.6.

**Proof.** (of Theorem 2.4.6.) Fix some uniform contention resolution algorithm \( A \) that solves contention resolution in expected time \( t_X(n) \) when run in a network of size \( n \) with collision detection and a participant size determined by \( X \). By Lemma 2.4.9, there exists a constant \( \alpha \geq 1 \), such that the range finding tree \( T_A \) constructed by applying our procedure to \( A \), solves \((n, \alpha \log \log \log n)\)-ranging finding in expected time \( T \leq 8t_X(n) \) when ranges are drawn from \( c(X) \). Applying Lemma 2.4.7 further tells us that \( T \geq \frac{1}{2}H(c(X)) - O(\log \log \log \log n) \). It follows that \( 8t_X(n) \geq \frac{1}{2}H(c(X)) - O(\log \log \log \log n) \Rightarrow t_X(n) \geq \frac{1}{16}H(c(X)) - O(\log \log \log \log n) \).

\[
2.5 \text{ \textbf{Upper Bounds for the Contention Resolution Problem with Predictions}}
\]

We now turn our attention to upper bounds for the contention resolution problem with predictions. The challenge here is to design algorithms which can effectively exploit the provided probability distribution over the possible network sizes. Once again, we begin with the no collision detection setting.
2.5.1 Upper Bound for No Collision Detection

In this section we introduce an algorithm for solving contention resolution without collision detection. We assume the algorithm is provided as input the definition of a random variable $Y$ defined over network sizes. Let $X$ be the actual random variable from which the sizes will be drawn. We will produce our round complexity bounds with respect to the KL divergence between $Y$ and $X$, quantifying the cost of inaccuracy in predictions.

In Section 2.4.1, we proved that with accurate predictions (i.e., $Y = X$), $\Omega \left( \frac{2^{H_Y(Y)}}{\log \log n} \right)$ rounds are needed in expectation to solve contention resolution. Our goal here is produce a result that comes close to matching this exponential bound. To do so, we analyze a natural strategy: trying range predictions in $c(Y)$ in decreasing order of likelihood. We prove that with constant probability, this strategy solves the problem in $2^T$ rounds, where $T = 2H(c(X)) + 2D_{KL}(c(X) \parallel c(Y))$. If every probability used by $Y$ is within some bounded constant factor of the corresponding probability in $X$, this reduces to $2^{2H(c(Y))}$, which is within the same general form as the lower bound but with an extra factor in the exponent. It is not obvious how to remove the extra factor in the exponent. Indeed, as elaborated in Section 2.2, we have reason to believe that some exponential factor greater than 1 is necessary for this algorithm.

Algorithm

Let $\pi = \langle \pi_1, \ldots, \pi_{\log n} \rangle$ represent an ordering over $R(\log n)$ sorted by non-decreasing probability of the corresponding range with respect to $c(Y)$. In other words, for all $i < j$, $\Pr[k \in (2^{\pi_i-1}, 2^{\pi_i})] \geq \Pr[k \in (2^{\pi_j-1}, 2^{\pi_j})]$. Our algorithm (which seeks to solve the problem
with at least constant probability\(^8\)) then takes \([\log n]\) rounds where in round \(i\) each node broadcasts with probability exactly \(1/2^i\).

**Complexity Analysis**

We prove the following time complexity statement regarding the above algorithm.

**Theorem 2.5.1.** In the above algorithm, a node broadcasts alone after at most \(O(2^T)\) rounds where \(T = 2H(c(X)) + 2D_{KL}(c(X) \parallel c(Y))\) rounds with probability at least \(1/16\), where \(X\) is the actual distribution over the network sizes and \(Y\) is the provided distribution.

First we prove that in round \(i\) that if \(k \in (2^{i-1}, 2^i]\) then a single node broadcasts alone during the round with probability at least \(1/8\).

**Lemma 2.5.2.** Fix a round \(i > 0\). For any \(k \geq 2\) such that \(k \in (2^{i-1}, 2^i]\), a single node broadcasts alone in round \(i\) with probability at least \(1/8\).

**Proof.** If \(k\) nodes broadcast with probability \(p\), the probability that a single node broadcasts alone is equal to \(\Pr[B(k, p) = 1]\) where \(B\) denotes the binomial distribution. In round \(i\), nodes broadcast with probability \(p = 1/2^i\) which, by our assumption that \(k \in (2^{i-1}, 2^i]\), then means that \(p \in \left(\frac{1}{2^i}, \frac{1}{k}\right]\). Therefore, the probability that a single node broadcasts in round \(i\) can be derived as follows:

\[
\Pr[B(k, p) = 1] = \binom{k}{1} p(1-p)^{k-1} = k \cdot \frac{1}{2^i} \left(1 - \frac{1}{k}\right)^{k-1} = (1/2)(1 - 1/k)^{k-1}.
\]

\(^8\)Because we seek only a constant probability result, we analyze here only one pass through all \(\log n\) possible probabilities. In the pursuit of good expected times, you would instead cycle through these probabilities in a clever manner. We do not prove expectation bounds on this algorithm here, and note that an expectation close to our constant probability result is not necessarily easily obtained.
Since $k \geq 2$ and therefore $1/k \in (0, 1/2]$, Lemma 1.3.8 then gives us for $x = 1/k$ that $1 - 1/k \geq 4^{-1/k}$. Therefore, we have that

\[
(1/2)(1 - 1/k)^{k-1} \geq 4^{-(1/k)(k-1)}/2 \geq 4^{-k/k}/2 \geq 1/8, \quad (2.108)
\]

\[
\Rightarrow \Pr[B(k, p) = 1] \geq 1/8. \quad (2.111)
\]

Therefore the lemma follows. \qed

Consider an optimal variable-length code $f$ over the values of $R(\log n)$ based on their probabilities according to $c(Y)$. We will relate this code to the number of rounds required by our algorithm.

**Lemma 2.5.3.** With probability at least $1/8$, the above algorithm takes at most $2^{S+1}$ rounds where $S$ is the random variable representing the code lengths of $f$ applied to $c(Y)$.

**Proof.** Let $k$ be the actual size of the network and fix a round $i > 0$. From Lemma 2.5.2 we know that in round $i$ of our algorithm where $k \in (2^{\pi_i-1}, 2^{\pi_i}]$ we succeed with probability at least $1/8$. Fix this round $i$ and let $f(\pi_i)$ be the codeword assigned to $\pi_i$ under $f$ according to $c(Y)$. Note that since we assume $f$ is optimal we have that $|f(\pi_i)| \geq |f(\pi_j)|$ for all $j < i$. Therefore, with $S$ as the random variable over this codeword length we have that the total number of estimates preceding $\pi_i$ is at most $2^{S+1} - 1$ and the number of rounds for our algorithm to test a value that yields constant success probability is given by $2^{S+1}$. \qed

We can now prove our main theorem.

**Proof.** (of Theorem 2.5.1.) From Lemma 2.5.3 we have that the distribution over the number of rounds of our algorithm is given by $2^S + 1$. Furthermore from Theorem 2.3.3
we have that for an optimal variable-length code $f$ over distribution $c(Y)$ where $c(X)$ is the actual distribution, $\mathsf{E}[S] \leq H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1$. Therefore, applying Markov’s inequality from Theorem 1.3.1 to $S$ for $a = 2 \cdot (H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)$ gives us that

$$\Pr[S \geq 2 \cdot (H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)] \leq \frac{\mathsf{E}[S]}{2 \cdot (H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)} \leq \frac{H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1}{2 \cdot (H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)} \leq \frac{1}{2}.$$  

(2.112)  

(2.113)  

(2.114)

Therefore, the code length corresponding to the correct estimate is at most $2(H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)$ with probability at least $1/2$. Multiplying the probability of this event with the success probability of $1/8$ from Lemma 2.5.2 then satisfies the theorem. □

Note that since $D_{KL}(c(X) \parallel c(X)) = 0$, if instead our algorithm learns from the actual distribution $c(X)$ we can bound the running time with respect to the entropy of $c(X)$ alone.

**Corollary 2.5.4** (of Theorem 2.5.1). *In the above algorithm, a single node broadcasts alone after at most $O(2^{2H(c(X))})$ rounds with probability at least $1/16$, where $X$ is both the actual distribution over the network sizes and the distribution provided to the algorithm.*

2.5.2 **Upper Bound for Collision Detection**

We now turn our attention to a setting with collision detection. We once again make use of an optimal variable-length code $f$ constructed for source $c(Y)$, and use its code words to structure our algorithm’s behavior.
Algorithm

We group the values from $R(\log n)$ into $x$ equivalence classes based on the length of their code according to $f(c(Y))$. Let $\pi_1, \ldots, \pi_x$ be these classes where the $i$th class contains all values from $c(Y)$ which have codes of length exactly $i$. More formally, for all $i \in R(x)$ let $\pi_i = \{ j \in R(\log n) \mid |f(j)| = i \}$.

Our algorithm then divides rounds into $x$ phases, one dedicated to each $\pi_i$. In the phase for $\pi_i$, we use transmissions and collision detection to perform a binary search over the possible network size ranges represented by the values in class $\pi_i$. The binary search algorithm we use is an adaptation of the classical strategy presented in [102] which searches over a collection of $\lceil \log n \rceil$ geometrically distributed network size guesses, transmitting with a corresponding probability for each guess, and using collision and silence to indicate if a guess is too small or too large, respectively.

More formally, when searching over $\pi_i$, the nodes order the ranges in $\pi_i$ from smallest to largest. They then broadcast with probability $2^{-m}$, where $m$ is the median of the ranges in $\pi_i$. If a collision is detected, the nodes then recurse over the values greater than $m$. Otherwise, if silence is detected, they recurse over the values smaller than $m$. If a single node broadcasts, then contention resolution is solved.

The algorithm proceeds through the phases in order of the classes; i.e., $\pi_1$ then $\pi_2$, and so on. If the problem is not solved during the search for class $\pi_i$, then it moves on to search $\pi_{i+1}$. As without our no collision detection algorithm, we present this result here as a one-shot attempt that solves contention resolution with constant probability. For higher probability, it can be repeated, but we do not analyze this form.

Complexity Analysis

Our goal is to prove the following about the time complexity of our algorithm.
Theorem 2.5.5. In the above algorithm, with constant probability: contention resolution is solved in $O\left( (H(c(X)) + D_{KL}(c(X) \parallel c(Y))^2 \right)$ rounds, where $X$ corresponds to the actual distribution over the network sizes and $Y$ is the distribution provided to the algorithm.

We first analyze our algorithm with respect to the random variable describing the code lengths generated by $f$ when symbols are drawn from $c(Y)$.

Lemma 2.5.6. With constant probability, the algorithm solves contention resolution in $O(S^2)$ rounds, where $S$ is the random variable describing the length of the code generated by $f$ when applied to $c(Y)$.

Proof. The analysis of Willard’s strategy from [102] provides the following useful property of this search strategy: if class $\pi_j$ contains the target value $t = \lceil \log k \rceil$, where $k$ is the actual network size, then with constant probability the search of this class solves contention resolution. Note, by definition, all code words corresponding to values in $\pi_j$ are of length $j$. Therefore, $|\pi_j| \leq 2^j$, meaning the search for phase $j$ requires no more than $O(\log |\pi_j|) = O(j)$ rounds.

Therefore, if the target is in $\pi_j$, which is equivalent to saying the symbol has a code of length $j$, then we will complete the search for phase $j$ after first completing searches for phases $1, 2, \ldots, j - 1$. Total, this requires $O(j^2) = O(S^2)$ rounds. Therefore, we solve the problem with constant probability after $O(S^2)$ rounds, as claimed.

We now prove our final result by leveraging the optimality of the code to express the expected code length with respect to the entropy of $c(Y)$.

Proof. (of Theorem 2.5.5) We have from Lemma 2.5.6 that the algorithm solves contention resolution with constant probability in $O(S^2)$ rounds where $S$ is the random variable describing the length of the code word for the target value drawn from $c(Y)$. We have from our optimal code $f$ and Theorem 2.3.3 that the expected value of $S$ is at most
\[ H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1. \]

Therefore, as in the proof of Theorem 2.5.1, applying Markov’s inequality from Theorem 1.3.1 for \( a = H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1 \), we have that

\[
\Pr[S \geq 2 \cdot (H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)] \leq \frac{E[S]}{2 \cdot (H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)}
\]

(2.115)

\[
\leq \frac{H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1}{2 \cdot (H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)}
\]

(2.116)

\[
= \frac{1}{2}.
\]

(2.117)

Therefore, the algorithm solves contention resolution with probability at least 1/2 in at most

\[
O(S^2) \leq O\left(4(H(c(X)) + D_{KL}(c(X) \parallel c(Y)) + 1)^2\right)
\]

(2.118)

\[
= O\left((H(c(X)) + D_{KL}(c(X) \parallel c(Y)))^2\right)
\]

(2.119)

rounds as required. \( \square \)

As in Section 2.5.1, if instead our algorithm learns from the actual distribution \( c(X) \) we can bound the running time with respect to the entropy of \( c(X) \) alone.

**Corollary 2.5.7 (of Theorem 2.5.5).** In the above algorithm, if the input distribution \( Y \) equals the actual network size distribution \( X \): a single node broadcasts alone after at most \( O(H^2(c(X))) \) rounds.

Once again, this comes from the fact that \( D_{KL}(c(X) \parallel c(X)) = 0 \) for any \( X \).
We next turn our attention to the transport layer of the wireless network stack, augmenting existing algorithmic theory to unlock powerful strategies for end-to-end communication protocols, robust to the complex topologies induced by certain peer-to-peer settings.

Whereas the data link layer coordinates exchanges over a single link which connects adjacent hosts, the transport layer extends this functionality to provide reliable communication over much greater distances (traversing paths which are potentially composed of many links). Naturally, achieving this sophisticated functionality requires solving a number of difficult problems, such as the scheduling and routing of a large number of data packets. To this end, transport layer protocols seek to maximize the throughput of information throughout the network by devising efficient strategies which minimize congestion.

This functionality is closely mirrored by the gossip problem of distributed algorithms theory. Typical settings for this problem consist of a network of \( n \) nodes, where some subset of these participants possess a set of tokens. The goal of the problem is then to spread these tokens throughout the network. In the case of one-shot gossip, which we consider in Sections 3.3 and 3.4, there is a finite set of \( k \) tokens meant to be sent to all \( n \) network participants.\(^1\) An instance of this problem is parameterized with a number of tokens, \( k \in \ldots \)

\(^1\)Note that, in contrast to Chapter 2 where \( n \) denoted an upper bound on the possible number of participants, here \( n \) denotes the actual size of the network. In the previous chapter, the true network size was represented by \( k \) while, in this chapter, \( k \) is used to describe the number of tokens in the one-shot gossip problem. While this change in notation is somewhat inconvenient, these are the most standard ways to define the problems we study.
and the problem is solved once all tokens are received by all participants then (Note that when $k = 1$, this special case of the problem is called *rumor spreading*.) There is another variation of the gossip problem which we study in Section 3.5 called *capacity*, that is concerned with "ongoing" communication. This problem definition provides certain nodes with an infinite stream of packets which must continuously be routed to a specified set of recipients. The goal of this problem is to devise a schedule for making this routing as efficient as possible, where efficiency is measured by the rate at which packets are delivered.

Given the core responsibilities of the transport layer mentioned previously, the relevance of these problems should be clear. In particular, efficient solutions to the capacity and one-shot gossip problems will most likely enable powerful methods for the transport layer’s vital role of spreading information in a network.

However, despite the importance of the transport layer, protocols for this layer have not been a subject of substantial interest in the context of distributed algorithms theory (at least for most networks). This is because by far the most popular wireless network paradigm in use today is to link wireless devices to a dedicated centralized hub, such as a cell tower or a wired access point, which then handles the bulk of the communication responsibilities. This network architecture, due to its simplicity and efficiency, has left little room to explore interesting theoretical results for communication problems in this setting. That is, until the recent emergence and rapid development of *peer-to-peer (P2P)* communications technologies, which have substantially revitalized the theoretical interest behind problems related to the transport layer.

These technologies (such as Bluetooth and WiFi-direct) allow devices to connect with nearby devices using local one-hop radio links, establishing high-throughput pairwise communication channels (e.g., 2MB/s on Bluetooth channels [9]). While the links themselves have limited range, they can be used to string together large, multi-hop ad hoc networks of wireless devices, capable of communicating without the assistance of a central access
point. These networks (called P2P networks) have increasingly become an area of significant practical interest for several reasons. Among which is a movement by manufacturers to equip modern wireless devices with native P2P hardware support (e.g., 4.1 billion new bluetooth-enabled devices in 2020 [98]) and software support (e.g., Apple’s Multipeer Connectivity (MPC) framework [35]). Another reason these networks have recently become so relevant is that they serve the important role of allowing large networks of users to stay connected when centralized wireless architecture has been disrupted. Take, for example, recent pro-democracy protests involving thousands of users where traditional cellular communication channels must be avoided as they are either disrupted or heavily censored [23].

The burgeoning popularity of these technologies in real-world applications has spurred a similar explosion in theoretical interest. The reason being that P2P communication protocols must contend with many complexities that are not encountered in the classic centralized paradigm used for most wireless communication. For example, as we have mentioned, communication links can only be formed between peers within some fixed proximity, causing most P2P networks to possess multi-hop topologies. Furthermore, due to the implementations of supporting P2P libraries, devices are also limited to only a fixed number of simultaneous connections. These considerations induce highly sophisticated communication topologies for P2P networks, invalidating many classic algorithmic results and posing many interesting theoretical questions.

In this chapter, we build on existing distributed algorithms research which seeks to answer these questions. In particular, we seek to study interesting theoretical problems which are pertinent to the transport layer of peer-to-peer networks. In Section 3.1 we start by surveying the existing work pertaining to versions of the gossip problem for peer-to-peer networks. This allows us to provide a comprehensive overview of the results we achieve in this chapter in Section 3.2. These results begin in Section 3.3 where we study the simplest such problem definition, that is one-shot gossip in an existing synchronous peer-to-peer
networking model known as the mobile telephone model, which was introduced by Ghaffari and Newport [57]. As a result, we achieve a new upper and lower bound for the fundamental problem of one-shot gossip in P2P networks. Recognizing, however, that the assumption of synchronization is unrealistic for practical implementations for real networks, in Section 3.4 we study one-shot gossip again but with this assumption removed. This requires us to develop a new asynchronous peer-to-peer model and rely on simplified algorithms which do not required synchronized rounds. The result is the first upper bound for asynchronous one-shot gossip for these networks. Finally, in Section 3.5 we return to the synchronous mobile telephone model to study the harder problem of capacity in peer-to-peer networks. Specifically, we study the version of the problem called all-to-all capacity as it bears the closest resemblance to one-shot gossip. Consequently, not only do we achieve a tight bound for the all-to-all capacity problem in the mobile telephone model, we use this result to establish a corollary regarding the complexity of one-shot gossip in the mobile telephone model. Note that our one-shot gossip results in Sections 3.3 and 3.4 first appeared in [90], while those for the all-to-all capacity problem in Section 3.5 first appeared in [34] among results for other variations of the capacity problem in the MTM.

As previously mentioned, we now discuss the extensive literature regarding different versions of the gossip problem for existing peer-to-peer networking models.

3.1 Existing Work on Gossip in Peer-to-Peer Networks

In this section, we survey the existing body of research regarding variations of the gossip problem in a variety of abstractions used to represent peer-to-peer networks.

The earliest iteration of models we consider is the classic telephone model, first explored in the work of Frieze and Grimmett [49]. Time in this model proceeds in synchronous rounds, where in each round, participants are allowed to a select a single one of
their immediate neighbors which to create a connection (i.e., make a "phone call"). Once
the connection is formed, it can be used to exchange a bounded amount of information
between the connected peers before the round concludes. A variation of the classic tele-
phone model is the random phone call model (sometimes called the gossip model) where
participants randomly choose their communication partner each round [11, 72]. While this
model is also typically synchronous, variants of this model have also been studied where
each node is assigned an independent random clock with ticks according to some defined
probability distribution [11, 61].

Admittedly, in the context of classic wireless network architectures (where commonly
information is broadcast), restricting devices to only a fixed number of outgoing commu-
nication links may make little sense. However, the reason that this limitation is enforced in
all of the models we consider is that they are necessary to accurately represent the imple-
mentations of modern P2P software libraries. For instance, the Bluetooth specification only
guarantees high-throughput communication between at most eight concurrently connected
peers, and therefore any interface built on top of these low-level protocols must adhere to
the same restrictions. Furthermore certain peer-to-peer communication technologies, such
as those supported by commodity smartphones (e.g., [35]) place even further restrictions
on the number of incoming connections which a device may participate in at one time.
This additional restriction prevents many results shown for more classic peer-to-peer net-
working models from being applicable to networks built on these libraries, known in the
literature as smartphone P2P networks. Furthermore many of these libraries allow for very
small amounts of information to be broadcast to all nearby devices, a capability which is
not possible in the models seen so far.

For this reason, the mobile telephone model (MTM) was introduced by Ghaffari and
Newport [57] to more accurately capture the behavior of modern P2P networking libraries
by modeling it on actual software interfaces exposed by modern P2P libraries. In this model
(with time proceeding in synchronous rounds) participants begin each round by broadcast a small amount of identifying information to all nearby neighbors, before selecting a single neighbor with which to connect. Critically, any device which receives some number of connection attempts can accept at most one per round as well.

This model best captures the software side of P2P libraries for modern wireless devices, however the restriction on the number of incoming connection attempts has been shown to be significant, invalidating many of the results implemented in the classic telephone model [28, 57]. That being said, recent work over the past few years has shown that, despite this restriction, there exist efficient algorithms in the MTM for a number of important distributed problems, including load balancing [32], leader election [86], and rumor spreading [57]. Of these, the problem of rumor spreading is by far the most pertinent to the transport layer’s goal of enabling end-to-end communication. The problem of rumor spreading describes the setting in which a single node \( s \in V \) (denoted the source) possesses a token (representing the rumor) and the goal is for \( s \) to disseminate this token to the remaining nodes of the graph.

This problem is extensively studied in the classic telephone model. The problem is first studied in the work by Frieze and Grimmett [49] where the model is first introduced. The work of Censor-Hillel et al. [15] produced an algorithm which completed in \( O(D + \text{polylog } n) \) expected rounds. Recall here \( D \) is the diameter of the graph (the greatest distance between any two nodes in the network). Later works achieve improved complexities in topologies that are more well-connected. These bounds are often expressed with respect to common metrics of graph connectivity, such as the conductance of the graph (see Definition 1.3.12), \( \phi \), or its vertex expansion (see Definition 1.3.11), \( \alpha \). One such work is that of Giakkoupis [59] which provides a tight bound with respect to conductance of \( \Theta((1/\phi) \log n) \) rounds with high probability. Similarly for vertex expansion, Giakkoupis [60] achieves an upper bound \( \Theta((1/\alpha) \log^2 n) \) rounds with high probability (matching a
known lower bound, making this bound tight). Lastly, in the MTM, the work of Ghaffari and Newport [57] provides a bound of $O((1/\alpha) \log^4 n)$ rounds in expectation.

_One-shot gossip_ (which we will often refer to simply as _gossip_) generalizes rumor spreading in that some subset of nodes start with distinct tokens and our goal is to ensure that every node in the graph gets every message. As was the case with rumor spreading, we can consider both single hop and multi-hop networks. The existing work on gossip is tightly coupled with that of rumor-spreading, and some preliminary results for the MTM are known. Namely, the work of Newport [85] provided the first gossip algorithm for the MTM with a complexity of $O((k/\alpha) \log^5 n)$ rounds in expectation, where $k$ is the number of distinct tokens. This was later improved upon by the work of Newport and Weaver [89] which achieves a complexity of $O((k/\alpha) \log^4 n)$ rounds with a far simpler algorithm. One-shot gossip has also been studied in works such as that of Mosk-Aoyama and Shah [84] (see Shah [96] for a complete overview).

As the name implies, one-shot gossip describes the task of spreading a _finite_ number of tokens throughout the network, at which point the problem is solved. There is also an important problem in distributed algorithms theory called _capacity_ which can be thought of as an "ongoing" version of the gossip problem. In this setting, instead an infinite stream of packets is routed continuously throughout the network. The seminal starting point for this problem in classic wireless networking models is the work of Gupta and Kumar [65].

The above problems have also been considered in many other types of models which we do not consider here. In radio networks, the work of [5] provides one of the first deterministic algorithms for rumor spreading in the setting where nodes have no knowledge of the network topology beyond their one-hop neighborhood. Further works, like those of [13, 19, 20, 22] remove even the assumption of one-hop knowledge. The state-of-the art algorithms which achieve complexities of $O(n \log^2 D)$ and $O(n \log n \log \log n)$ (with high probability in $n$) come from the works of Czumaj and Rytter [27] and De Marco [29], respectively.
Randomized algorithms are studied by [5, 74, 76]. The work of [30, 31, 95] studies the problem in geometric networks. With no knowledge $O(n)$ is known to be possible, $O(D + \log n)$ rounds in symmetric geometric networks (Dessmark and Pelc [30]), or $O(n \log n)$ rounds in a line graph. Other works [66, 30, 40, 14] study the impact of advice on broadcasting in radio networks, where some analyze the minimum number of bits needed (such as Ilicinkas, Kowalski, and Pelc [66]) while others (like Dessmark and Pelc [30]) shows that $O(D)$ time is sufficient if the advice in question provides full knowledge of the network topology.

3.2 Summary of New Results

As we have just seen, there has been an increasing interest in studying communication tasks in peer-to-peer networks. However, only recently has work been done to study these problems in models which more accurately represent the capabilities of certain peer-to-peer networks (namely smartphone peer-to-peer networks). In this thesis, we considerably extend these results in the following ways.

3.2.1 Synchronous One-Shot Gossip

Beginning with the simplest case, the state-of-the-art algorithm for one-shot gossip in the MTM (before the results presented here) is that analyzed in the work of Newport and Weaver [89] which takes $O((k/\alpha) \log^4 n)$ rounds in expectation. However, there is little reason to believe that this result is optimal for this problem. Indeed, little is known about what is optimal for this setting as there are currently no lower bounds for one-shot gossip in the MTM. In Section 3.3 we seek to prove tighter bounds for this setting.
In Section 3.3.2 we start with a lower bound to understand what we should expect to be possible given a graph of size $n$, vertex expansion $\alpha$, and number of tokens $k$. Here we prove the first lower bound for one-shot gossip in the MTM of $\Omega(k/\alpha)$ expected rounds.

We then, in Section 3.3.3, attack the problem from the perspective of upper bounds. Namely, we show an algorithm which we call random diffusion gossip that requires only $O((k/\alpha) \log n \log^2 \Delta) = O((k/\alpha) \log^3 n)$ rounds in expectation. This algorithm has each node continually advertise two pieces of information about its current rumor set: a hash of the set and its size. When faced with multiple neighbors with different rumor set hash values, a node will randomly select a recipient of a connection proposal from among those with the smallest rumor set sizes. Worth noting for our results presented in Section 3.4.2 for asynchronous one-shot gossip, this strategy is easily adapted to asynchrony as it does not explicitly use rounds unlike that of [89].

Note that the improved complexity of $O((k/\alpha) \log^3 n)$ expected rounds of our algorithm not only improves on the gossip result of [89], it also improves the best known result for rumor spreading (i.e., for $k = 1$) from [57]. Interestingly, the source of this speed-up is a new and improved version of the core technical lemma from [57], which bounds the performance of a random matching strategy in bipartite graphs.

3.2.2 ASYNCHRONOUS ONE-SHOT GOSSIP

In Section 3.4 we explore the asynchronous version of the one-shot gossip problem by removing the assumption of synchronized rounds. This is an important variation if we wish our algorithms to be implemented more easily in practice, but it is not easy to accomplish in general. Furthermore, this task necessitates we devise a new model. To this end, in Section 3.4.1 we extend the MTM to remove the synchronized round structure. We call this new model the asynchronous mobile telephone model (aMTM).
Since the pattern of communication in the asynchronous setting can be complex, this model imposes a simple but flexible structure for how processes communicate with each other. To this end, we introduce a meta-algorithm called the *aMTM interface* and provide the pseudocode in Algorithm 3. In an algorithm execution in the aMTM, each process implements this interface individually, independent of all others processes in the network. This allows us to analyze the running time of a particular instance of an algorithm and, from there, the performance of the algorithm across all concurrent network instances. In particular, we parameterize time in the aMTM with the parameter $\delta_{\text{max}}$ which denotes the maximum duration of one iteration of the behavioral loop defined by the aMTM interface. This parameter encapsulates network conditions such as bandwidth restrictions and communication overheads which are independent of the network topology and unknown to the algorithm itself (critical to the assumption of asynchrony).

In order to design one-shot gossip algorithms for our asynchronous model, we may just wish to adapt existing synchronous algorithms for the MTM. This, however, is not possible in general. Take for example the previous best algorithm for one-shot gossip in the MTM from [89]. This algorithm is not a good candidate for this purpose because it requires nodes to advertise whether or not they were involved in a connection at any point during the previous $\log n$ rounds. This behavior cannot be easily adapted to an environment with no rounds. Fortunately, a useful property of our algorithm presented in 3.3.3 is that it removes this dependence. This allows us in Section 3.4.2 to adapt this algorithm to our asynchronous setting.

3.2.3 **All-to-All Capacity**

Having extensively studied the one-shot gossip problem, we return to the synchronous MTM to study the more difficult problem of capacity in Section 3.5. Compared to one-shot gossip, this problem is concerned with what can be thought of as "ongoing" gossip;
where nodes have an infinite stream of tokens which they want to spread throughout the network (not the finite, bounded token set we have assumed thus far). There are many variations of the capacity problem, where each defines several pairs of nodes \((s, t)\) where the source \(s\) must route an infinite number of packets to sink \(t\). Whereas in our previous algorithms the goal was to minimize the time complexity, with capacity the goal is to maximize throughput, roughly defined as the worst-case rate at which packets are delivered to any source in the network.

In the work of Dinitz et al. [34] (in which the capacity results in this thesis first appeared), the authors describe and analyze an algorithm that achieves a throughput in \(\Omega\left(\frac{1}{\Delta_{mdst}(G)}\right)\) for delivering packets from a single source to the whole network, called the broadcast capacity problem. Here, the parameter \(\Delta_{mdst}(G)\) denotes the minimum degree of the maximum degree spanning tree (MDST) of \(G\) (for a formal definition of this property, please refer to Section 1.3). To instead solve all-to-all capacity, we could simply run \(n\) instances of this algorithm: one for each source, rotating through the different instances in a round robin fashion. This approach provides a baseline throughput result of \(\Omega\left(\frac{1}{n \cdot \Delta_{mdst}(G)}\right)\).

The key questions we consider here are whether or not this bound is tight, and whether there are simpler or more natural strategies than deploying round robin interleaving of single-source broadcast.

In Section 3.5 we answer both questions in the affirmative. In Section 3.5.1 we begin by defining the all-to-all capacity problem. We then prove in Section 3.5.2 that no schedule achieves a throughput which exceeds \(O\left(\frac{1}{n \cdot \Delta_{mdst}(G)}\right)\). Then, in Section 3.5.3, we describe and analyze a distributed algorithm that achieves this optimal throughput by solving all-to-all gossip with a simple flood on a good spanning tree for each packet. This algorithm then becomes our basis for largely resolving an open question from [85] regarding one-shot gossip in the mobile telephone model.
We then connect this back to the problem of one-shot gossip, using our upper bound on achievable throughput to find a corresponding lower bound for the one-shot gossip problem in the synchronous MTM.

3.3 Synchronous One-Shot Gossip in Peer-to-Peer Networks

In this section, we study the one-shot gossip problem in peer-to-peer networks using the mobile telephone model introduced by Ghaffari and Newport [57]. We begin in Section 3.3.1 by defining the model and the one-shot gossip problem. Then, in Section 3.3.2, we analyze a lower bound for the easier case of the synchronous model. Lastly, we see if we can derive a corresponding upper bound in Section 3.3.3.

3.3.1 Model and Problem Definition

The mobile telephone model (MTM) introduced in [57] describes a peer-to-peer network of wireless devices. Time in the MTM proceeds in synchronous rounds with all nodes starting at round 1. In each round, each node begins by broadcasting an advertisement containing $O(\log n)$ bits to its neighbors in $G$. After receiving advertisements, each node can decide to send a connection proposal to at most one neighbor. Any node that receives one or more proposals must accept exactly one. We allow the model to arbitrarily select which proposal is accepted in this case. (That is, we do not necessarily assume that each node successfully receives all incoming proposals and is therefore able to make a careful decision on which to accept.) When a connection proposal is accepted, the sender and receiver of the proposal are connected and given access to a reliable, bidirectional communication channel for the duration of the round. Note that, by this construction, each node is limited to at most two connections each round.

\footnote{Note that, while the MTM definition can be used to study networks with dynamic graph topologies, in the results that follow we assume that the communication graph is static.}
The one-shot gossip problem we study assumes that \( k \geq 1 \) gossip rumors (called *tokens* in the following) are distributed arbitrarily to nodes at the beginning of the execution (that is, some nodes can start with many tokens, some can start with none). The problem is solved once all nodes know all \( k \) rumors. Critically, we assume that the network topology is connected (i.e., there is a path between every pair of nodes in the network), as otherwise it is possible to spread the tokens in such a way that the problem cannot be solved. Nodes do not know \( k \) in advance. We treat the gossip tokens as comparable black boxes but assume that an individual token can be quite large (e.g., a sizeable file). Although we do not bound the token size, we assume it is big enough such that only way for a node \( u \) to communicate a token to node \( v \) is if \( u \) and \( v \) are connected. In the synchronous setting, we limit nodes to communicating at most a constant number of tokens over a given connection in a single round. (Later, in Section 3.4 where we study this problem in the asynchronous setting, we instead bound the maximum time required to transmit a single token over a connection.)

### 3.3.2 Lower Bound for Synchronous One-Shot Gossip

We begin by proving the first lower bound for the problem of one-shot gossip in the MTM, as it will provide relevant context for discussing many results throughout this section.

**Theorem 3.3.1.** Fix a network size \( n > 0 \), token count \( k > 0 \), and value \( \alpha \in [1/n, 1] \). There exists a graph of size \( n \) with vertex expansion at least \( \alpha \) and an initial assignment of \( k \) tokens to the nodes in this graph, such that any gossip algorithm requires at least \( \Omega(k/\alpha) \) rounds to spread all tokens to all nodes.

**Proof.** Fix a network size \( n \) and token count \( k \) such that \( n, k > 0 \). Let \( q = n\alpha/2 \) for a fixed value \( \alpha \in [1/n, 1] \). Construct our graph \( G \) by creating a \( q \)-clique of nodes and connecting the remaining \( n - q \) nodes to every node in the \( q \)-clique. Note, for example, that this graph is equivalent to a star graph when \( q = 1 \).
First, we prove that this graph has vertex expansion at least \( \alpha \). Recall from Definition 1.3.11, that the vertex expansion of \( G = (V, E) \) is the minimum value of \( |\partial(S)|/|S| \) over all cuts \( S \subset V \) of size at most \( |S| \leq n/2 \). When considering all possible cuts of our graph, we can choose to either include nodes from the \( q \)-clique or nodes not in the clique, or both. If we select any node from the \( q \)-clique, by our construction, every remaining node is now in \( \partial(S) \). Therefore, the only freedom we have to minimize \( |\partial(S)|/|S| \) is to increase the size of \( S \) as to maximize the denominator. Since we can include at most \( n/2 \) nodes in \( S \), the minimum value of \( |\partial(S)|/|S| \) is 1 which is at least \( \alpha \) since we assume \( \alpha \in [1/n, 1] \).

The only remaining option that might violate our claim would be to try to minimize \( |\partial(S)|/|S| \) by not including any nodes from the \( q \)-clique in our set \( S \). As soon as we include a single node outside of the \( q \)-clique, \( |\partial(S)| = q \). Furthermore, since \( q \leq n/2 \) and so there are at least \( n/2 \) nodes not in the \( q \)-clique, we are compelled to include up to \( n/2 \) nodes this way. This is because doing so minimizes the value of \( |\partial(S)|/|S| \) since it has no effect on the numerator while maximizing the denominator. However, even when the quantity is minimized in this way, it is always the case that \( |\partial(S)|/|S| \geq q/(n/2) \). Lastly, note that since \( q = n\alpha/2 \), this quantity is at least \( \alpha \). In both cases, we have now shown that the vertex expansion of \( G \) is at least \( \alpha \), satisfying the first condition of the lemma statement.

We now show, under a particular assignment of \( k \) tokens, that every gossip solution takes \( \Omega(k/\alpha) \) rounds on this graph \( G \) to spread all tokens to all \( n \) nodes of \( G \). This assignment entails giving all \( k \) tokens to every node in the \( q \)-clique. To solve the gossip problem, all \( k \) tokens must be delivered to the \( n - q \geq n/2 \) nodes not in the clique. This requires at least \( kn/2 \) total connections to be made. However, since at most \( q \) connections can occur per round (since nodes outside the clique are not connected and the nodes in the clique are limited to at most one connection per round) a total of at least \( kn/(2q) \) rounds are required. Substituting \( q = n\alpha/2 \) then gives that the number of rounds required to spread all tokens to
all nodes is at least,

\[
\frac{kn}{2q} = \frac{kn}{2(n\alpha/2)} = \Omega(k/\alpha)
\]

Therefore, the theorem is satisfied. □

3.3.3 **Upper Bound for Synchronous One-Shot Gossip**

Here we present and analyze the random diffusion gossip algorithm in order to demonstrate an improved upper bound for one-shot gossip in the MTM.

**Algorithm**

The core strategy of our algorithm, which we formalize in the pseudocode of Algorithm 2, is for nodes to send tokens to their neighbors with the smallest token sets. This contrasts to the strategy of [89] in which nodes bias connection attempts toward neighbors that have not participated in connections in recent rounds.

In each round, \( r \), each node, \( u \), with token set, \( T \), advertises a hash of its entire token set, \( H(T) \), the size its entire token set, \( |T| \), and its unique identifier, \( u \). As in [89], a couple of simplifying assumptions are made here regarding these advertisements. For the sake of clarity, we first assume that hash collisions do not occur in the executions we consider, allowing us to make the reasonable assumption that different token set hashes always indicate different token sets. Given that the execution length is polynomial in the network size \( n \), and there are at most \( n \) different token sets hashed each round, for standard hash function parameters the probability of a collision among this set would be extremely small. \(^3\)

\(^3\)Note that the worst-case outcome of a hash collision in a given round of Algorithm 2 is that a potentially productive connection is ignored. However, as long as less than a constant fraction
Algorithm 2 Random diffusion gossip (for process $u$)

1: $T \leftarrow$ initial token set of $u$
2: $H \leftarrow$ shared hash function
3:
4: for round $r > 0$ do
5: 6: Advertise($\langle H(T), |T|, u \rangle$)
7: $A \leftarrow$ ReceiveAdvertisements()
8:
9: $s \leftarrow \min \{s_v \mid \langle h, s_v, * \rangle \in A, h \neq H(T)\}$
10: $N \leftarrow \{v \mid \langle h, s, v \rangle \in A, h \neq H(T)\}$
11:
12: $v \leftarrow$ node chosen uniformly at random from $N$
13:
14: (send a connection proposal to $v$ if such a node, $v$, exists)
15: (if $v$ accepts, send/receive a token from the set difference with $v$’s token set)

Node $u$ then considers advertisements from neighbors that advertised different token set hashes, identifying the smallest token set size from this set. It randomly selects one of these nodes $v$ to send a connection proposal. If $v$ then receives several connection proposals, by the guarantees of the MTM (reviewed in Section 3.3.1), $v$ will accept exactly one, chosen arbitrarily. Then, if $u$’s proposal to $v$ is accepted, one to two tokens from the set difference are transferred over the connection.\(^4\) In more detail, if $u$ possesses a token that $v$ does not, of these connections are blocked by hash collisions each round, the proof strategy we deploy to prove Theorem 3.3.2 can be used to derive the same asymptotic complexity. Again, note that the probability that even a single hash collision occurs during a particular polynomial-time algorithm execution is extremely small even for conservative hash function parameters.

Furthermore, we emphasize that the algorithm can be easily modified in such a way that, even if many of the potentially productive connections are blocked in a particular round, the impact would be minimal on the overall performance of the algorithm. For example, we could additionally hash the round number $r$ alongside the token set in each round such that any hash collisions in a particular round are likely to be resolved in the following round.

\(^4\)As is common when studying gossip in the MTM, we disregard the time required for two connected nodes to determine which token to transfer. Our algorithm simply specifies that they transfer some token in the set difference of their token sets. This is because, in the application
then \( u \) sends a token to \( v \) and vice versa. Furthermore, if both \( u \) and \( v \) possess tokens that the other does not, then \( u \) and \( v \) send tokens. Note that this means whichever node has the smallest token set is guaranteed to learn at least one new token and grow its token set.

Further note that each node can potentially participate in at most two connection attempts per round: one which it initiated by sending a connection proposal, and one which was initiated by a neighbor’s proposal.

**Complexity Analysis**

We now analyze the time complexity of the random diffusion gossip algorithm.

**Theorem 3.3.2.** With probability at least \( 1 - 2/n \), the random diffusion gossip algorithm solves gossip in \( O((k/\alpha) \log n \log^2 \Delta) \) rounds, where \( k \) is the number of initial tokens, \( \alpha \) is the vertex expansion of the graph, \( n \) is the size of the graph, and \( \Delta \) is the maximum degree of the graph.

Fix a topology graph \( G = (V, E) \) and recall from Section 1.3 that, for node \( u \in V \), we use \( N(u) \) to denote the exclusive neighborhood of \( u \) in \( G \). We now define some additional notation for the analysis that follows. Fix a round \( r > 0 \). Let \( T_u(r) \) be the token set of node \( u \) and let \( s_u(r) \) be the minimum token set size among nodes in \( N(u) \) at the beginning of round \( r \). Furthermore, let \( N_u(r) \subseteq N(u) \) be the *productive* neighbors for \( u \) at the beginning of round \( r \), where we define,

\[
N_u(r) = \{ v \in N(u) \mid s_u(r) = |T_v(r)|, H(T_u(r)) \neq H(T_v(r)) \}. \tag{3.3}
\]

scenarios we envision, the token sizes are large enough to subsume any overhead required to identify which token to transfer (which would depend on the token set sizes, bounded by \( k \)). In the event that the token set sizes are allowed to become massive, however, we can leverage the token transfer subroutine from [85] to decide this set difference using only \( O(\text{polylog} \, k) \) additional bits with at least constant probability.
In other words, the productive neighbor set \( N_u(r) \) denotes the subset of \( u \)'s neighbors from which \( u \) selects a node to send a connection proposal in round \( r \).

For integer sizes \( i \in 0, \ldots, k \); let \( S_i(r) = \{v \in V | i = |T_v(r)|\} \) be the set of nodes that know exactly \( i \) tokens at the beginning of round \( r \). Next, let \( n_i(r) = |S_i(r)| \) and \( n_i^*(r) = \min\{n_i(r), n - n_i(r)\} \). We also define \( i_{\min}(r) = \min\{i \in 0, \ldots, k | n_i(r) > 0\} \) as the minimum token set size for which there is at least one node with exactly that many tokens. For convenience, let \( S_{\min}(r) = S_{i_{\min}}(r) \), \( n_{\min}(r) = n_{i_{\min}}(r) \), and \( n_{\min}^*(r) = n_{i_{\min}}^*(r) \) for \( j = i_{\min}(r) \). Finally, we define \( C(r) = \{i \in 0, \ldots, k | n_i(r) > 0\} \) as the number of unique token set sizes for which at least one node possesses exactly that many tokens.

We continue by defining the productive subgraph, \( G(r) \), of \( G \), defined with respect to a fixed round \( r \), which describes all the connections which nodes might attempt to form in the given round, \( r \).

**Definition 3.3.3.** At the beginning of round \( r > 0 \), define the productive subgraph \( G(r) \) of the graph topology \( G = (V, E) \) as the undirected graph \( G(r) = (V, E(r)) \) such that,

\[
E(r) = \{[u, v] \in E | v \in N_u(r)\}.
\]

(3.4)

For the purposes of our analysis, it will be sufficient to focus on a subgraph of the productive subgraph which only considers nodes in \( S_{\min}(r) \) and their neighbors.

**Definition 3.3.4.** At the beginning of round \( r > 0 \), define the minimum productive subgraph \( G_{\min}(r) \) as the undirected bipartite subgraph \( G_{\min}(r) = (L_{\min}(r), R_{\min}(r), E_{\min}(r)) \) such that:

- \( L_{\min}(r) = \{u \in V \setminus S_{\min}(r) | N(u) \cap S_{\min}(r) \neq \emptyset\} \),

- \( R_{\min}(r) = \{u \in S_{\min}(r) | N(u) \cap (V \setminus S_{\min}(r)) \neq \emptyset\} \), and

- \( E_{\min}(r) = \{[u, v] \in E(r) | u \in L_{\min}(r), v \in R_{\min}(r)\} \).
In other words, the minimum productive subgraph $G_{\text{min}}(r)$ only contains edges representing the potential connections which would result from connection proposals sent to nodes with the fewest number of tokens in the entire network at the beginning of round $r$ (from nodes with more than this number of tokens). The significance of $G_{\text{min}}(r)$ is that every productive connection in this graph causes a node with the fewest number of tokens to no longer have the fewest number of tokens.

The approach we will take when proving Theorem 3.3.2 is to bound how long any minimum token set size $i_{\text{min}}(r)$ can remain the minimum token set size. Since the minimum token set size can never decrease, this will then allow us to prove the total time complexity for our algorithm. For most of our analysis, we will focus on the connections between nodes in $S_{\text{min}}(r)$ and $V \setminus S_{\text{min}}(r)$. In order for this cut to exist though, clearly it must be the case that $C(r) > 1$. Therefore we quickly handle the case where $C(r) = 1$.

**Lemma 3.3.5.** Fix a round $r > 0$ such that $C(r) = 1$ and $i_{\text{min}}(r) < k$. Either:

1. $C(r + 1) > 1$ or
2. $i_{\text{min}}(r + 1) > i_{\text{min}}(r)$.

**Proof.** Fix a round $r > 0$ such that $C(r) = 1$. Note that all nodes $x \in V$ have the same number of tokens, therefore $s_x(r) = |T_y(r)|$ for all edges $\{x, y\} \in E$. Since we assume $G$ is connected (refer to our problem definition in Section 3.3.1), since $i_{\text{min}} < k$ and, therefore, not all nodes have $k$ tokens, there is some node $u$ with a neighbor $v$ such that $H(T_u) \neq H(T_v)$. Furthermore, we have that $s_u(r) = |T_v(r)|$ for all edges $\{u, v\} \in E, v \in N_u(r)$. Therefore, since $|N_u(r)| > 0$, $u$ will send a connection proposal to some neighbor $w \in N_u(r)$.

If $w$ receives a connection proposal from $u$, $w$ is guaranteed to accept at least one connection proposal this round and participate in at least one connection in this round (initiated by a neighbor’s proposal). Therefore, after this round $w$ will possess a new token such that
\(|T_w(r + 1)| > i_{\text{min}}(r)|. There are now two possibilities: for all \(x \in V, |T_x(r + 1)| \geq |T_w(r + 1)|\) or there exists \(x \in V\) such that \(|T_x(r + 1)| < |T_w(r + 1)|\). In the first case, \(i_{\text{min}}(r + 1) > i_{\text{min}}(r)|\) since all nodes possess more than \(i_{\text{min}}(r)|\) tokens, satisfying our second criterion. In the second case \(C(r + 1) > 1\) since there is a node \(x\) with fewer tokens than \(w\), satisfying the first criterion. \(\square\)

The purpose of Lemma 3.3.5 is to simply establish that if \(C(r) = 1\) for some round \(r > 0\), there are some nodes which quickly achieve a token set larger than the minimum number of tokens held by any node. Therefore in many of the lemmas that follow, we fix a round \(r > 0\) and assume the more general case that \(C(r) > 1\), revisiting the case that \(C(r) = 1\) in the proof of Theorem 3.3.2 using Lemma 3.3.5.

This allows us to analyze the cut between these nodes in \(V \setminus S_{\text{min}}(r)|\) and the nodes that still possess exactly \(i_{\text{min}}(r)|\) tokens, \(S_{\text{min}}(r)|\). Productive connections made over this cut will provide nodes of \(S_{\text{min}}(r)|\) with new tokens, increasing their token set size, and shrinking \(S_{\text{min}}(r)|\). When no nodes remain, the minimum token set size must be larger than \(i_{\text{min}}(r)|\).

We continue by lower bounding the number of potential productive connections that are available in \(G_{\text{min}}(r)|\).

**Lemma 3.3.6.** For a fixed round \(r > 0\) such that \(C(r) > 1\), let \(m\) be the size of the maximum matching over \(G_{\text{min}}(r)|\). It follows that, \(m \geq (\alpha/4) \cdot n^\ast_{\text{min}}(r)|\).

**Proof.** Fix a round \(r > 0\) such that \(C(r) > 1\). This implies that there exists a nonempty bipartite subgraph \(B(S_{\text{min}}(r)|) = (S_{\text{min}}(r)|, V \setminus S_{\text{min}}(r)|)\) of \(G|\). Fix \(B(S_{\text{min}}(r)|)\) and recall that \(n^\ast_{\text{min}}(r)|\) is the minimum of the sizes of the two partitions. Let \(M\) be the maximum matching over \((S_{\text{min}}(r)|, V \setminus S_{\text{min}}(r)|)\) where \(|M| = m|\). In the following, we will show that both: \(m \geq (\alpha/4) \cdot n^\ast_{\text{min}}(r)|\) and every edge of \(M\) is in \(E_{\text{min}}(r)|\), where recall that \(\alpha\) is the vertex expansion.
of $G$ (see Definition 1.3.11). Next note that, from Lemma 1.3.13, we have that,

$$
\min_{S \subset V, |S| \in R(n/2)} \frac{\nu(B(S))}{|S|} \geq \alpha/4.
$$

(3.5)

Recall from our notation introduced in Section 1.3 that for node subset $S \subseteq V$ and bipartite subgraph $B(S)$, $\nu(B(S))$ denotes the size of the maximum matching over $B(S)$. First, observe that clearly either $S_{min}(r)$ or $V \setminus S_{min}(r)$ is a subset with size at most $n/2$. Since it follows that $B(S_{min}(r)) = B(V \setminus S_{min}(r))$, without loss of generality we have that,

$$
\nu(B(S_{min}(r))) \geq \alpha/4.
$$

(3.6)

Note that Line 3.8 follows directly from the definition of $n^*_r$. Since, by definition, $m = \nu(B(S_{min}(r)))$, this gives that $m \geq (\alpha/4) \cdot n^*_r(r)$.

We now show that $M \subseteq E_{min}(r)$, noting that this would imply that $M$ is a valid matching over $G_{min}(r)$ as well. Consider an arbitrary edge $\{u, v\} \in M$ and, without loss of generality, assume $u \in V \setminus S_{min}(r)$ and $v \in S_{min}(r)$. By the definition of $S_{min}(r)$, no node has fewer tokens than $v$ and therefore $v$ must have the smallest token set size out of all $u$’s neighbors. Since $v \in N(u)$, this means that $s_u(r) = |T_v(r)|$ which implies that $H(T_u(r)) \neq H(T_v(r))$. This is sufficient to show that $\{u, v\} \in E(r)$.

Furthermore, clearly $N(u) \cap S_{min}(r) \neq \emptyset$ and $N(v) \cap (V \setminus S_{min}(r)) \neq \emptyset$ and therefore $u \in L_{min}(r)$ and $v \in R_{min}(r)$. Therefore, it must be the case that $\{u, v\} \in E_{min}(r)$ as well. Since we can show this for any arbitrary edge $\{u, v\} \in M$, it is true for every such edge in the matching. Therefore $M \subseteq E_{min}(r)$ such that $M$ is also a matching of size at least $m \geq (\alpha/4) \cdot n^*_r(r)$ over $G_{min}(r)$. $\square$
We now have a lower bound for the number of potential connections that nodes in $S_{\text{min}}(r)$ could participate in for a given round. To show that our algorithm is able exploit these possible connections, we now analyze the behavior of the algorithm with respect to the size of the maximum matching over $G_{\text{min}}(r)$ for a fixed round $r$. First, to simplify the proofs that follow, we handle up front the simple edge case of $\Delta = O(1)$.

**Lemma 3.3.7.** For a fixed round $r > 0$, let $m$ be the size of the maximum matching over $G_{\text{min}}(r)$. If $\Delta < \delta$ for any constant $\delta \geq 1$. It follows that the expected number of nodes in $R_{\text{min}}(r)$ that participate in a productive connection in round $r$ is strictly greater than $m/\delta$.

**Proof.** Fix a round $r > 0$ and constant $\delta \geq 1$ and let $m$ be the size of a maximum matching, $M$, over $G_{\text{min}}(r)$. For each edge $\{u, v\} \in M$ where $u \in L_{\text{min}}(r)$ and $v \in R_{\text{min}}(r)$, we now calculate the probability that $u$ sends a connection proposal to $v$ in round $r$. (Note that the probability of this event lower bounds the probability that $v$ receives a connection proposal at all this round.) Given that $\{u, v\} \in E_{\text{min}}(r)$, we know that it signifies a productive connection. By our algorithm, if $d(u) = |N(u)|$ this then means that the probability that $u$ sends $v$ a connection proposal is at least $1/d(u) \geq 1/\Delta > 1/\delta$. Note that this event is sufficient for $v$ to participate in a productive connection this round.

Let $R \subseteq R_{\text{min}}(r)$ denote the subset of nodes which are endpoints of an edge in $M$ (i.e., every node $v \in R_{\text{min}}(r)$ such that $\{u, v\} \in M$ for some node $u \in L_{\text{min}}(r)$). Note that by the above, for each $v \in R$ we have that the probability $v$ participates in a productive connection in round $r$ is greater than $1/\delta$. Since $|R| = m$, it then follows that the expected number of nodes in $R$ that participate in a productive connection this round is more than $m/\delta$. Since $R \subseteq R_{\text{min}}(r)$, the lemma then follows. $\square$

We now move on to the significantly more challenging case of $\Delta = \omega(1)$, which begins with us proving and applying a significantly reworked version of a core lemma from [57]
which bounds the behavior of randomized connection attempts in bipartite graphs satisfying certain properties. In the immediate context of our synchronous analysis, this new version of the lemma provides a log-factor time complexity improvement as compared to the version in [57]. Therefore, this improved version of the lemma can be plugged into the analysis of [57] to provide a log factor improvement to the complexity of its rumor spreading algorithm.

Fix a round \( r > 0 \). For the analysis that follows, let \( G(L, R) \) be the bipartite subgraph of \( G_{\min}(r) \) induced by node subsets \( L \subseteq L_{\min}(r) \) and \( R \subseteq R_{\min}(r) \). For these partitions, let \( N_{L,R}(u) \) be the neighbor set of node \( u \) in \( G(L, R) \) and let \( d_{L,R}(u) = |N_{L,R}| \). Furthermore, for convenience, let \( N_{\min}(u) = N_{L_{\min}(r),R_{\min}(r)}(u) \) be the neighbors of \( u \) in the entire graph \( G_{\min}(r) \) and \( d_{\min}(u) = |N_{\min}(u)| \).

**Lemma 3.3.8** (Replaces Theorem 7.4 in [57]). For a fixed round \( r > 0 \) and \( \Delta > 1 \), let \( L \subseteq L_{\min}(r) \) and \( R \subseteq R_{\min}(r) \) be node subsets such that:

1. there is a matching over \( G(L, R) \) of size \( |L| \) and
2. for every \( u \in L \), every neighbor of \( u \) in \( R_{\min}(r) \) is in \( R \).

It then follows that either:

1. The expected number of nodes in \( R_{\min}(r) \) that participate in a productive connection in round \( r \) is at least \( \frac{|L|}{12 \log \Delta} \), or
2. we can identify node subsets \( L'' \subseteq L \cap L_{\min}(r') \) and \( R'' \subseteq R \cap R_{\min}(r') \) for some \( r' \in \{r, r + 1\} \) such that:

   (a) there is a matching over \( G(L'', R'') \) of size \( |L''| \),
   
   (b) for every \( u \in L'' \), every neighbor of \( u \) in \( R_{\min}(r') \) is in \( R'' \),
   
   (c) \( |L''| \geq \left(1 - \frac{13}{12 \log \Delta}\right) \cdot |L| \), and
Proof. Our proof is broken up into several steps. Fix a round \( r > 0 \) and subsets \( L \subseteq L_{\min}(r) \) and \( R \subseteq R_{\min}(r) \) as defined by the lemma statement. Let \( m = |L| \) be the size of the maximum matching over \( G(L, R) \) by Assumption 1 of the lemma statement and let \( M \) be this matching. We denote a node \( v \in R \) as the original match of a node \( u \in L \) if \( \{u, v\} \in M \). This terminology was first introduced in the proof from [57].

We begin by removing the high-degree nodes from \( L \). Let \( D = \sum_{u \in L} d_{L,R}(u) \) be the degree sum over \( G(L, R) \) of all nodes in \( L \) and let \( \delta = (D/m) \log \Delta \). By definition of \( m \) and \( D \), this choice of \( \delta \) ensures that at most a \( 1/\log \Delta \) fraction of the nodes \( u \in L \) can have \( d_{L,R}(u) > \delta \). Let \( L' \subseteq L \) be the subset of nodes once we remove all nodes with degree greater than \( \delta \) from \( L \) and note it follows that \( |L'| \geq \left(1 - \frac{1}{\log \Delta}\right) \cdot |L| \geq \left(1 - \frac{13}{12 \log \Delta}\right) \cdot |L| \).

We then remove all nodes from \( R \) that are not connected to \( L' \) and denote the remaining set \( R' \). Note that for every node \( u \in L' \) we have that \( N_{L,R}(u) = N_{L',R'}(u) \). We note (as first pointed out in [57]) that this implies \( G(L', R') \) has a matching of size \( |L'| \) since for every node \( u \in L' \), \( u \)'s original match is in \( R' \). These observations alone fulfill Conditions 2a, 2b, and 2c of the lemma. Therefore, if \( \sum_{u \in L'} d_{L',R'}(u) \leq \frac{5}{6}D \) such that Condition 2d is met, then Objective 2 of the lemma is already satisfied by setting \( L'' = L', R'' = R', \) and \( r' = r \). We therefore assume for the remainder of the proof that,

\[
\sum_{u \in L'} d_{L', R'}(u) > \frac{5}{6}D.
\]  

At this point, we diverge significantly from the proof strategy of [57] and introduce a new technique for leveraging the assumption on Line 3.9 regarding the degree sum in \( G(L', R') \).

First, recall Definition 1.3.17 regarding good and bad nodes and edges first introduced by the work of [1]. In \( G(L', R') \) let \( R'_b \subseteq R' \) be the bad nodes in \( R' \) and let \( R'_g \subseteq R' \) be the good nodes, where good and bad are defined with respect to \( G(L', R') \). Since every edge
\(G(L', R')\) has an endpoint in \(R'\), it follows from our assumption on Line 3.9 that either 
\[\sum_{u \in R'_g} d_{L', R'}(u) \geq \frac{1}{6} D \text{ or } \sum_{u \in R'_b} d_{L', R'}(u) \geq \frac{2}{3} D.\]

Simply speaking, since \(R'_b \cup R'_g = R'\), there are at least \(\frac{5}{6} D\) edges which are incident on either a node in \(R'_b\) or a node in \(R'_g\). Therefore, at least \(\frac{1}{6} D\) of these edges in \(G(L', R')\) are incident on \(R'_b\) or at least \(\frac{4}{3} D = \frac{2}{3} D\) of these edges are incident on \(R'_g\).

We first assume the former case. In this case, \(P_u \in R'_b d_{L', R'}(u) \geq \frac{1}{6} D\) or \(P_u \in R'_g d_{L', R'}(u) \geq \frac{2}{3} D\). Simply speaking, since \(R'_b \cup R'_g = R'\), there are at least \(\frac{5}{6} D\) edges which are incident on either a node in \(R'_b\) or a node in \(R'_g\). Therefore, at least \(\frac{1}{6} D\) of these edges in \(G(L', R')\) are incident on \(R'_b\) or at least \(\frac{4}{3} D = \frac{2}{3} D\) of these edges are incident on \(R'_g\).

Divide the nodes of \(R'_b\) into \([\delta]\) classes based on their degree in \(G(L'_b, R'_b)\) such that, for all \(i \in R([\delta])\), the \(i\)th class contains nodes \(v \in R'_b\) with degree \(d_{L'_b, R'_b}(v) = i\). Let \(R'_b(i)\) denote the \(i\)th class and let \(E_i\) be the edges incident on nodes of class \(R'_b(i)\). Note that by our case assumption,

\[
\sum_{i=1}^{[\delta]} |E_i| \geq \frac{1}{6} D. \tag{3.10}
\]

Now note that each node \(u \in L_{\min}(r)\) sends a proposal to a particular node \(v \in N_{\min}(u)\) with probability at least \(1/d_{\min}(u)\). Therefore, the probability that a node \(v \in R_{\min}(r)\) participates in a productive connection in round \(r\) is at least,

\[
1 - \prod_{u \in N_{\min}(v)} \left(1 - \frac{1}{d_{\min}(u)}\right). \tag{3.11}
\]

\[\text{Further note that, by our construction of } G(L'_b, R'_b), d_{L'_b, R'_b}(v) > 0 \text{ for all } v \in R'_b. \text{ Therefore every node } v \in R'_b \text{ is contained in a class } R'_b(i) \text{ for some } i \in R([\delta]).\]
Further note that, by our construction of $G(L', R')$ and Assumption 2 of the lemma statement, for all $u \in L'$ it follows that $N_{L', R'}(u) = N_{min}(u)$ (such that $d_{L', R'}(u) = d_{min}(u)$. Therefore, the probability that a node $v \in R'_b$ participates in a productive connection in round $r$ is at least,

$$1 - \prod_{u \in N_{min}(v)} \left( 1 - \frac{1}{d_{min}(u)} \right) = 1 - \prod_{u \in N_{min}(v)} \left( 1 - \frac{1}{d_{L', R'}(u)} \right) \geq 1 - \prod_{u \in N_{L', R'}(v)} \left( 1 - \frac{1}{d_{L', R'}(u)} \right).$$

(3.12)

(3.13)

Since for every node in $u \in L'_b \subseteq L'$, we have that $d_{L', R'}(u) \leq \delta$, the probability that $v \in R'_b(i)$ participates in a productive connection in round $r$ is at least,

$$1 - \prod_{u \in N_{L', R'}(v)} \left( 1 - \frac{1}{d_{L', R'}(u)} \right) \geq 1 - \prod_{u \in N_{L', R'}(v)} \left( 1 - \frac{1}{\delta} \right) = 1 - (1 - 1/\delta)^{d_{L', R'}(v)} \geq 1 - (1 - 1/\delta)^i. \quad (3.14)$$

(3.15)

(3.16)

Next, from the facts that $i \geq 1$ for $i \in R(\lceil \delta \rceil)$ and $\delta \geq 1$, and Lemma 1.3.7 (letting $x = -1/\delta$) we have that,

$$e^{-1/\delta} \geq 1 - 1/\delta \quad (3.17)$$

$$\Rightarrow \left( e^{-1/\delta} \right)^i \geq (1 - 1/\delta)^i \quad (3.18)$$

$$\Rightarrow 1 - (1 - 1/\delta)^i \geq 1 - \left( e^{-1/\delta} \right)^i \quad (3.19)$$

$$= 1 - e^{-1/\delta}. \quad (3.20)$$
Reapplying Lemma 1.3.7, this time for $x = i/\delta$, then gives,

$$e^{i/\delta} \geq 1 + i/\delta$$  \hspace{1cm} (3.21)

$$\Rightarrow \frac{1}{1 + i/\delta} \geq e^{-i/\delta}$$  \hspace{1cm} (3.22)

$$\Rightarrow 1 - e^{-i/\delta} \geq 1 - \frac{1}{1 + i/\delta}$$  \hspace{1cm} (3.23)

$$= \frac{i/\delta}{1 + i/\delta}$$  \hspace{1cm} (3.24)

$$\geq \frac{i}{2\delta},$$  \hspace{1cm} (3.25)

where the last line follows since $i \leq \delta$ and therefore $1 + i/\delta \leq 2$. Therefore, the probability that $v \in R'_b(i)$ participates in a productive connection in round $r$ is at least $i/(2\delta)$. Now, since there are $|E_i|$ edges incident on nodes in $R'_b(i)$ and each node in $R'_b(i)$ has degree $i$, there are $|E_i|/i$ nodes in this class. Therefore, the expected number of nodes from $R'_b(i)$ that participate in a productive connection in round $r$ is at least,

$$\frac{|E_i|}{i} \cdot \frac{i}{2\delta} = \frac{|E_i|}{2\delta}.$$  \hspace{1cm} (3.26)

We can then derive the the expected number of nodes which participate in a productive connection across all $\lfloor \delta \rfloor$ classes using the inequality on Line 3.10 regarding the degree sum,

$$\sum_{i=1}^{\lfloor \delta \rfloor} \frac{|E_i|}{2\delta} = \frac{1}{2\delta} \sum_{i=1}^{\lfloor \delta \rfloor} |E_i|$$  \hspace{1cm} (3.27)

$$\geq \frac{D/6}{2\delta}$$  \hspace{1cm} (3.28)

$$= \frac{D}{12\delta}.$$  \hspace{1cm} (3.29)
Substituting $\delta = (D/m) \log \Delta$, we can simplify further,

$$\frac{D}{12\delta} = \frac{D}{12 \cdot (D/m) \log \Delta}$$  \hspace{1cm} (3.30)

$$= \frac{m}{12 \log \Delta}. \hspace{1cm} (3.31)$$

Therefore, the first objective of the lemma is satisfied in this case.

We now consider the second case, in which $\sum_{u \in R'_g} d_{L',R'}(u) \geq \frac{2}{3} D$. For $v \in R'_g$, let $N_{L',R'}^f(v) = \{u \in N_{L',R'}(v) \mid d_{L',R'}(u) \leq d_{L',R'}(v)\}$ denote $v$’s lower degree neighbors in $G(L',R')$. First, note that, in the same way that we derived the inequality on Line 3.13 from Line 3.11, we have that the probability that a node $v \in R'_g$ participates in a productive connection this round is at least,

$$1 - \prod_{u \in N_{\text{min}}(v)} \left(1 - \frac{1}{d_{\text{min}}(u)}\right) = 1 - \prod_{u \in N_{\text{min}}(v)} \left(1 - \frac{1}{d_{L',R'}(u)}\right)$$  \hspace{1cm} (3.32)

$$\geq 1 - \prod_{u \in N_{L',R'}^f(v)} \left(1 - \frac{1}{d_{L',R'}(u)}\right). \hspace{1cm} (3.33)$$

Simplifying further, the probability that $v \in R'_g$ participates in a productive connection in round $r$ is at least,

$$1 - \prod_{u \in N_{L',R'}(v)} \left(1 - \frac{1}{d_{L',R'}(u)}\right) \geq 1 - \prod_{u \in N_{L',R'}^f(v)} \left(1 - \frac{1}{d_{L',R'}(v)}\right)$$  \hspace{1cm} (3.34)

$$\geq 1 - (1 - 1/d_{L',R'}(v))^{\lvert N_{L',R'}^f(v) \rvert} \hspace{1cm} (3.35)$$

$$\geq 1 - (1 - 1/d_{L',R'}(v))^{d_{L',R'}(v)/3} \hspace{1cm} (3.36)$$

$$\geq 1 - e^{-1/3} \hspace{1cm} (3.37)$$

$$> 1/4. \hspace{1cm} (3.38)$$
Note that for Line 3.34 we use our definition of $N_{L',R'}^\ell(v)$ to replace the degree of $u$ with that of $v$ and Line 3.36 is where we leverage the fact that $v$ is good.

Now remove every node from $R'$ that is selected in round $r$, denote the remaining set $R''$, and remove from $L'$ every node $u$ for which $u$’s original match was removed from $R'$. Denote the remaining nodes $L''$. Since we know from the above that each node $v \in R'_g$ is removed from $R'$ with probability at least $1/4$ and the probability that any edge $\{u,v\}$ is removed from $G(L',R')$ is at least the probability that $v$ is removed from $R'$, the probability that an edge $\{u,v\}$ incident on $R'_g$ is removed is at least $1/4$. Since by our case assumption that there are at least $\frac{2}{3}D$ edges incident on nodes in $R'_g$, in expectation, at least $\frac{1}{4} \cdot \frac{2}{3}D = \frac{1}{6}D$ edges are removed from $G(L',R')$. Therefore, the expected number of edges $X$ remaining in $G(L'',R'')$ is at most,

$$E[X] \leq D - \frac{1}{6}D$$

(3.39)

$$= \frac{5}{6}D.$$  

(3.40)

Since this satisfies Condition 2d of Objective 2 of the lemma, we conclude by showing that either the remaining conditions of this objective are satisfied or Objective 1 is satisfied.

If $|L''| < \left(1 - \frac{1}{\log \Delta}\right) \cdot m - \frac{m}{12 \log \Delta}$ then note that this means $|L''| \leq |L'| - \frac{m}{12 \log \Delta}$ since $|L'| \geq \left(1 - \frac{1}{\log \Delta}\right) \cdot m$. Note that this implies that at least $\frac{m}{12 \log \Delta}$ nodes in $L'$ had their original match removed in round $r$ which means that at least $\frac{m}{12 \log \Delta}$ nodes of $R'$ participated in a productive connection in this round. Since this satisfies Objective 1 of the lemma, assume
instead that,

\[ |L''| \geq \left(1 - \frac{1}{\log \Delta}\right) \cdot m - \frac{m}{12 \log \Delta} \]  
\[ = \left(1 - \frac{13}{12 \log \Delta}\right) \cdot m \]  
\[ = \left(1 - \frac{13}{12 \log \Delta}\right) \cdot |L|. \]

Furthermore, by our construction, for every node \( u \in L'' \), every neighbor of \( u \) in \( R_{\min}(r + 1) \) is in \( R'' \). This includes \( u \)'s original match in \( R' \) such that there is a matching over \( G(L'', R'') \) of size \( |L''| \). Note that this also implies \( |R''| \geq |L''| \). Furthermore by our construction of \( G(L'', R'') \), \( L'' \subseteq L \cap L_{\min}(r + 1) \) and \( R'' \subseteq R \cap R_{\min}(r + 1) \). Therefore, the second objective is satisfied for node subsets \( L'' \) and \( R'' \) when \( r' = r + 1 \).

\[ \square \]

For the sake of the analysis that follows, we now consider a "phase" of \([4 \log \Delta]\) consecutive rounds.

**Lemma 3.3.9.** Consider a \( T \)-round phase of the algorithm where \( T = [4 \log \Delta] \) beginning at round \( r > 0 \) and let \( m \) be the size of the maximum matching over \( G_{\min}(r) \). It follows that one of the following holds; either:

1. \( i_{\min}(r + T) > i_{\min}(r) \), or

2. there exists a round \( r' \) such that \( r \leq r' < r + T \) and the expected number of nodes from \( S_{\min}(r) \) that participate in a productive connection in round \( r' \) is at least,

\[ \min\left\{ \frac{m}{2^{22}}, \frac{m}{2^{13} \log \Delta} \right\} = \frac{m}{2^{13} \max\{2^9, \log \Delta\}}. \]
Proof. Fix round $r > 0$ and assume that there exists some round $r + i$ where $i \in R(T)$ such that $i_{\text{min}}(r + i) > i_{\text{min}}(r + i - 1)$. In other words, there is a round where the minimum token set size held by all nodes increases in round $r + i - 1$. If this is the case, it then follows that $i_{\text{min}}(r + T) > i_{\text{min}}(r)$ since we have $i_{\text{min}}(r + T) \geq i_{\text{min}}(r + i) > i_{\text{min}}(r + i - 1) \geq i_{\text{min}}(r)$. Therefore, clearly Objective 1 of the lemma is satisfied.

We now assume for the rest of the proof that, for all $i \in R(T)$, $i_{\text{min}}(r + i) = i_{\text{min}}(r + i - 1)$, implying that $i_{\text{min}}(r + i) = i_{\text{min}}(r)$. Note that this implies, for all $i \in R(T)$,

\[ S_{\text{min}}(r + i) \subseteq S_{\text{min}}(r) \quad (3.45) \]

as nodes with the fewest number of tokens continue to have the fewest number of tokens at least until the minimum token set size held across nodes increases (which we assumed it does not during this phase). Consider a $T$-round execution of the algorithm beginning at round $r$. We now describe a process for inductively applying Lemma 3.3.8 $T$ times over this phase. For iteration $i \in R(T)$ of Lemma 3.3.8, let round $r_i$ and node subsets $L_i \subseteq L_{\text{min}}(r_i)$ and $R_i \subseteq R_{\text{min}}(r_i)$ be the choices of $r$, $L$, and $R$, respectively, specified by the lemma. Furthermore, for all $i \in R(T)$, let $m_i = |L_i|$ and $D_i = \sum_{u \in L_i} d_{L_i,R_i}(u)$ be the size of the maximum matching over $G(L_i, R_i)$ and the degree sum over this graph of all nodes in $u \in L_i$.

To begin, let $r_1 = r$ and pick subsets $L_1 \subseteq L_{\text{min}}(r)$ and $R_1 \subseteq R_{\text{min}}(r)$ satisfying the criteria of Lemma 3.3.8 such that $|L_1| = m$ where $m$ is the size of the maximum matching over $G_{\text{min}}(r)$. Assuming that, in iteration $i$ of Lemma 3.3.8, it is Objective 2 that is satisfied, let $r_i', L_i''$, and $R_i''$ be the values of $r'$, $L''$, and $R''$ as specified by this objective of the lemma. For our induction, in each iteration $i \in R(T)$ such that Objective 2 is satisfied, we define $r_i' = r_{i+1}$, $L_i'' = L_{i+1}$ and $R_i'' = R_{i+1}$. Note that if there is an iteration $i$ for which instead Objective 1 of Lemma 3.3.8 is met, we cease the induction immediately and the values $r_j'$, $L_j''$, and $R_j'$ for $i \leq j \leq T$ are not defined. Now, if $\Delta < 8$ then setting $\delta = 8$ in Lemma
3.3.7 gives us that the expected number of nodes in $R_{\text{min}}(r) \subseteq S_{\text{min}}(r)$ that participate in a productive connection in the first round $r$ is more than $m/8$. As $m/8 \geq m/2^{22}$, this satisfies Objective 2 of our lemma. Therefore, going forward assume that

$$\Delta \geq 8. \quad (3.46)$$

We divide the proof that follows into two cases: either there exists an iteration $i \in R(T)$ of Lemma 3.3.8 in this phase where Objective 1 is satisfied, or in every such iteration Objective 2 is satisfied. We show that, in both cases, the expected number of nodes of $R_{\text{min}}(r)$ that participate in a productive connection this phase is in $\Omega\left(\frac{m}{\log \Delta}\right)$. Consider the first case and note that, by definition, this means that there exists an iteration $i \in R(T)$ of Lemma 3.3.8 for which at least $\frac{m}{12 \log \Delta}$ expected nodes of $R_{\text{min}}(r_i)$ participate in a productive connection. Therefore, if we can ensure that $m_i \geq \frac{12}{22}m$, Objective 2 of our lemma is satisfied in this case.

Fix iteration $i \in R(T)$, such that $i$ is the first iteration of the lemma in this $T$-round execution such that Objective 1 of Lemma 3.3.8 is satisfied. Since it then follows that Objective 2 of Lemma 3.3.8 is satisfied in every iteration $i' < i$, we have that, for all $i' \in R(i - 1)$, $|L_{i'}''| \geq \left(1 - \frac{13}{12 \log \Delta}\right) \cdot |L_{i'}|$. By our induction, this implies that for every such iteration $i'$,

$$|L_{i'}''| \geq \left(1 - \frac{13}{12 \log \Delta}\right) \cdot |L_{i'}| \quad (3.47)$$

$$\Rightarrow |L_{i'+1}| \geq \left(1 - \frac{13}{12 \log \Delta}\right) \cdot |L_{i'}| \quad (3.48)$$

$$\Rightarrow m_{i'+1} \geq \left(1 - \frac{13}{12 \log \Delta}\right) \cdot m_i. \quad (3.49)$$
Recursively applying the inequality on Line 3.49 over the first $i - 1$ iterations of the induction yields that,

$$m_i \geq \left(1 - \frac{13}{12 \log \Delta}\right) \cdot m_{i-1}$$  \hspace{1cm} (3.50)

$$\geq \left(1 - \frac{13}{12 \log \Delta}\right)^{i-1} \cdot m_{i-1}$$  \hspace{1cm} (3.51)

$$= \left(1 - \frac{13}{12 \log \Delta}\right)^{i-1} m_{i}$$  \hspace{1cm} (3.52)

$$= \left(1 - \frac{13}{12 \log \Delta}\right)^{i-1} m.$$  \hspace{1cm} (3.53)

Since, by Line 3.46, we have that $\Delta \geq 8$ which implies that $\frac{13}{12 \log \Delta} \in (0, 1/2]$, we can invoke Lemma 1.3.8 for $x = \frac{13}{12 \log \Delta}$. Furthermore, since $i \leq T = \lceil 4 \log \Delta \rceil < 4 \log \Delta + 1$, it follows that for all $i \in R(T)$,

$$\left(1 - \frac{13}{12 \log \Delta}\right)^{i-1} m \geq \left(1 - \frac{13}{12 \log \Delta}\right)^{T-1} m$$  \hspace{1cm} (3.54)

$$\geq \left(1 - \frac{13}{12 \log \Delta}\right)^{4 \log \Delta + 1 - 1} m$$  \hspace{1cm} (3.55)

$$= \left(1 - \frac{13}{12 \log \Delta}\right)^{4 \log \Delta} m$$  \hspace{1cm} (3.56)

$$\geq \left(4^{-\frac{13}{12 \log \Delta}}\right)^{4 \log \Delta} m$$  \hspace{1cm} (3.57)

$$= 4^{-13/3} m$$  \hspace{1cm} (3.58)

$$= 2^{-26/3} m.$$  \hspace{1cm} (3.59)

Therefore, in the first case, where there is an iteration $i \in R(T)$ where the first objective of Lemma 3.3.8 is satisfied, we have that the expected number of nodes in $R_i \subseteq R_{\min}(r_i) \subseteq S_{\min}(r_i) \subseteq S_{\min}(r)$ (Line 3.45) that participate in a productive connection in round $r_i$ is at
least,

\[
\frac{m_i}{12 \log \Delta} \geq \frac{2^{-26/3} m}{12 \log \Delta} \geq \frac{m}{2^{13} \log \Delta}.
\] (3.60)

Therefore, Objective 2 of the lemma is satisfied in this case.

Now consider the second case such that, in each application of Lemma 3.3.8, it is Objective 2 of the lemma that is satisfied. For the remainder of this proof, let the notation \( E_i[D_i] \) denote the expectation of the degree sum \( D_i \) given the execution of the algorithm up until the beginning of round \( i \) (i.e., in all rounds before round \( i \)). By our induction and Condition 2d of Lemma 3.3.8, we then have that, for all \( i \in R(T - 1) \), \( E_{i+1}[D_{i+1}] \leq \frac{5}{6} D_i \). Therefore, applying this inequality recursively for \( T - 1 \) total steps yields that,

\[
E_T[D_T] \leq (5/6)D_{T-1} \]

(3.62)

\[
\Rightarrow E_{T-1}[E_T[D_T]] \leq E_{T-1}[(5/6)D_{T-1}] \]

(3.63)

\[
\Rightarrow E_T[D_T] = (5/6)E_{T-1}[D_{T-1}] \leq (5/6)(5/6)D_{T-2} \]

(3.64)

\[
\ldots \]

\[
E_T[D_T] \leq (5/6)^{T-1} D_1 \leq (5/6)^{T-1} m \Delta. \]

(3.65)
where the last line follows since $L_1$ contains $m$ nodes, each with degree at most $\Delta$. Since $T = \lceil 4 \log \Delta \rceil \geq 4 \log \Delta$ we can simplify this expression further,

$$\left(\frac{5}{6}\right)^{T-1} m \Delta = \frac{6}{5} \cdot \left(\frac{5}{6}\right)^T m \Delta$$  \hfill (3.69)

$$\leq \frac{6}{5} \cdot \left(\frac{5}{6}\right)^{4 \log \Delta} m \Delta$$  \hfill (3.70)

$$= \frac{6}{5} \cdot \Delta^{4(\log 5 - \log 6)} m \Delta$$  \hfill (3.71)

$$< \frac{6}{5} m.$$  \hfill (3.72)

All together, this means that the expected degree sum of $L_T$ is at most $\frac{6}{5} m$. By Markov’s inequality (see Theorem 1.3.1) for $a = \frac{11}{5} m$, we then get that the probability that $D_T < \frac{11}{5} m$ is at least $5/11$.

Assume $D_T < \frac{11}{5} m$. Let $\ell = 2^{-26/3}$ and recall that we proved earlier (Line 3.59) that for all $i \in R(T)$ we have that $m_i \geq 2^{-26/3} m$, such that we know $|L_T| = m_T \geq 2^{-26/3} m = \ell m$. Since we assume $D_T < \frac{11}{5} m$, it therefore follows that at most $\ell m/2$ nodes in $u \in L_T$ can have degree $d_{L_T,R_T}(u) > \frac{22}{5\ell}$\footnote{Note that since we assume Objective 2 of Lemma 3.3.8 was satisfied in iteration $T - 1$, by Condition 2b of the lemma we have that for every node $u \in L_T$, every neighbor of $u$ in $R_{\text{min}}(r_T)$ is in $R_T$.}. Therefore, at least $\ell m/2$ nodes in $L_T$ have degree at most $\frac{22}{5\ell}$. Let $M_T$ be the maximum matching over $G(L_T, R_T)$. For each $\{u,v\} \in M_T$, refer to $v$ as $u$’s "match" with respect to $M_T$ and vice versa. Each node $u \in L_T$ which is an endpoint of an edge in $M_T$ with $d_{L_T,R_T}(u) \leq \frac{22}{5\ell}$ sends a connection proposal to its match, $v$, with respect to $M_T$ with probability at least $5\ell/22$. Therefore, we have that at least $(\ell m/2) \cdot (5\ell/22) = \frac{5}{44} \ell^2 m$ expected nodes in $R_T$ receive a connection proposal from their match in round $r_T$. Since this event is sufficient for each node to participate in a productive connection, the expected number of nodes in $R_T \subseteq S_{\text{min}}(r)$ who do so is also at least $\frac{5}{44} \ell^2 m$. 

99
Since this event occurs as long as \( D_T < \frac{11}{5} m \), which we showed to occur with probability at least \( \frac{5}{11} \), this means that this expectation in this case is at least \( \frac{5}{44} \ell^2 m = (5/11) \cdot \frac{5}{44} 2^{-52/3} m \geq 2^{-22} m \). Note that this satisfies Objective 2 of the lemma and, therefore, we are done. \( \square \)

For a given round \( r \), Lemma 3.3.9 gives us a lower bound on the expected number of nodes from \( R_{\min}(r) \subseteq S_{\min}(r) \) which participate in a productive connection over a phase of \( [4 \log \Delta] \) rounds. However, to apply our stochastic dominance argument from Lemma 1.3.5, we must show this expectation is achieved (within a constant factor) with at least constant probability. In addition, we also combine the guarantees of Lemma 3.3.9 with the result proved in Lemma 3.3.6 about the size of the matching over \( G_{\min}(r) \).

**Lemma 3.3.10.** Fix a round \( r > 0 \) such that \( C(r) > 1 \). By round \( r + [4 \log \Delta] \) rounds, at least

\[
\frac{2^{-16} \alpha \cdot n^*_\min(r)}{\max \{2^9, \log \Delta\}} = \Omega((\alpha / \log \Delta) \cdot n^*_\min(r))
\]  

(3.73)

nodes of \( S_{\min}(r) \) participate in a productive connection with probability at least \( 2^{-26} \).

**Proof.** Fix a phase of \( [4 \log \Delta] \) rounds beginning with a fixed round \( r > 0 \) such that \( C(r) > 1 \) and let \( m \) be the size of the maximum matching over \( G_{\min}(r) \), noting that \( C(r) > 1 \) implies that \( m \geq 1 \). Our first goal will be to show that, with at least constant probability, at least,

\[
\frac{2^{-14} m}{\max\{2^9, \log \Delta\}}
\]  

(3.74)

nodes in \( S_{\min}(r) \) participate in a productive connection before round \( r + [4 \log \Delta] \).\(^7\) We then apply our lower bound on \( m \) from Lemma 3.3.6 to this result.

\(^7\)Note that the value on Line 3.74 is equal to half the expectation given by Lemma 3.3.9.
By Lemma 3.3.9, we have that either the size of the minimum token set size increase this phase, or there is a round \( r' \) this phase where the expected number of nodes in \( S_{\text{min}}(r) \) which participate in a productive connection in round \( r' \) is at least \( \frac{2^{-13}m}{\max\{2^3, \log \Delta\}} \). In the former case, clearly every node in \( S_{\text{min}}(r) \) must have participated in a productive connection this phase, as every node with the fewest number of tokens must have learned a new token.

Put plainly, this means that at least \( |S_{\text{min}}(r)| \) nodes of \( S_{\text{min}}(r) \) participated in a productive connection. Since of course \( |S_{\text{min}}(r)| \geq m > \frac{2^{-14}m}{\max\{2^3, \log \Delta\}} \), the claim on Line 3.74 is satisfied in this case. Therefore, we assume the latter case going forward, showing that in this case too, at least \( \Omega(m/ \log \Delta) \) nodes of \( S_{\text{min}}(r) \) participate in a connection this phase with at least constant probability.

Note that the number of nodes in \( S_{\text{min}}(r) \) which participate in a productive connection in a particular round can be described as the sum of negatively associated random variables. To see why this is true, observe the fact that if any node \( v \in S_{\text{min}}(r) \) receives a connection proposal in round \( r' \) from a node \( u \), this means \( u \) cannot send a proposal to any other node this round. This can only hurt the probability that any other node receives and accepts a proposal this round from \( u \). Alternatively if \( v \) initiated the connection with \( u \), this still can only hurt the chance that another node’s proposal is accepted by \( u \). Since, by the guarantees of the MTM, accepted connection proposals always result in a productive connection, the negative association then follows.

Let \( X \) be the random variable for the number of nodes in \( S_{\text{min}}(r) \) that participate in a productive connection in round \( r' \). Given the negative association of these events, we can apply the Chernoff bound from Theorem 1.3.4 for \( \mu = \frac{2^{-13}m}{\max\{2^3, \log \Delta\}} \) and \( \epsilon = 1/2 \) where \( \mathbb{E}[X] \geq \mu \). This gives that the probability fewer than \( \mu/2 \) nodes receive a connection proposal in this
phase is at most,

$$\Pr[X \leq \mu/2] \leq \exp(-\mu/8)$$

(3.75)

$$= \exp\left(-\frac{1}{8} \frac{2^{-13}m}{\max\{2^9, \log \Delta\}}\right)$$

(3.76)

$$= \exp\left(-\frac{2^{-16}m}{\max\{2^9, \log \Delta\}}\right).$$

(3.77)

Since $\max\{2^9, \log \Delta\} \geq 2^9$ and $C(r) > 1$ implies $m \geq 1$, we have that the probability that at least $\frac{2^{-14}m}{\max\{2^9, \log \Delta\}}$ nodes in $S_{\text{min}}(r)$ participate in a productive connection this phase is at least,

$$1 - \exp\left(-\frac{2^{-16}m}{\max\{2^9, \log \Delta\}}\right) \geq 1 - \exp\left(-\frac{2^{-16}m}{2^9}\right)$$

(3.78)

$$= 1 - \exp\left(-2^{-25}m\right)$$

(3.79)

$$\geq 1 - \exp\left(-2^{-25}\right)$$

(3.80)

$$> 2^{-26}.$$  

(3.81)

Therefore, at least $\frac{2^{-14}m}{\max\{2^9, \log \Delta\}}$ nodes in $S_{\text{min}}(r)$ participate in a productive connection this phase with probability at least $2^{-26}$. Finally, from Lemma 3.3.6 we know that for any round $r$ where $C(r) > 1$ there is a matching over $G_{\text{min}}(r)$ of size $(\alpha/4) \cdot n^*_{\text{min}}(r)$.

Substituting this value for $m$ into $\frac{2^{-14}m}{\max\{2^9, \log \Delta\}}$ then satisfies the lemma. \(\square\)

We have now bounded the probability that a particular phase of $O(\log \Delta)$ rounds achieves a sufficient number of productive connections. Our next goal will be to bound the number of rounds required to increase the minimum token set size in the entire network. To establish this lemma we first leverage Lemma 3.3.11 which only bounds the time required for at least half the nodes to have more than the minimum token set size. This requires us to analyze several phases of $T$ rounds each, applying the stochastic dominance argument from Lemma 1.3.5 given the guarantees of Lemma 3.3.10.
Lemma 3.3.11. Fix a round $r > 0$ such that $C(r) > 1$ and $n_{min}(r) \geq n/2$. With probability at least $1 - 1/n^2$, more than $n/2$ nodes possess more than $i_{min}(r)$ tokens by round,

$$r + \left[2^{47}(1/\alpha) \log n \log \Delta \cdot \max[2^9, \log \Delta]\right] + 1 = r + O((1/\alpha) \log n \log^2 \Delta).$$

(3.82)

Proof. In the following, let $\psi = \frac{2^{-16}\alpha}{\max[2^9, \log \Delta]}$. For phase $p'$ of $[4 \log \Delta]$ rounds beginning with round $r'$, we define $p'$ to be successful if at least $\psi \cdot n_{min}^*(r')$ nodes of $S_{min}(r')$ participate in a productive connection during this phase. Let $T > 0$ represent a number of successful phases of $[4 \log \Delta]$ rounds each that we will fix later in the proof. For this sequence of successful phases $p_1, \ldots, p_T$ let $p_i \in R(T)$ denote the $i$th phase of this sequence where $r_i$ is the first round of phase $p_i$. Fix a round $r > 0$ as specified by the lemma statement. For convenience of notation, let $r = r_1$ and define the additional round $r_{T+1}$ to be the round immediately after the last round of phase $p_T$. Our first goal will be to show that, for sufficiently large $T$, by the end of the $T$th successful phase, the number of nodes which possess more than $i_{min}(r_1)$ tokens is more than $n/2$. Note that the sequence $p_1, \ldots, p_T$ only considers successful phases, such that these phases may not appear consecutively in an execution of the algorithm. Therefore, we will further require $T'$ total phases, for some $T' \geq T$, in order to achieve at least $T$ successful phases with at least constant probability. Therefore, after identifying a sufficiently large value for $T$, we next identify such a value for $T'$.

Note that if for any phase $p_i$, we have that $i_{min}(r_i) > i_{min}(r)$ then of course all nodes possess more than $i_{min}(r)$ tokens and so this claim follows easily. Therefore, assume that for all $i \in R(T)$ that,

$$i_{min}(r_{i+1}) = i_{min}(r_i)$$

(3.83)

$$\Rightarrow S_{min}(r_{i+1}) \subseteq S_{min}(r_i).$$

(3.84)
Note that recursively applying these inequalities also implies that for all \( i \in R(T) \),

\[
i_{\text{min}}(r_i) = i_{\text{min}}(r)
\]

(3.85)

\[
\Rightarrow S_{\text{min}}(r_i) \subseteq S_{\text{min}}(r).
\]

(3.86)

Furthermore, if there exists a phase \( p_i \) where \( n_{\text{min}}(r_i) < n/2 \), since we assume that \( i_{\text{min}}(r_i) = i_{\text{min}}(r) \), by definition this means that more than \( n/2 \) nodes possess more than \( i_{\text{min}}(r) \) tokens. Since this again satisfies the claim, assume for this sequence of \( T \) successful phases that for all \( i \in R(T) \) that,

\[
n_{\text{min}}(r_i) \geq n/2
\]

(3.87)

\[
\Rightarrow n_{\text{min}}^*(r_i) = n - n_{\text{min}}(r_i),
\]

(3.88)

where Line 3.88 follows from the definition of \( n_{\text{min}}^*(r_i) \). Fix any successful phase \( p_i \) for some \( i \in R(T) \), beginning at round \( r_i \). Now, notice that since we assume \( i_{\text{min}}(r_{i+1}) = i_{\text{min}}(r_i) \) (Line 3.83), it follows that \( n_{\text{min}}(r_{i+1}) \leq n_{\text{min}}(r_i) \). Combining this fact with the assumption on Line 3.88, it follows that,

\[
n_{\text{min}}^*(r_{i+1}) = n - n_{\text{min}}(r_{i+1})
\]

(3.89)

\[
\geq n - n_{\text{min}}(r_i)
\]

(3.90)

\[
= n - (n - n_{\text{min}}^*(r_i))
\]

(3.91)

\[
= n_{\text{min}}^*(r_i)
\]

(3.92)

\[
\Rightarrow n_{\text{min}}^*(r_{i+1}) \geq n_{\text{min}}^*(r_i).
\]

(3.93)

\footnote{Note that this follows since we can only shrink the number of nodes with the minimum number of tokens as long as the minimum number of tokens stays the same.}
Recall, by definition, for all $i \in R(T)$ we have that the number of nodes in $S_{\text{min}}(r_{i+1})$ which participate in a productive connection during successful phase $p_{i+1}$ is at least $\psi \cdot n^*_{\text{min}}(r_{i+1})$. By Line 3.86, this implies that at least $\psi \cdot n^*_{\text{min}}(r_{i+1})$ nodes of $S_{\text{min}}(r_i)$ participate in a productive connection this phase. Lastly, by Lines 3.93 and 3.88, we have that for all $i \in R(T)$ this quantity is at least,

$$\psi \cdot n^*_{\text{min}}(r_{i+1}) \geq \psi \cdot n^*_{\text{min}}(r_i) \quad (3.94)$$

$$= \psi \cdot (n - n_{\text{min}}(r_i)) \quad (3.95)$$

for all $i \in R(T)$. Note that by Line 3.88, $n - n_{\text{min}}(r_i)$ denotes the number of nodes which begin phase $p_i$ with more than $i_{\text{min}}(r_i) = i_{\text{min}}(r_1)$ tokens. Therefore, after phase $p_i$, the number of nodes with more than $i_{\text{min}}(r)$ tokens is at least,

$$n - n_{\text{min}}(r_{i+1}) \geq n - n_{\text{min}}(r_i) + \psi \cdot (n - n^*_{\text{min}}(r_i)) \quad (3.96)$$

$$= (1 + \psi) \cdot (n - n_{\text{min}}(r_i)). \quad (3.97)$$

Taking a step back, what we have shown with Line 3.97 that each successful phase $p_i$ for $i \in R(T)$ grows the number of nodes with more than $i_{\text{min}}(r)$ tokens by a multiplicative factor of $1 + \psi$. We can therefore recursively apply this observation $T$ times to get the number of nodes with more than than $i_{\text{min}}(r)$ tokens after $T$ successful phases. This gives that the
number of nodes with more than $i_{\text{min}}(r)$ tokens after $T$ successful phases is at least,

$$n - n_{\text{min}}(r_{T+1}) \geq (1 + \psi) \cdot (n - n_{\text{min}}(r_T))$$ (3.98)

$$\geq (1 + \psi)^2 \cdot (n - n_{\text{min}}(r_{T-1}))$$ (3.99)

$$\ldots$$ (3.100)

$$\geq (1 + \psi)^T \cdot (n - n_{\text{min}}(r_1))$$ (3.101)

$$= (1 + \psi)^T \cdot (n - n_{\text{min}}(r))$$ (3.102)

$$\Rightarrow n - n_{\text{min}}(r_{T+1}) \geq (1 + \psi)^T.$$ (3.103)

Note that the last line follows as long as $n - n_{\text{min}}(r) \geq 1$, meaning that at least some node possesses more than $i_{\text{min}}(r)$ tokens. This is true as long as $C(r) > 1$, assumed by our lemma statement. Therefore, to find a value for $T$ such that more than $n/2$ nodes possess more than $i_{\text{min}}(r)$ tokens after $T$ successful phases, it is sufficient to solve for any value $T$ satisfying the following,

$$(1 + \psi)^T > n/2$$ (3.104)

$$\Rightarrow T \ln (1 + \psi) > \log n - 1$$ (3.105)

$$\Rightarrow T > \frac{\log n - 1}{\log (1 + \psi)}.$$ (3.106)

Therefore, our next step is to find a more convenient value of $T$ which satisfies the inequality on Line 3.106. Leveraging the inequality $1 + \psi \geq 2^\psi$ for all $\psi \in [0, 1]$ (e.g., see Fact 3.1 of [57]), we can simplify the above to show that it is sufficient to choose
\[ T = \lceil (1/\psi) \log n \rceil \] since, 

\[
T = \lceil (1/\psi) \log n \rceil \tag{3.107}
\]

\[
\geq (1/\psi) \log n \tag{3.108}
\]

\[
= \frac{\log n}{\log (2^\psi)} \tag{3.109}
\]

\[
> \frac{\log n - 1}{\log (2^\psi)} \tag{3.110}
\]

\[
\geq \frac{\log n - 1}{\log (1 + \psi)}. \tag{3.111}
\]

First, assume that \(2^9 \geq \log \Delta\) such that \(\psi = 2^{-25} \alpha\). Substituting this value for \(\psi\) into \(\lceil (1/\psi) \log n \rceil\) yields that it is sufficient to choose \(T\) in this case such that, 

\[
T = \lceil (1/\psi) \log n \rceil \tag{3.112}
\]

\[
= \left\lfloor \frac{\log n}{2^{-25} \alpha} \right\rfloor \tag{3.113}
\]

\[
= \left\lfloor 2^{25} (1/\alpha) \log n \right\rfloor. \tag{3.114}
\]

Similarly, if instead \(2^9 < \log \Delta\) such that \(\psi = 2^{-16}(\alpha/\log \Delta)\), it is sufficient to choose \(T\) such that, 

\[
T = \lceil (1/\psi) \log n \rceil \tag{3.115}
\]

\[
= \left\lfloor 2^{16} (1/\alpha) \log \Delta \log n \right\rfloor. \tag{3.116}
\]

107
Taking the maximum value for $T$ required by both Lines 3.114 and 3.116, this gives that it is sufficient to choose $T$ such that,

$$
T = \max \left\{ \left\lceil 2^{25}/\alpha \right\rceil \log n, \left\lceil 2^{16}/\alpha \right\rceil \log \Delta \log n \right\} 
$$

(3.117)

$$
= \left\lceil 2^{16}/\alpha \right\rceil \log n \cdot \max \left\{ 2^{9}, \log \Delta \right\} 
$$

(3.118)

All together, this means that after $\left\lceil 2^{16}/\alpha \right\rceil \log n \cdot \max \left\{ 2^{9}, \log \Delta \right\}$ successful phases, more than $n/2$ nodes will possess more than $i_{\min}(r)$ tokens.

We now bound the total number of phases $T'$ required in order to achieve at least $T$ successful phases with at least constant probability. Let $p'_1, \ldots, p'_T$ denote a sequence of $T'$ phases which are not necessarily all successful. Similar to our previous notation, let $r'_i$ for all $i \in R(T')$ denote the first round of phase $p'_i$. Furthermore, let $X_1, \ldots, X_T$ be the sequence of arbitrary random indicator variables where for all $i \in R(T')$, $X_i = 1$ if phase $p'_i$ is successful; otherwise, $X_i = 0$. Let $X = \sum_{i=1}^{T'}$. Furthermore, let $\phi = 2^{-26}$ and note that, by Lemma 3.3.10, we have that for all $i \in R(T')$,

$$
\Pr[X_i = 1 \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}] \geq \phi
$$

(3.119)

for any phase "history" $x_1, \ldots, x_{i-1} \in \{0, 1\}^{i-1}$. This follows since Lemma 3.3.10 provides a lower bound on $\Pr[X_i = 1]$ while making no assumptions about the algorithm execution before round $r'_i$. Therefore, we can apply Lemma 1.3.5 to the sequence $p'_1, \ldots, p'_T$, for $p = \phi$
and $t = T' = \lceil 2T/\phi \rceil$ to get that,

\begin{align*}
\Pr[X \geq T] &= \Pr[X \geq \phi(2T/\phi)/2] \\
&\geq \Pr[X \geq \phi\lceil 2T/\phi \rceil/2] \\
&\geq 1 - \exp(-\phi\lceil 2T/\phi \rceil/8) \\
&\geq 1 - \exp(-\phi(2T/\phi)/8) \\
&= 1 - \exp(-T/4) \\
&= 1 - \exp\left(-\frac{1}{4}\left[2^{16}(1/\alpha) \log n \cdot \max\{2^9, \log \Delta\}\right]\right) \\
&\geq 1 - \exp\left(-2^{14}(1/\alpha) \log n \cdot \max\{2^9, \log \Delta\}\right) \\
&\geq 1 - \exp\left(-2^{14}(1/\alpha) \log n \cdot 2^9\right) \\
&= 1 - \exp\left(-2^{23}(1/\alpha) \log n\right) \\
&\geq 1 - \exp\left(-2^{23} \log n\right).
\end{align*}

Observe that Line 3.129 follows since, by definition, $\alpha \leq 1$. Furthermore, since $\log n \geq \ln n$ for all $n \geq 1$, we can simplify further such that,

$$
\Pr[X \geq T] \geq 1 - \exp\left(-2^{23} \log n\right) 
$$

\begin{align*}
&\geq 1 - n^{-2^{23}} \\
&\geq 1 - 1/n^2.
\end{align*}
Therefore, we have shown that, with probability at least $1 - 1/n^2$, the number of nodes which possess strictly more than $i_{\min}(r)$ tokens is strictly greater than $n/2$ after,

$$T' = \left\lceil \frac{2T}{\phi} \right\rceil$$

(3.133)

$$= \left\lceil 2 \frac{2^{16}(1/\alpha) \log n \cdot \max\{2^9, \log \Delta\}}{2^{-26}} \right\rceil$$

(3.134)

$$< \left\lceil 2 \frac{2^{16}(1/\alpha) \log n \cdot \max\{2^9, \log \Delta\} + 1}{2^{-26}} \right\rceil$$

(3.135)

$$< \left\lceil \frac{2^{18}(1/\alpha) \log n \cdot \max\{2^9, \log \Delta\}}{2^{-26}} \right\rceil$$

(3.136)

$$< \left\lceil 2^{44}(1/\alpha) \log n \cdot \max\{2^9, \log \Delta\} \right\rceil$$

(3.137)

phases of $[4 \log \Delta]$ rounds each.\(^9\) Therefore, we can apply Lemma 1.3.10 to derive the final round complexity to achieve the same guarantee as follows:

$$\left\lceil 2^{44}(1/\alpha) \log n \cdot \max\{2^9, \log \Delta\} \right\rceil \cdot [4 \log \Delta]$$

(3.138)

$$\leq \left\lceil 2^{47}(1/\alpha) \log n \log \Delta \cdot \max\{2^9, \log \Delta\} \right\rceil + 1.$$  

(3.139)

Therefore, the lemma follows. \(\square\)

**Lemma 3.3.12.** Fix a round $r > 0$ such that $n_{\min}(r) < n/2$. With probability at least $1 - 1/n^2$, all of the nodes possess more than $i_{\min}(r)$ tokens by round,

$$r + \left\lceil 2^{47}(1/\alpha) \log n \log \Delta \cdot \max\{2^9, \log \Delta\} \right\rceil + 1 = r + O((1/\alpha) \log n \log^2 \Delta).$$

(3.140)

**Proof.** The strategy of this proof is symmetric to that of 3.3.11. We once again examine a sequence of $T$ successful phases, where $p_i$ denotes the $i$th phase of the sequence, beginning at round $r_i$. Again, for convenience, define the additional round $r_{T+1}$ to be the round

\(^9\)Note that Line 3.136 assumes that $n \geq \Delta > 1$. 

110
immediately after the last round of phase \( p_T \). Fix a round \( r > 0 \) as specified by the lemma statement and let \( r_1 = r \). Note that, by our lemma assumption, we have that \( n_{\min}(r) < n/2 \) which implies that \( C(r) > 1 \). Again, assume that for each \( i \in R(T) \) that \( i_{\min}(r_{i+1}) = i_{\min}(r_i) \). Otherwise, this would indicate that all nodes in \( S_{\min}(r) \) learned a new token, satisfying the claim.

Our goal is to show that at the end of this sequence, all nodes possess more than \( i_{\min}(r) \) tokens with high probability with respect to \( n \). However, instead of the inequality on Line 3.88 of the previous proof which followed from the assumption that \( n_{\min}(r_i) \geq n/2 \) for all \( i \in R(T) \), we have in this case that \( n_{\min}(r_i) < n/2 \). Note by definition this indicates that for all \( i \in R(T) \), we have that \( n_{\min^*}(r_i) = n_{\min}(r_i) \). As in the proof of the previous lemma, let \( \psi = \frac{2^{-16\alpha}}{\max\{2^\alpha \log \Delta \}} \) and recall that we say phase \( p_i \) is successful if at least \( \psi \cdot n_{\min^*}(r_i) \) nodes of \( S_{\min}(r_i) \) participate in a productive connection (and, therefore, learn a new token) this phase. Since there are \( n_{\min}(r_i) \) nodes with the minimum number of tokens at the beginning of phase \( i \), we have that the number of nodes with at most \( i_{\min}(r_i) \) tokens by the end of successful phase \( i \) is upper bounded by,

\[
\begin{align*}
n_{\min}(r_{i+1}) &\leq n_{\min}(r_i) - \psi \cdot n_{\min^*}(r_i) \\
&= n_{\min}(r_i) - \psi \cdot n_{\min}(r_i) \\
&= (1 - \psi) \cdot n_{\min}(r_i).
\end{align*}
\]

We wish to find a value for \( T \) such that, after \( T \) successful phases, there are no nodes with at most \( i_{\min}(r) \) tokens. First, note that after \( T \) recursive applications of the inequality on Line 3.141.
3.143 we have that,

\[ n_{\min}(r_{T+1}) \leq (1 - \psi) \cdot n_{\min}(r_T) \]  

(3.144)

\[ \leq (1 - \psi)^2 n_{\min}(r_{T-1}) \]  

(3.145)

\[ \ldots \]  

(3.146)

\[ \leq (1 - \psi)^T n_{\min}(r_1) \]  

(3.147)

\[ = (1 - \psi)^T n_{\min}(r) \]  

(3.148)

\[ \Rightarrow n_{\min}(r_{T+1}) < (1 - \psi)^T (n/2). \]  

(3.149)

Note that the above leverages our assumption that \( n_{\min}(r) < n/2 \). Further note that Line 3.143 gives the inequality that is symmetric to that of Line 3.97 in the previous lemma.

Intuitively, where before we were growing the number of nodes with more than the minimum number of tokens, here we are shrinking the number of nodes with the minimum number of tokens. Therefore, the remainder of this proof follows a similar to structure to the proof of Lemma 3.3.11 after Line 3.97. Namely, we wish to find the minimum value of \( T \) such that \( n_{\min}(r_{T+1}) < 1 \). Using the above, this gives,

\[ 1 \geq (1 - \psi)^T (n/2) \]  

(3.150)

\[ \Rightarrow 0 \geq T \log (1 - \psi) + \log n - \log 2 \]  

(3.151)

\[ = -T \log \left( \frac{1}{1 - \psi} \right) + \log n - 1 \]  

(3.152)

\[ \Rightarrow T \log \left( \frac{1}{1 - \psi} \right) \geq \log n - 1 \]  

(3.153)

\[ \Rightarrow T \geq \frac{\log n - 1}{\log \left( \frac{1}{1 - \psi} \right)} \]  

(3.154)

Our next step is to more convenient value of \( T \) which satisfies the inequality on Line 3.154. We first invoke the inequality from Lemma 1.3.7 to show that \( 1 - \psi \leq e^{-\psi} \) for all \( \psi \in \mathbb{R} \).
Furthermore, note that for all $x \geq 0$, it follows that $x \ln 2 \leq x$ and, therefore, we have that $e^{-x} \leq 2^{-x}$. Combined with the inequality from Lemma 1.3.7, this gives that $1 - \psi \leq 2^{-\psi}$ since clearly $\psi > 0$. This lets us derive that, in order to satisfy Line 3.154, it is sufficient to choose $T = \lceil (1/\psi) \log n \rceil$ since,

$$T = \lceil (1/\psi) \log n \rceil$$  \hspace{1cm} (3.155)

$$\geq (1/\psi) \log n$$  \hspace{1cm} (3.156)

$$> \frac{\log n - 1}{\psi}$$  \hspace{1cm} (3.157)

$$= \frac{\log n - 1}{\log (2^\psi)}$$  \hspace{1cm} (3.158)

$$= \frac{\log n - 1}{\log \left( \frac{1}{1 - \psi} \right)}$$  \hspace{1cm} (3.159)

$$\geq \frac{\log n - 1}{\log \left( \frac{1}{1 - \psi} \right)}$$  \hspace{1cm} (3.160)

Note that this value of $T = \lceil (1/\psi) \log n \rceil$ is exactly the value of $T$ derived in the proof of Lemma 3.3.11. Therefore, the remainder of the proof is identical to the previous proof to derive that for any round $r > 0$, all nodes possess more than $i_{\min}(r)$ tokens after at most,

$$\left[ 2^\alpha (1/\alpha) \log n \log \Delta \cdot \max \{2^g, \log \Delta \} \right] + 1$$  \hspace{1cm} (3.161)

taxt rounds with probability at least $1 - 1/n^2$. \hfill \Box

We now have everything we need to prove our main theorem, combining the claims of Lemmas 3.3.11 and 3.3.12 to bound the number of rounds required to grow the minimum token set size.

Proof (of Theorem 3.3.2). Fix a round $r > 0$ with minimum token set size $i_{\min}(r) < k$. We begin by showing that after at most $O((1/\alpha) \log n \log^2 \Delta)$ rounds, all nodes possess more
than $i_{\text{min}}(r)$ tokens with probability at least $1 - 2/n^2$. First, recall that $C(r)$ denotes the number of unique token set sizes at the beginning of round $r$ for which at least one node contains a token set of that size. From Lemma 3.3.5, we have that if $C(r) = 1$ then in round $r + 1$ either $C(r + 1) > 1$ or $i_{\text{min}}(r + 1) > i_{\text{min}}(r)$. If the latter is true, we are done. Therefore, assume that instead $C(r) > 1$, which is required by Lemmas 3.3.11 and 3.3.12 (where the latter holds implicitly).

In Lemma 3.3.11 we showed that if $n_{\text{min}}(r + 1) \geq n/2$, then there is a round $r'$ where $r' = r + O((1/\alpha) \log n \log^2 \Delta)$, such that more than $n/2$ nodes possess more than $i_{\text{min}}(r + 1)$ tokens (i.e., $n_{\text{min}}(r') < n/2$) with probability at least $1 - 1/n^2$. Assuming this occurs, then by Lemma 3.3.12, there exists a round $r'' = r' + O((1/\alpha) \log n \log^2 \Delta)$ such that all nodes possess more than $i_{\text{min}}(r + 1)$ tokens with at least this same probability.\footnote{Note that if $n_{\text{min}}(r + 1) < n/2$ then we an "skip" immediately to applying the result of Lemma 3.3.12 (i.e., $r' = r + 1$).} Via a union bound over either lemma failing to satisfy its objective, we have that with probability at least $1 - 2/n^2$ that $i_{\text{min}}(r'') \geq i_{\text{min}}(r) + 1$ for round $r$ and and some round $r'' = r + O((1/\alpha) \log n \log^2 \Delta)$.

Furthermore, note that the minimum token set size can increase at most $k$ times as there are at most $k$ tokens to begin with. Therefore, via union bound over all $k$ token set sizes, with probability at least $1 - 2k/n^2$, all nodes learn all $k$ tokens in at most $O((k/\alpha) \log n \log^2 \Delta)$ total rounds. Since $k \leq n$, we have that $1 - 2k/n^2 \geq 1 - 2/n$.

Therefore, the theorem follows. \hfill \Box

### 3.4 Asynchronous One-Shot Gossip in Peer-to-Peer Networks

The mobile telephone model captures the basic dynamics of the peer-to-peer libraries included in standard smartphone operating systems. This abstraction, however, makes simplifying assumptions—namely, the assumption of synchronized rounds. In this section we analyze the performance of random gossip processes in a more realistic version of
the model that eliminates this assumption. In particular, we first define the asynchronous mobile telephone model (aMTM), which describes an event-driven peer-to-peer abstraction in which an adversarial scheduler controls the timing of key execution events.

An algorithm specified in the aMTM should be directly implementable on real hardware without the need to synchronize or simulate rounds. This significantly closes the gap between theory and practice. With this in mind, after defining the aMTM, we adapt and analyze our algorithm for the synchronous setting for this new model. In this more realistic asynchronous model, different processes can be running at vastly different and changing speeds, invalidating the clean round-based analysis from the previous section. We will show, however, that even in this more difficult setting, random gossip processes can still be shown to spread tokens with speed that increases with available connectivity.

3.4.1 Model Definition

Similar to the MTM, the topology of the underlying network is defined by an undirected graph. Furthermore, the behavior of the nodes in the aMTM is similarly constrained by a fixed scan-and-connect behavioral loop in which nodes: update their own advertisement, wait to hear new advertisements from at least some neighbors, decide whether to act on these advertisements by attempting to form a connection with a neighbor, then repeat. Unlike the MTM, however, nodes do not progress through this loop in a synchronized manner. Instead, their execution is subject to variable delays which are decided by an adversarial scheduler. This loop is formalized in Algorithm 3. The model implements the methods update, receiveAds, and blockForConn, which abstract the details of the underlying asynchronous communication. In contrast, the methods INITIALIZE, and GETTAG, SELECT, and COMMUNICATE describe the behavior of a particular algorithm. Therefore, it is these methods that must be implemented by an algorithm designed for the aMTM.
Algorithm 3 The aMTM interface (for device $u$)

1: $state \leftarrow \text{idle}$
2: $\text{INITIALIZE}()$
3: 
4: while true do
5:   $tag \leftarrow \text{GetTag}()$
6:   $\text{update}(tag)$
7: 
8:   $receiver \leftarrow \text{null}$
9:   $A \leftarrow \text{receiveAds}()$
10:  if $A \neq \emptyset$ then
11:     $receiver \leftarrow \text{SELECT}(A)$
12:  
13:  if $receiver \neq \text{null}$ then
14:     $state \leftarrow \text{blockForConn}(receiver)$
15:  
16:  if $state = \text{connected}$ then
17:     $\text{COMMUNICATE}(receiver)$
18:  $state \leftarrow \text{idle}$

As is standard with asynchronous network models, we constrain the model’s behavior with respect to a set of maximum delays and bit rates specified for its key communication activities. We define these delays for a given execution with the parameters $\delta_{\text{update}}$, $\delta_{\text{conn}}$, and $R_{b}$. The values of these parameters are not known to the algorithm and can change from execution to execution. We detail the guarantees they help specify below.

Advertisement Guarantees

If a node $u$ calls $\text{update}$ at some time $t$, then the model guarantees that every neighbor of $u$ must receive an advertisement from $u$ in the interval $t$ to $t + \delta_{\text{update}}$, and that only advertisements $u$ passed to $\text{update}$ during this interval are received in this interval. Notice that there is no guarantee that $u$’s neighbors receive all of its advertisements. It is possible,
for example, that $u$ advertises $a$ at a given time, then loops back around in less than $t_{update}$
time and replaces this with a new advertisement $a'$ before any neighbor had a chance to receive $a$. On the other hand, once $u$ begins advertising, its neighbors will hear from $u$ at least once every $\delta_{update}$ time.

**Connection Attempt Guarantees**

The parameter $\delta_{conn}$ bounds the maximum time required for the `blockedForConn` model method to resolve a connection attempt and return whether or not the attempt succeeded. In more detail, when $u$ calls `blockedForConn`(v), for some neighbor $v$, the model guarantees to deliver a connection proposal to $v$. If $v$ is already engaged in a connection (i.e., it has previously accepted a proposal and the resulting connection is still open), it will reject $u$’s proposal. Otherwise, it will accept the proposal. The model must deliver the proposal and the response within this interval of length $\delta_{conn}$. The loop blocks until this underlying communication completes and `blockForConn` can return the status of the connection. Notice that as in the synchronous model, these guarantees restrict each node to at most one incoming connection at a time.

**Communication Guarantees**

Assume $v$ accepts $u$’s connection proposal. At this point, they are connected and can communicate as specified by their respective `Communicate` methods. For many algorithms, such as the gossip strategy studied in this thesis, we simply specify what occurs during this connection in the sender’s `Communicate` routine. When implementing algorithms, however, this behavior must be explicitly specified for both the sender and receiver roles. The amount of time required by these interactions depends on both the amount of information transmitted by `Communicate` and the transmission rates determined by the model. We use the parameter $R_b$ to bound the *minimum* bit rate at which the model can transmit information between
a connected pair of neighbors. We assume that when a call to \textsc{Communicate} returns, the connection is closed. It follows that each node can participate in at most one outgoing connection at a time.

3.4.2 Upper Bound for Asynchronous One-Shot Gossip

We now introduce our asynchronous random diffusion gossip algorithm to achieve an upper bound for the asynchronous counterpart of the algorithm analyzed in Section 3.3.3.

\textbf{Algorithm}

This algorithm, formalized in the pseudocode of Algorithm 4, adapts the strategy of synchronous random diffusion gossip to the asynchronous setting. The main difference is that in each loop iteration, a node selects a neighbor for connection from the set of advertisements it has received since the last iteration. In the synchronous setting, by contrast, a node is always considering the latest advertisements from all of its neighbors.\(^{11}\) If a node receives some number of connection proposals, it accepts one such "incoming" connection proposal (chosen arbitrarily) as long as it not currently participating in an incoming connection with another node. We assume that these incoming connections are handled separately from the main algorithmic loop instantiated by Algorithm 4 which is responsible for initiating outgoing connections. Note that this implies each node can be involved in at most two concurrent connections at any one time.

\(^{11}\)Note that, as in Section 3.3.3, we assume that hash collisions do not occur. However, for completeness, we could specify that the hashed value includes a random seed, \(r\). In the synchronous setting, we used \(r\) to denote the round number, which could be hashed as part of the advertisement in Algorithm 2 in order to eventually break symmetry due to hash collisions. Since we do not have access to synchronized round numbers in the asynchronous setting, each node, \(u\), could instead simply generate \(r\) randomly and includes \(r\) in its advertisement so that \(u\)'s neighbors can recompute their own hash values to compare their token sets against \(u\)'s. This would ensure that symmetry is likely to be broken eventually and, therefore, may help approximate the setting where hash collisions do not occur.
Once nodes are connected, they perform the same token transfer protocol leveraged in Algorithm 2, in which at most two tokens are transferred between the connected nodes in such a way that the node with the fewest tokens is guaranteed to learn a new token.

**Algorithm 4** Asynchronous random diffusion gossip (for node u)

1: function **Initialize**
2:   \( T \leftarrow \) initial tokens (if any) known by \( u \)
3:   \( H \leftarrow \) shared hash function
4: 
5: function **GetTag**
6:   \( \text{return} \langle H(T), |T|, u \rangle \)
7: 
8: function **Select(A)**
9:   \( \hat{s} \leftarrow \min \{|s \in 0, 1, \ldots, k | \langle h, s, v \rangle \in A, h \neq H(T)\}\)
10:  \( \hat{N} \leftarrow \{v \in N(u) \mid \langle h, \hat{s}, v \rangle \in A, h \neq H(T)\}\)
11:  \( \text{return} \) node chosen uniformly at random from \( \hat{N} \)
12: 
13: function **Communicate(v)**
14:   \( (\text{send a connection proposal to } v \text{ if such a node, } v, \text{ exists}) \)
15:   \( (\text{if } v \text{ accepts, send/receive a token from the set difference with } v \text{'s token set}) \)

**Complexity Analysis**

We next analyze the time complexity of the asynchronous random diffusion gossip algorithm. To simplify our formal statement of the claim, we define the parameter \( \delta_{\text{max}} \) such that

\[
\delta_{\text{max}} = \delta_{\text{update}} + \delta_{\text{conn}} + 2b_{\text{max}}/R_h \tag{3.162}
\]

where \( b_{\text{max}} \) describes the maximum size, in bits, of a gossip token.
Note that, because our algorithm only transfers at most two tokens in each call to Communicate, each such call requires time at most \(2\delta_{\text{max}}/R_b\).\(^{12}\) The \(\delta_{\text{update}}\) and \(\delta_{\text{conn}}\) parameters upper bound the time required to get through the update and blockedForConn methods, respectively. It follows that each iteration of our gossip algorithm’s main aMTM loop requires at most \(\delta_{\text{max}}\) time, making \(\delta_{\text{max}}\) a useful aggregate parameter for bounding asynchronous time complexity.

**Theorem 3.4.1.** With probability at least \(1 - 2/n\), the asynchronous random diffusion gossip algorithm solves the gossip problem in \(O((k/\alpha)\log n \log^2 \Delta \cdot \delta_{\text{max}})\) time, where \(k\) is the number of tokens, \(n\) is the network size, \(\alpha\) is the vertex expansion of the network, and \(\Delta\) upper bounds the maximum degree of the network.

Fix the topology graph \(G = (V, E)\). For the analysis that follows, we repurpose much of our notation from the synchronous version of the one-shot gossip problem studied in Section 3.3.3. We will accomplish this through a slight abuse of notation in which we take any element parameterized with an integer round \(r > 0\) in Section 3.3.3 and redefine it here with respect to a real time \(t > 0\). First, fix a real time \(t > 0\) and node \(u \in V\) and let \(T_u(t)\) be the token set of node \(u \in V\) at time \(t\). Note that this definition resembles our definition of \(T_u(r)\) in the previous section which described \(u\)’s token set at the beginning of round \(r\). Similarly, we can adapt our notions of \(s_u(r)\) and \(N_u(r)\) for the asynchronous setting. Let \(s_u(r)\) be the minimum token set size of any node \(v \in N(u)\) such that \(s_u(t) = \min\{|T_v(t)| \in 0, \ldots, k \mid v \in N(u)\}\). Recall from our graph preliminaries defined in Section 1.3, that \(N(u)\) denotes \(u\)’s exclusive neighborhood in \(G\). Next, let \(N_u(t)\) denote \(u\)’s productive neighbors at time \(t\). Formally, \(N_u(t) = \{v \in N(u) \mid s_u(t) = |T_v(t)|, H(T_u(t)) = H(T_v(t))\}\).

\(^{12}\)As in Algorithm 2 for the synchronous MTM in Section 3.3.3, we omit the time to determine which token in the set difference to transfer, since we assume this overhead is subsumed by the cost of sending the token itself.
For a fixed integer size \( i \in 0, \ldots, k \), let \( S_i(t) = \{ u \in V \mid i = |T_u(t)| \} \), be the set of nodes with exactly \( i \) tokens at time \( t \). Furthermore, let \( n_i(t) = |S_i(t)| \) be the number of these tokens where \( n^*_i(t) = \min\{n_i(t), n - n_i(t)\} \). We define \( i_{\min}(t) = \min\{i \in 0, \ldots, k \mid n_i(t) > 0\} \) as the minimum number of tokens \( i \) for which at least one node possesses exactly \( i \) tokens. For convenience of notation, we define \( S_{\min}(t) = S_{i_{\min}(t)}, n_{\min}(t) = n_{i_{\min}(t)} \), and \( n^*_{\min}(t) = n^*_j(t) \) when \( j = i_{\min}(t) \). Lastly, let \( C(t) = ||\{i \in 0, \ldots, k \mid n_i(t) > 0\}|| \) be the number of unique token set sizes \( i \) for which at least one node possesses exactly \( i \) tokens.

With the above notation, we can further adapt our notions of the productive and minimum productive subgraphs of \( G \) from Definitions 3.3.3 and 3.4.4.

**Definition 3.4.2.** At time \( t > 0 \), define the productive subgraph \( G(t) \) of the graph topology \( G = (V, E) \) as the undirected graph \( G(t) = (V, E(t)) \) such that,

\[
E(t) = \{ \{u, v\} \in E \mid v \in N_u(t) \} .
\]  

(3.163)

As was the case with its synchronous counterpart, \( G(t) \) describes the subgraph of \( G \) whose edge set \( E(t) \) denotes which pairwise connections in \( E \) would result in a productive transfer of tokens. We further define the minimum productive subgraph, whose edge set considers only the bipartite subgraph of \( G(t) \), consisting of only the nodes in \( S_{\min}(t) \) and their neighbors in \( V \setminus S_{\min}(t) \).

**Definition 3.4.3.** At time \( t > 0 \), define the minimum productive subgraph \( G_{\min}(t) \) as the undirected bipartite subgraph \( G_{\min}(t) = (L_{\min}(t), R_{\min}(t), E_{\min}(t)) \) such that:

- \( L_{\min}(t) = \{u \in V \setminus S_{\min}(t) \mid N(u) \cap S_{\min}(t) \neq \emptyset\} \),

- \( R_{\min}(t) = \{u \in S_{\min}(t) \mid N(u) \cap (V \setminus S_{\min}(t)) \neq \emptyset\} \), and

- \( E_{\min}(t) = \{\{u, v\} \in E(t) \mid u \in L_{\min}(t), v \in R_{\min}(t)\} \).
Lastly, for fixed time $t > 0$, let $G(L, R)$ be the bipartite subgraph of $G_{min}(t)$ induced by node subsets $L \subseteq L_{min}(t)$ and $R \subseteq R_{min}(t)$. For these partitions, let $N_{L,R}(u)$ be the neighbor set of node $u \in V$ in $G(L, R)$ and let $d_{L,R}(u) = |N_{L,R}(u)|$. Furthermore, for convenience, let $N_{min}(u) = N_{I_{min}(t),R_{min}(t)}(u)$ be the neighbors of $u$ in the minimum productive subgraph $G_{min}(t)$ and $d_{min}(u) = |N_{min}(u)|$.

Now that we have imported all of the required notation from the synchronous setting, note that some additional care is required in dealing with these graphs in the asynchronous model. Namely, in a round-based setting, we can fix the productive subgraph at the beginning of the round and know that all nodes will make connection decisions based on that exact graph during the round. In the asynchronous model no such guarantees hold. We might fix a productive subgraph at some time $t$, for example, but that graph can change before all the nodes get a chance to learn it and make a connection decision.

To handle this nuance, we introduce our first pieces of notation unique to our asynchronous analysis. Fix some time $t$ at which some node $u$ calls Select. Let $\hat{N}_u(t)$ and $\hat{s}_u(t)$ be the values calculated on Lines 9 and 10 of Algorithm 4, respectively, during this call to Select. These values are calculated from the advertisement set $A_u(t)$ which is passed to node $u$’s call to the Select function at time $t$. Note that for a particular time $t$ and node $u$, $N_u(t)$ and $\hat{N}_u(t)$, and $s_u(t)$ and $\hat{s}_u(t)$, can differ from one another, as $N_u(t)$ and $s_u(t)$ are based on the status of the network at exactly time $t$, whereas $\hat{N}_u(t)$ and $\hat{s}_u(t)$ are based on the advertisement set passed to Select at time $t$ (which may by that point already be out of date). Also note that $\hat{N}_u(t)$ and $\hat{s}_u(t)$ are undefined for times that do not correspond to a Select call. To help tame this reality that a given node’s snapshot of the network can become out of date before it has a chance to act on it, we introduce the following definition concerning snapshots of the changing minimum productive subgraph.
Definition 3.4.4. Fix a time interval \([t_1, t_2]\) and two nodes \(u, v \in V\) such that \(\{u, v\} \in E_{\min}(t_1)\). We say that \(u\) properly considers \(v\) with respect to \(t_1\) during this interval if there exists a time \(t_{\text{consider}}\), where \(t_1 \leq t_{\text{consider}} \leq t_2\), such that \(u\) calls \texttt{Select} at \(t_{\text{consider}}\), \(v \in \hat{N}_u(t_{\text{consider}})\), and \(|\hat{N}_u(t_{\text{consider}})| \leq d_{\min}(u)\).

Note that an important implication of the above definition. That is, if \(u\) properly considers \(v\) with respect to \([t_1, t_2]\), then \(u\) attempts to connect with \(v\) in this interval with at least the same probability as it would in a round of the synchronous algorithm, corresponding to minimum productive subgraph \(G_{\min}(t_1)\). This property is critical to applying the techniques used to prove Lemma 3.3.8, where the selection behavior of the synchronous algorithm is noted on Line 3.11 and leveraged throughout the proof.

It would simplify our analysis if for any time \(t_1\) we could identify an interval \([t_1, t_2]\) such that \(u\) properly considers \(v\) for every edge \(\{u, v\} \in E_{\min}(t_1)\), as we could then directly apply our analysis from the synchronous case. We cannot, however, guarantee that such intervals always exist in our asynchronous setting. Consider an edge \(\{u, v\} \in E_{\min}(t_1)\), for some time \(t > 0\). It might be the case that before \(u\) can receive an advertisement from \(v\), that some other node connects to \(v\) and transmits a token that removes \(v\) from the minimum productive subgraph. By the time \(u\) subsequently hears from \(v\), it might no longer include it in its set of productive neighbors. In some sense, however, this is a good case as it only increases the probability that \(v\) receives a connection attempt. The following lemma formalizes this intuition by proving that for any endpoint \(v\) in a snapshot of the minimum productive subgraph, \(v\) will be selected with at least the probability that it would if we had run a round of the synchronous algorithm on that snapshot. This will allow us to subsequently apply Lemma 3.3.8, which we carefully reworked from its original version in [57] so that it now only requires that this lower bound on selection probabilities holds. (The original version made use of the exact selection probabilities from the synchronous algorithm.)
Lemma 3.4.5. Fix any time $t_1 > 0$ and node $v \in R_{min}(t_1)$. There exists a time $t_2$, where $t_1 \leq t_2 \leq t_1 + 3\delta_{max}$, such that $v$ participates in a productive connection in the interval $[t_1, t_2]$ with probability at least,

$$1 - \prod_{u \in N_{max}(v)} \left(1 - \frac{1}{d_{min}(u)}\right).$$

(3.164)

Proof. Fix any time $t_1 > 0$ and node $v \in R_{min}(t_1)$ as specified in the lemma statement. We now upper bound a time $t_{select}$ for which we are guaranteed that: for every node in $u \in N(v)$, by time at most $t_{select}$, $u$ call its SELECT method after receiving at least one advertisement from $v$ which was sent at time $t_1$ or later.

First, let $t'_1 \geq t_1$ be the earliest time that is no earlier than $t_1$ and $v$ calls its update method at time $t'_1$. While, at time $t_1$, $v$ might be at an arbitrary point in its execution of the aMTM loop, by the definition of $\delta_{max}$, we have that $v$ will update its advertisement and call its update method by time at most $t'_1 \leq t_1 + \delta_{max}$. At this point, we have that after at most $\delta_{update}$ additional time, every node in $N(v)$ will receive an advertisement which was sent by $v$ in the interval $[t'_1, t'_1 + \delta_{update}]$. Now consider one such node $u \in N(v)$ and note that $u$ may also be at an arbitrary point in the execution of its algorithm loop when it receives such an advertisement from $v$. However, we are still guaranteed by our model definition that $u$ will call its SELECT method after at most $\delta_{max}$ additional time. All together, this means that $u$ calls its SELECT method after receiving an advertisement sent by $v$ at time $t'_1 \geq t_1$ by time at most $t_{select} \leq t'_1 + \delta_{update} + \delta_{max}$. Since $t'_1 \leq t_1 + \delta_{max}$, it follows that,

$$t_{select} \leq t'_1 + \delta_{update} + \delta_{max}$$

(3.165)

$$\leq t_1 + 2\delta_{max} + \delta_{update}.$$  

(3.166)
We will now consider all possible executions of our algorithm over the interval \([t_1, t_{select}]\).

We partition these executions into two disjoint event spaces with respect to \(v\). The first space, which we will denote \(\mathcal{A}\), will contain all executions in which every node \(u \in N_{\text{min}}(v)\) properly considers \(v\) during the interval \([t_1, t_{select}]\). (Recall that, for fixed time \(t_1 > 0\), we define \(N_{\text{min}}(v)\) to be the neighbor set of \(v\) in \(G_{\text{min}}(t_1)\), and \(d_{\text{min}}(v) = |N_{\text{min}}(v)|\).) The second space \(\overline{\mathcal{A}}\) will then simply be the complement of \(\mathcal{A}\), containing all other executions.

Begin with some execution \(a \in \mathcal{A}\). Recall by the definition of \(\mathcal{A}\), every node in \(N_{\text{min}}(v)\) properly considers \(v\) during the interval \([t_1, t_{select}]\) in execution \(a\). Fix one such neighbor \(u \in N_{\text{min}}(v)\). There is some time \(t \in [t_1, t_{select}]\) in our interval such that, at time \(t\), \(u\) makes a call to \(\text{Select}\), during which \(v \in \hat{N}_u(t)\) and \(|\hat{N}_u(t)| \leq d_{\text{min}}(u)\). During this call, \(u\) will select \(v\) for a connection attempt with probability \(1/|\hat{N}_u(t)| \geq 1/d_{\text{min}}(u)\). It follows that the probability that \(u\) does not send \(v\) a proposal is at most \(1 - 1/d_{\text{min}}(u)\).

Since execution \(a\) is in the event space \(\mathcal{A}\), we know that every node \(u \in N_{\text{min}}(v)\) properly considers \(v\) during this interval. Moreover, given that these nodes properly consider \(v\), the probability that two nodes return \(v\) from \(\text{Select}\) is independent (since the selected neighbor is chosen uniformly at random from \(\hat{N}\)). Therefore, if \(X\) denotes the event that \(v\) is returned via a neighbor’s invocation of \(\text{Select}\), we can then lower bound \(\Pr[X]\) as follows:

\[
\Pr[-X] \leq \prod_{u \in N_{\text{min}}(v)} (1 - d_{\text{min}}(u)) \tag{3.167}
\]

\[
\Rightarrow \Pr[X] \geq 1 - \prod_{u \in N_{\text{min}}(v)} (1 - d_{\text{min}}(u)). \tag{3.168}
\]

Assume this occurs and let \(u\) be the node such that \(u\)’s call to \(\text{Select}\) returns \(v\), by definition, \(u\) sends \(v\) a connection proposal which is received at time \(t'\) where \(t' \leq t + \delta_{\text{conn}}\). There are now two possibilities. The first is that \(v\) rejects \(u\)’s connection proposal since it is already engaged in a connection initiated by another node in \(w \in N(v)\). However, this of course means that \(v\) will have participated in a connection by time at most \(t' + 2b_{\text{max}}/R_p\) since each
connection is bounded by time $2b_{\text{max}}/R_b$ and the blocking connection must have started no later than time $t'$. The second possibility is that $v$ accepts $u$’s connection proposal and they engage in a connection. By the same reasoning, this connection also concludes by time at most $t' + 2b_{\text{max}}/R_b$. Now, if both of the connections identified above are productive, the lemma is satisfied in this case. However, it is still possible that once a connection is finally established, the connected nodes have the same token set due to $v$ learning new tokens in the interval $[t_1, t']$. However, this also implies that $v$ participated in a productive connection before time $t'$. Therefore, we have that for any execution $a \in \mathcal{A}$, $v$ participates in a productive connection with probability at least $1 - \Pi_{w \in N_{\text{min}}(v)}(1 - d_{\text{min}}(u))$ by time at most,

$$t' + 2b_{\text{max}}/R_b \leq t + \delta_{\text{conn}} + 2b_{\text{max}}/R_b$$  \hspace{1cm} (3.169) \\
$$\leq t_{\text{select}} + \delta_{\text{conn}} + 2b_{\text{max}}/R_b$$  \hspace{1cm} (3.170) \\
$$\leq t_1 + 2\delta_{\text{max}} + \delta_{\text{update}} + \delta_{\text{conn}} + 2b_{\text{max}}/R_b$$  \hspace{1cm} (3.171) \\
$$= t_1 + 3\delta_{\text{max}}.$$  \hspace{1cm} (3.172)

Since this satisfies the lemma statement, we now consider some execution $a \in \overline{\mathcal{A}}$.

By the definition of $\overline{\mathcal{A}}$, there must be in $a$ some node $u \in N_{\text{min}}(v)$, such that $u$ does not properly consider $v$ in the interval $[t_1, t_{\text{select}}]$. Fix $t'_{\text{select}}$ to be the first time that $u$ calls Select after receiving an advertisement from $v$ that was passed to update at a time greater than or equal to $t_1$. By our definition of $t_{\text{select}}$, we can always identify a time $t'_{\text{select}}$ that satisfies these properties in $[t_1, t_{\text{select}}]$. By assumption, we know that $u$ does not properly consider $v$ during the call to Select at $t'_{\text{select}}$. We consider the two possible reasons for this behavior, and show in both cases $v$ must have already participated in a productive connection between $t_1$ and

---

Note that if only the node connected to $v$ learned new tokens in this interval, this connection will still be productive since we assume that $v \in R_{\text{min}}(t_1)$ and by definition must be connected with a node possessing tokens $v$ does not have.
t′_{select}. The first possible reason is that \( v \notin \hat{\mathcal{N}}_u(t'_{select}) \). By definition, the minimum token set size in the network can never decrease. It follows that if \( v \notin \hat{\mathcal{N}}_u(t'_{select}) \), then \( v \) must have learned at least one token in the interval \([t_1, t'_{select}]\). Therefore, \( v \) participated in a productive connection by time at most \( t'_{select} \).

The second reason that \( u \) might not properly consider \( v \) would be if \( v \in \hat{\mathcal{N}}_u(t'_{select}) \), but \( \hat{\mathcal{N}}_u(t'_{select}) \) is too large such that \(|\hat{\mathcal{N}}_u(t'_{select})| > d_{min}(u)\). However, at time \( t_1 \), exactly \( d_{min}(u) \) neighbors of \( u \) had a token set size of \( i_{min}(t_1) \) (by the definition of \( G_{min}(t_1) \)), with all other neighbors of \( u \) in \( G \) having strictly more tokens. Since nodes cannot lose tokens, the number of \( u \)'s neighbors with at most \( i_{min}(t_1) \) tokens can never increase. If \(|\hat{\mathcal{N}}_u(t'_{select})| > d_{min}(u)\), then \( \hat{s}_u(t'_{select}) > i_{min}(t_1) \), from which it follows that \( v \), along with all of \( u \)'s neighbors with token set size \( i_{min}(t_1) \) at \( t_1 \), must have received at least one token in the interval \([t_1, t'_{select}]\), meaning \( v \) participated in a productive connection during this interval. Therefore, we have shown that, for any \( a \in \overline{\mathcal{A}} \), \( v \) participates in a productive connection in \([t_1, t'_{select}] \subseteq [t_1, t_{select}] \subseteq [t_1, t_1 + 3\delta_{max}] \) in \( a \) with probability 1. Pulling together these pieces, we have partitioned the possible executions in the interval \([t_1, t_{select}] \) into two event spaces and shown that in both, the probability of \( v \) participating in a productive connection by time at most \( t_1 + 3\delta_{max} \) is at least \( 1 - \prod_{u \in N_{min}(v)}(1 - d_{min}(u)) \), as required by the lemma statement. □

The remainder of our proof of Theorem 3.4.1 runs parallel to that of Theorem 3.3.2 for the synchronous version of the algorithm in the MTM. We begin by proving Lemma 3.4.6 which handles the case where \( C(t) = 1 \), similar to that of Lemma 3.3.5 from the synchronous proof. Unlike most of the remaining results used to prove Theorem 3.4.1, the proof of Lemma 3.4.6 is sufficiently straightforward to prove it directly, instead of building off of its synchronous counterpart, Lemma 3.3.5.

**Lemma 3.4.6.** Fix a time \( t > 0 \) such that \( C(t) = 1 \) and \( i_{min}(t) < k \). Either:

1. \( C(t + 3\delta_{max}) > 1 \), or
2. $i_{\text{min}}(t + 3\delta_{\text{max}}) > i_{\text{min}}(t)$.

Proof. This proof follows a somewhat similar structure to that of Lemma 3.4.5. Fix a time $t > 0$ for which $C(t) = 1$. Since $i_{\text{min}}(t) < k$, note that it follows that $E_{\text{min}}(t) \neq \emptyset$. Fix an arbitrary edge $\{u, v\} \in E_{\text{min}}(t)$ and, without loss of generality assume that $u \in L_{\text{min}}(t)$. Note that by time at most $t + \delta_{\text{max}} + \delta_{\text{update}}$, $u$ will receive an advertisement from $v$ which was sent no earlier than time $t$. Furthermore, by no later than time $t + 2\delta_{\text{max}} + \delta_{\text{update}}$, $u$ will call its Select method after receiving such an advertisement. Recall that this is the same line of reasoning used to derive $t_{\text{select}}$ on Line 3.166.

Furthermore, by the same style of proof as that of Lemma 3.4.5, we have that this invocation of $u$’s Select method has only two outcomes, both of which imply that some neighbor $w \in N_{\text{min}}(u)$ participates in a productive connection by time no later than $t + 3\delta_{\text{max}}$. (Note that $N_{\text{min}}(u)$ here is still defined with respect to our fixed time $t$.) The first possibility is that $v$’s advertisement, sent by time no earlier than $t$, indicates that a connection between $u$ and $v$ would be productive. If this is the case, $u$ will send a connection proposal to some node $w \in N_{\text{min}}(u)$. Again, by the logic of Lemma 3.4.5, every result of this event involves $w$ participating in a productive connection by time at most $t + 3\delta_{\text{max}}$.

In the second outcome, since we assume $v \in R_{\text{min}}(t)$, $v$ must have learned a new token since this is the only way for a connection between $u$ and $v$ to be considered unproductive. We now that have some neighbor $w \in N_{\text{min}}(u)$ is guaranteed to learn a new token in the time interval $[t, t + 3\delta_{\text{max}}]$. Now, if every other node in $S_{\text{min}}(t)$ has also learned a new token, it follows that the second objective of the lemma is satisfied. Otherwise, clearly the first objective is satisfies. $\square$

From this point forward, we adapt each result from the analysis in Section 3.3 to the asynchronous setting, arguing that an equivalent argument holds when we remove the
assumption of synchronous rounds. We begin with adapting Lemma 3.3.6 regarding the maximum matching over the minimum productive subgraph.

**Claim 3.4.7** (Adapted from Lemma 3.3.6). For a fixed time \( t > 0 \), let \( m \) be the size of the maximum matching over \( G_{\min}(t) \). It follows that, \( m \geq (\alpha/4) \cdot n^*_{\min}(t) \).

Note that the proof of Lemma 3.3.6 depends only on the topology of \( G_{\min}(r) \) for some round \( r \). As such, the behavior of the algorithm is irrelevant. Therefore, Claim 3.4.7 follows immediately from Lemma 3.3.6. Adapting the remaining analysis used to the prove of Theorem 3.3.2 will require much more care, as several of these results make assumptions regarding the behavior of the synchronous algorithm and model. Therefore, in the discussion that follows, after repurposing each technical result from the synchronous analysis, we carefully argue why the claim still applies without the assumption of synchronized rounds.

Fix a time \( t > 0 \). Although the asynchronous setting provides no notion of synchronized rounds, for the sake of our analysis we can conceptually divide an execution of Algorithm 4 into "simulated rounds" corresponding to disjoint time intervals of length \( 3\delta_{\max} \). Note that this duration comes from the result of Lemma 3.4.5 which showed that this length of time is sufficient for nodes in \( R_{\min}(t) \) to be properly considered by their neighbors in \( G_{\min}(t) \). This observation turns out to be quite significant, as it allows us to argue that the behavior of the asynchronous algorithm is sufficiently similar to that of the synchronous setting, such that it can be combined with the proof of Lemma 3.3.7 to prove the following claim.

**Claim 3.4.8** (Adapted from Lemma 3.3.7). For a fixed time \( t > 0 \), let \( m \) be the size of the maximum matching over \( G_{\min}(t) \). If \( \Delta < \delta \) for any constant \( \delta > 0 \). It follows that the expected number of nodes in \( R_{\min}(t) \) that participate in a productive connection by time \( t + 3\delta_{\max} \) is strictly greater than \( m/\delta \).
Proof. The proof of Lemma 3.3.7 begins with the assumption that there exists a matching of size \( m \) over \( G_{\min}(r) \) for some fixed round \( r > 0 \). It then uses the fact that the probability that some node \( v \in R_{\min}(r) \) engages in a productive connection in round \( r \) is strictly greater than \( 1/\delta \) to show that the expected number of nodes in \( R_{\min}(r) \) which engage in such a connection is strictly greater than \( m/\delta \). Note that we can claim the same result in the asynchronous setting by leveraging Lemma 3.4.5. Fix a node \( v \in R_{\min}(t) \) and note that by Lemma 3.4.5, \( v \) participates in a productive connection before time \( t + 3\delta_{\max} \) with probability at least \( 1 - \prod_{u \in N_{\min}(v)} (1 - 1/d_{\min}(u)) \). Since \( v \in R_{\min}(t) \), clearly \( N_{\min}(v) \neq \emptyset \). Fix some node \( x \in N_{\min}(v) \) and note that,

\[
1 - \prod_{u \in N_{\min}(v)} \left(1 - \frac{1}{d_{\min}(u)}\right) \geq 1 - \left(1 - \frac{1}{d_{\min}(x)}\right)
\]

(3.173)

\[
= \frac{1}{d_{\min}(x)}
\]

(3.174)

\[
> 1/\delta.
\]

(3.175)

Therefore, the claim follows in the same style as the proof of Lemma 3.3.7. \( \square \)

To prove Claim 3.4.8, we required a behavioral property of the asynchronous algorithm which is much weaker than what is provided by Lemma 3.4.5. This is because the proof of Lemma 3.3.7 does not require us to make too many assumptions regarding the behavior of the synchronous algorithm. However, this is not the case for the proof of Lemma 3.3.8, the main technical result of Section 3.3.3, which we adapt next.

**Claim 3.4.9** (Adapted from Lemma 3.3.8). For a fixed time \( t > 0 \) and \( \Delta > 1 \), let \( L \subseteq L_{\min}(t) \) and \( R \subseteq R_{\min}(t) \) be node subsets such that:

1. there is a matching over \( G(L, R) \) of size \( |L| \) and

2. for every \( u \in L \), every neighbor of \( u \) in \( R_{\min}(t) \) is in \( R \).
It then follows that either:

1. The expected number of nodes in $R_{\min}(r)$ that participate in a productive connection by time $t + 3\delta_{\max}$ is at least $\frac{|L|}{12 \log \Delta}$, or

2. we can identify $L'' \subseteq L \cap L_{\min}(t')$ and $R'' \subseteq R \cap R_{\min}(t')$ for some $t' \in \{t, t + 3\delta_{\max}\}$ such that:

   (a) there is a matching over $G(L'', R'')$ of size $|L''|$, 

   (b) for every $u \in L''$, every neighbor of $u$ in $R_{\min}(t')$ is in $R''$, 

   (c) $|L''| \geq \left(1 - \frac{13}{12 \log \Delta}\right)|L|$, and 

   (d) $\mathbb{E}[\sum_{u \in L''} d_{L'', R''}(u)] \leq \frac{5}{6} \sum_{u \in L} d_{L, R}(u)$.

Proof. Although the proof of Lemma 3.3.8 is significantly more sophisticated than that of Lemma 3.3.7, its adaptation to Claim 3.4.9 is, perhaps surprisingly, somewhat direct. This is because the proof of Lemma 3.3.8 was carefully designed with the asynchronous setting in mind. To this end, the only assumption that we make about the behavior of the algorithm in the proof of Lemma 3.3.8 is on Line 3.11.

This line assumes that for a fixed round $r > 0$, the probability that a node $v \in R_{\min}(r)$ participates in a productive connection this round is at least,

$$1 - \prod_{u \in N_{\min}(v)} \left(1 - \frac{1}{d_{\min}(u)}\right).$$

This property is then used to derive several facts in the proof of Lemma 3.3.8 under various assumptions (e.g., those on Lines 3.13 and 3.33). Note that Line 3.176 is equivalent to the expression given by Lemma 3.4.5, defined with respect to a time $t > 0$, over an interval of at most length $3\delta_{\max}$. Therefore, the claim follows. \qed
We now adapt Lemma 3.3.9 for the asynchronous setting. Like that of Claim 3.4.7, this argument does not require us to make any assumptions about the behavior of the algorithm.

**Claim 3.4.10** (Adapted from Lemma 3.3.9). Consider an interval of length $3\delta_{\max}[4 \log \Delta]$ beginning at time $t > 0$ and let $m$ be the size of the maximum matching over $G_{\min}(t)$. It follows that either:

1. $i_{\min}(t + 3\delta_{\max}[4 \log \Delta]) > i_{\min}(t)$, or

2. the expected number of nodes from $S_{\min}(t)$ that participate in a productive connection by time at most $t + 3\delta_{\max}[4 \log \Delta]$ is at least,

$$\min\left\{ \frac{m}{2^{22}}, \frac{m}{2^{13} \log \Delta} \right\} = \frac{m}{2^{13} \max\{2^9, \log \Delta\}}.$$  

This claim follows directly from Claim 3.4.8 and Claim 3.4.9, following the same logic as the proof of Lemma 3.3.9, requiring no additional assumptions about the behavior of the particular algorithm.

Lemma 3.3.10 is the last technical result of Section 3.3.3 which requires specific properties from the behavior of the synchronous algorithm. In particular, it assumes that the number of nodes in $R_{\min}(r)$ which participate in a productive connection during a round $r > 0$ can be treated as the sum of negatively associated random variables. We now argue that a similar claim can be made about the asynchronous algorithm.

**Claim 3.4.11** (Adapted from Lemma 3.3.10). Fix a time $t > 0$ such that $C(t) > 1$. By time at most $t + 3\delta_{\max}[4 \log \Delta]$, at least

$$\frac{2^{-16} \alpha \cdot n_{\min}^*(r)}{\max\{2^9, \log \Delta\}} = \Omega((\alpha / \log \Delta) \cdot n_{\min}^*(t)).$$
nodes of $S_{\min}(t)$ participate in a productive connection with probability at least $2^{-26}$.

Proof. The only behavioral property that the proof of Lemma 3.3.10 requires is that the number of nodes in $R_{\min}(r)$ for some fixed round $r > 0$ which participate in a productive connection in round $r$ can be treated as the sum of negatively associated random variables. In the synchronous setting, this could be shown without too much difficulty since nodes are limited to at most one connection attempt per round.

However, in the asynchronous setting, a node in $u \in L_{\min}(t)$, for some fixed time $t > 0$, can potentially send many connection proposals over an interval of length $3\delta_{\max}$. This can complicate the negative association argument. For example, fix a node $u \in L_{\min}(t)$ and two nodes $v, w \in N_{\min}(u)$ at time $t$. Consider an algorithm execution where $v$ quickly learns a new token at time $t' > t$ but $w$ still possesses $i_{\min}(t)$ tokens. Fix this time $t'$ and assume, for the sake of argument, that $u$ immediately discerns that $v \notin N_{\min}(u)$ at time $t'$ and $w \in N_{\min}(u)$ at time $t'$. Note that, now, the event that $u$ sends a connection proposal to $v$ and the event that it sends a connection proposal to $w$ are no longer necessarily negatively associated. This is because a connection between $u$ and $w$ at time $t'' > t'$ can potentially increase the minimum token set size among $u$’s neighbors and make it so that $v \in N_{\min}(u)$ again at time $t''$. However, the event that $u$ sends a connection proposal to $v$ after $v$ has already engaged in a productive connection (such as in the example above) has no effect on the number of nodes in $R_{\min}(t)$ which participate in a productive connection proposal during this interval.

To elaborate further, for $u \in L_{\min}(t)$, as long as the minimum token set size among a node $u$’s neighbors does not increase, the events that $u$ attempts to connect to different neighbors in $N_{\min}(u)$ are negatively associated by the same reasoning as the synchronous case. Namely, in this scenario, after attempting one connection with a node in $N_{\min}(u)$, $u$ cannot not have increased its opportunity to attempt a connection with a different node in
$N_{\min}(u)$. Therefore, the negative association hold, and the rest of the proof of Lemma 3.3.10, which makes no further assumptions about the algorithm’s behavior, follows easily. □

The proofs of Lemma 3.3.11 and 3.3.12 make no further assumptions regarding the behavior of the synchronous algorithm. Instead, they simply build on the results proved in Lemma 3.3.10 and Lemma 3.3.5. Therefore, we can quickly note that these arguments then follow in the asynchronous setting as well.

**Claim 3.4.12** (Adapted from Lemma 3.3.11). *Fix a time $t > 0$ such that $C(t) > 1$ and $n_{\min}(t) \geq n/2$. With probability at least $1 - 1/n^2$, more than $n/2$ nodes posses more than $i_{\min}(t)$ tokens by time,*

$$t + 3\delta_{\max} \left[ 2^{47}(1/\alpha) \log n \log \Delta \cdot \max\{2^9, \log \Delta\} \right] + 1 = t + O((1/\alpha) \log n \log^2 \Delta \cdot \delta_{\max}).$$

(3.179)

**Claim 3.4.13** (Adapted from Lemma 3.3.12). *Fix a time $t > 0$ such that $i_{\min}(t) < k$ and $n_{\min}(t) < n/2$. With probability at least $1 - 1/n^2$, all of the nodes posses more than $i_{\min}(t)$ tokens by time,*

$$t + 3\delta_{\max} \left[ 2^{47}(1/\alpha) \log n \log \Delta \cdot \max\{2^9, \log \Delta\} \right] + 1 = t + O((1/\alpha) \log n \log^2 \Delta \cdot \delta_{\max}).$$

(3.180)

Both of the above claims follow from $O((1/\alpha) \log n \log \Delta)$ applications of the result of Claim 3.3.9 regarding intervals with duration $O(\log \Delta \cdot \delta_{\max})$, in the same style as the proofs of Lemma 3.3.11 and Lemma 3.3.12, respectively. Note that the latter also builds on Lemma 3.4.6. We now have all the required pieces to prove Theorem 3.4.1 in the same way that we proved Theorem 3.3.2 in Section 3.3.3.
Proof (of Theorem 3.4.1). We have now shown that, for each supporting lemma of Theorem 3.3.2, an equivalent argument can be made in the asynchronous setting.

The proof of Theorem 3.4.1 then follows in the same style of that of Theorem 3.3.2, using Lemma 3.4.6 in place of Lemma 3.3.5 and Claims 3.4.7 through 3.4.13 in place of Lemmas 3.3.6 through 3.3.12. □

3.5 All-to-All Capacity in Peer-to-Peer Networks

We now consider the all-to-all capacity problem in the MTM, which assumes all nodes begin with an infinite sequence of packets to spread to all other nodes. After formally defining the problem in Section 3.5.1, we provide a bound on the maximum achievable throughput in Section 3.5.2. We then present in Section 3.5.3 an algorithm which produces a schedule to achieve this optimal throughput.

3.5.1 Problem Definition

The model we study in this section is the mobile telephone model (MTM), developed in the work of Ghaﬀari and Newport [57]. Note that this is the same model we studied in Section 3.3 in the context of the one-shot gossip problem. For the formal MTM definition, refer to Section 3.3.1.

We now move on to describing the all-to-all capacity problem, beginning with a general definition of our metric of interest, known as throughput. This definition makes use of an object we call a flow set, which is a set \( F = \{(s_i, R_i) \mid i \in R(k)\} \) (for some \( k \geq 1 \)) where each \( s_i \in V \) and \( R_i \subseteq V \) (for node set \( V \)). For a given flow set \( F \), each \((s_i, R_i) \in F\) describes a packet flow of type \( i \); i.e., source \( s_i \) is tasked with sending packets to all the destinations in set \( R_i \). We refer to the packets from \( s_i \) as \( i \)-packets.
A schedule for a given $G$ and $F$ describes a movement of packets through the flows defined by $F$. Formally, a schedule is an infinite sequence of directed matchings, $M_1, M_2, ...$ on $G$, such that the edges in each $M_t$ are labelled by packets, where we define a packet as a pair $(i, j)$ with $i \in R(|F|)$ and $j \in \mathbb{N}$ (i.e., $(i, j)$ is the $j$th packet of type $i$). We require that the packet labels for a schedule satisfy the property that if edge $\{u, v\}$ in $M_t$ is labelled with packet $p = (i, j)$, then there is a path in $\bigcup_{l<t} M_l$ from $s_i$ to $u$ where all edges on the path are labelled with $p$. (It is easy to see by induction that this corresponds precisely to the intuitive notion of packets moving through a peer-to-peer network under the mobile telephone model). We say that a packet $p$ is received by a node $u$ in round $r$ if there is an edge $(v, u) \in M_r$ which is labelled $p$. A packet $(i, j)$ is delivered by round $r$ if every $x \in R_i$ receives it in some round $t$ with $t \leq r$.

Given a schedule $S$ for a graph $G$ and flow set $F$, we can define the throughput achieved by the slowest rate, indicated in packets per round, at which any of the flows in $F$ are satisfied in the limit. Formally:

**Definition 3.5.1.** Fix a schedule $S$ defined with respect to network topology graph $G = (V, E)$ and flow set $F$. We say $S$ achieves throughput, $t$, with respect to $G$ and $F$, if there exists a convergence round $r_0 \geq 1$, such that for every $r \geq r_0$ and every packet type $i$:

$$\frac{\text{del}_i(r)}{r} \geq t,$$

where $\text{del}_i(r)$ is the largest $j$ such that for every $l \leq j$, packet $(i, l)$ has been delivered by round $r$.

The above definition of throughput concerns performance in the limit, since $r_0$ can be arbitrarily large. In some cases, though, we might also be concerned with how quickly we achieve this limit; which we can quantify by providing bounds on the convergence round when relevant.
While the concepts introduced so far generally apply to the wider class of capacity problems, in this thesis we only consider the all-to-all capacity problem, as it is the variation most relevant to the one-shot gossip problem discussed in Section 3.5.3. Formally, the all-to-all capacity problem considers only the following canonical flow set: \( F_{all} = \{(s, V \setminus \{s\}) \mid s \in V\} \). Put plainly, this variation describes the setting where every node in \( V \) has an infinite packet stream to send to every other node in the network.

### 3.5.2 Upper Bound on Achievable Throughput for All-to-All Capacity

In the work of Dinitz et al. [34], the authors prove a tight connection between the achievable broadcast capacity and the largest degree of the minimum degree spanning tree (MDST) in the graph. Here, we formalize the intuition that this connection also exists for the related problem of all-to-all capacity. Recall that, as defined as part of our graph preliminaries in Section 1.3, an MDST, \( T \), of a fixed graph \( G \) is the spanning tree of \( G \) for which the maximum degree of \( T \) is the minimum degree of that of all spanning trees of \( G \). In the following, we use the notation \( \Delta_{mdst}(G) \) to denote the largest degree of the MDST of \( G \).

**Theorem 3.5.2.** Fix a connected network graph \( G = (V, E) \) of size \( n = |V| \). Every schedule achieves all-to-all throughput at most \( O\left(\frac{1}{n \cdot \Delta_{mdst}(G)}\right) \) w.r.t. \( G \) in the mobile telephone model.

**Proof.** Fix some \( G = (V, E) \) as specified by the theorem statement. Because of the restriction that a given node in the mobile telephone model can receive at most one packet per round, it is trivial to calculate that for every packet count \( i \), it requires \( \Omega(n \cdot i) \) rounds for all nodes to deliver their first \( i \) packets to all other nodes. Therefore, if \( \Delta_{mdst}(G) \leq 4 \) (for example), the theorem statement holds trivially.

On the other hand, if \( \Delta_{mdst}(G) > 4 \), this allows us to apply Theorem 1.3.16 for \( k = \Delta_{mdst}(G) - 1 \), which establishes that there exists a non-empty subset \( S \subset V \) such that \( c(G \setminus S) > q \cdot |S| \), for \( q = k - 2 = \Delta_{mdst}(G) - 3 > 1 \) (where, as defined in Section 1.3,
$c(G \setminus S)$ is the number of connected components after removing nodes in $S$ from graph $G$.

For all $n$ nodes to successfully spread $i$ distinct packets requires $k = n \cdot i$ total packets to spread in the network.

We now bound the number of rounds for a single source $s$ to send $i$ distinct packets. Let $C$ be the set of components in $G \setminus S$ that do not include the source $s$. Fix a packet $t$ spread by $s$. We say $t$ arrives at $C_i \in C$ in round $r \geq 1$, if this is the first round in which a node in $C_i$ receives packet $t$. In this case, $t$ must have been previously received by some bridge node in $S$ that is adjacent to $C_i$. This holds because if $t$ can make it from $s$’s component to $C_i$ without passing through a node in $S$, then removing $S$ would not disconnect $C_i$.

Fix any packet count $i \geq 1$. Each packet requires $|C| = c(G \setminus S) - 1 \geq q \cdot |S|$ arrival events before it completes spreading. As we established above, each arrival event requires a given node in $S$ to receive the given packet. Because each node in $S$ can receive at most one packet per round, there are at most $|S|$ arrival events per round in the network. Pulling together these pieces, let $T_i$ be the number of rounds required to spread $i$ packets. We can lower bound this value as:

$$T_i \geq \frac{i \cdot |C|}{|S|} \geq \frac{i \cdot (q \cdot |S|)}{|S|} = iq \geq i \cdot (\Delta_{mdst}(G) - 3).$$

It follows that for every schedule, and every $i$, at least $T_i$ rounds are required to spread $i$ packets. Similarly, we have that at least $k \cdot |C|$ arrival events are required to spread all $k$
packets. Therefore, the number of rounds required to spread all \( k = ni \) packets is at least,

\[
\frac{k \cdot |C|}{|S|} \geq \frac{k \cdot (q \cdot |S|)}{|S|} \quad (3.186)
\]

\[
= niq \quad (3.187)
\]

\[
= ni \cdot (\Delta_{mdst}(G) - 3). \quad (3.188)
\]

This yields a throughput upper bounded by,

\[
\frac{i}{ni \cdot (\Delta_{mdst}(G) - 3)} = \frac{1}{n \cdot (\Delta_{mdst}(G) - 3)}, \quad (3.189)
\]

which yields the theorem. \( \square \)

3.5.3 An Optimal Algorithm for All-to-All Capacity

To match the all-to-all capacity bound from Theorem 3.5.2, here we describe a routing algorithm that achieves all-to-all capacity throughput in \( \Omega\left(\frac{1}{n \cdot (\Delta_{mdst}(G) - 3)}\right) \), when executed in a connected graph \( G \).

**Algorithm**

The high-level idea of the algorithm, formalized in the pseudocode of Algorithm 5, is to leverage a subroutine we call broadcast gossip which solves the one-shot gossip problem in the well-studied CONGEST message passing model, in which each node is permitted to send a different message of \( O(\log n) \) bits to each of its neighbors in every round. We then show that this algorithm spreads each gossip message to all nodes in \( O(D + n) \) rounds in the CONGEST model in a network with size \( n \) and diameter \( D \).

However, note that we cannot execute this algorithm directly in the MTM, in which a node is limited each round to either sending the same advertisement of \( O(\log n) \) bits to its
neighborhood, or a high-throughput pairwise connection with a single neighbor. Therefore, we first describe a mechanism for simulating the CONGEST model in the MTM, using several rounds of the MTM to represent a single round of CONGEST. To build our simulator, we first construct a spanning tree $T$ of $G$, whose maximum degree $\Delta(T)$ comes "close" to $\Delta_{mdst}(G)$. We will refer to this as an approximate MDST.$^{14}$ We then broadcast $\Delta(T)$ to all nodes of the network using standard techniques and edge color $T$ using $O(\Delta(T))$ colors. Note that the strategy used to construct our simulator uses algorithms designed for another well-studied message passing model known as broadcast-CONGEST. In this model, each node sends the same message of $O(\log n)$ bits to its neighborhood each round. In contrast to algorithms for the CONGEST model, note that we can directly implement broadcast-CONGEST algorithms in the MTM using the advertisements.$^{15}$

**Algorithm 5** Streaming gossip strategy for all-to-all capacity.

1: **construct** an approximate MDST $T$
2: 
3: use $T$ to **convergecast** and **broadcast** the max degree $d$ of $T$
4: 
5: **color** the edges in $T$ using $EdgeColor-MTM(d)$
6: (check validity of coloring and repeat if problem found)
7: 
8: **for** token $i > 0$ **do**
9: use edge colors to simulate **broadcast gossip** for token $i$

We begin by describing how the above methodology allows us to simulate the CONGEST model, before introducing our broadcast gossip strategy and analyzing how it combines with our simulator. First, construct a spanning tree $T$ of $G$ whose maximum degree somewhat approximates that of the MDST of $G$ (i.e., $\Delta_{mdst}(G)$). We have a few options available to us in order to accomplish this feat.

$^{14}$Note that the problem of constructing a true minimum-degree spanning tree is known to be NP-hard [51].

$^{15}$For a more detailed description of the CONGEST and broadcast-CONGEST models, refer to our discussion of models for the application layer in Section 4.1.
The first is that we can leverage the centralized algorithm due to Fürer and Raghavachari [51] which constructs a spanning tree with maximum degree \( \Delta_{mdst}(G) + 1 \) in at most \( O(\text{poly } n) \) computational steps. To execute this algorithm in a distributed fashion in the MTM, we can simply use the advertisements to have every node learn the entire graph topology over \( O(n^2) \) rounds.

Another option is to instead use the distributed MDST algorithm from the recent work of Dinitz et al. [33] which computes a spanning tree with maximum degree \( O(\Delta_{mdst}(G) + \log n) \) in at most \( \tilde{O}(D + \sqrt{n}) \) rounds. Note that this algorithm is designed for the broadcast-CONGEST model and therefore only requires messages of \( O(\log n) \) bits each. Therefore, this algorithm can also be easily simulated using the MTM advertisements.

Given our spanning tree \( T \), we now want to ensure that every node knows the maximum degree of \( T \), denoted by \( \Delta(T) \). In the case where the centralized algorithm from [51] was used to construct \( T \), this is trivial as each node knows \( T \) already. To accomplish this in a distributed manner, we instead use standard convergecast and broadcast techniques, which begins with rooting the tree \( T \).

Namely, each node \( u \in V \) initially sets itself as the root of \( T \). If \( u \) learns of a node \( s \in V \) with a larger identifier, \( u \) sets \( v \in N(u) \) as its parent in \( T \) where \( v \) is its unique neighbor on the path between \( u \) and \( s \) in \( T \). Note that, eventually, a node \( s \) will be identified as the unique root of \( T \).\(^{16}\) To convergecast \( \Delta(T) \) to all nodes in the network, \( u \) uses its advertisements to transmit its degree in \( T \) to its parent \( v \) in \( T \), unless \( u \) itself is the root of \( T \). If \( u \) receives a convergecast message from a child with larger than the degree than the one it sent to \( v \), it then sends this degree to \( v \) instead. Let \( s \in V \) be the node with the highest identifier and

\(^{16}\) Note that, in a distributed setting, nodes never truly know whether or not they have identified the true root \( s \) of \( T \). Therefore, to formally implement Algorithm 5, we can have each node \( u \) proceed to the next step of the algorithm (in this case the convergecast of \( \Delta(T) \)) assuming its current "belief" of the root node \( s_u \) is the actual root of \( T \). If \( u \) later learns of a node with a higher identifier than \( s_u \), it then sets this new node as the root of \( T \), updates its parent in \( T \) accordingly, and restarts the convergecast of \( \Delta(T) \) using this new information.
therefore the root of $T$ (although temporarily several nodes may believe they are the root of $T$ until they learn of the existence of $s$). Initially, $s$ sends a broadcast message to all of its children with respect to $T$, indicating the largest value for $\Delta(T)$ that it has heard so far. Similar to the convergecast mechanism, if $v$ then ever receives a convergecast message from a child with a larger degree than the one it most recently broadcast, it broadcasts this degree to its children. Any node which receives a broadcast message similarly forwards this broadcast to its children in $T$. It follows that eventually every node will learn $\Delta(T)$ via broadcast from the root $s$.

We now describe how to edge color $T$ using at most $O(\Delta(T))$ colors, beginning with the formal definition of an edge coloring.

**Definition 3.5.3.** Fix an undirected graph $G = (V, E)$ and palette size $c \in \mathbb{N}$. A $c$-edge coloring of $G$ is a function $\pi : E \to c$ that satisfies the following: if $e_1, e_2 \in E$ are adjacent then $\pi(e_1) \neq \pi(e_2)$.

Let $\Delta$ be the maximum degree of $G$. Clearly, an edge coloring requires at least $\Delta$ colors. Vizing [101] showed that this trivial bound is close to optimal by proving that every graph admits a $(\Delta + 1)$-coloring.

Achieving a $(\Delta + 1)$-coloring with a centralized algorithm is straightforward. Because we will also consider distributed broadcast algorithms, however, we must also discuss how to produce an efficient edge coloring in a distributed manner in the mobile telephone model.

One of the first distributed edge coloring algorithms is described in Luby’s seminal paper on the maximal independent set (MIS) problem [80]. This algorithm produces a $(2\Delta - 1)$-edge coloring of a graph in $O(\log n)$ rounds, with high probability, in the LOCAL model of distributed computing by performing a $(\Delta+1)$-vertex coloring of the line graph of $G$. Note that the LOCAL model, similar to CONGEST, is another well-studied distributed message
passing model. Unlike the CONGEST model, however, in the LOCAL model nodes are allowed to send messages of unbounded size each round.\textsuperscript{17}

We cannot, however, directly run this (or related) distributed coloring strategies in the mobile telephone model as their efficient time complexities heavily leverage the property of the LOCAL model that allows unbounded message sizes.\textsuperscript{18} Our distributed all-to-all capacity algorithm will need to execute the distributed coloring using the $O(\log n)$-bit advertisement tags allowed by our model—a challenge that is equivalent to edge coloring in the broadcast-CONGEST setting with an $O(\log n)$ bandwidth limit. Each broadcast message is therefore only large enough to describe a constant number of colors and/or nodes. Therefore, we propose a strategy, summarized in Algorithm 6, in which nodes repeatedly construct a maximal matching (MM) using the algorithm due to Israel and Itai [68].\textsuperscript{19} In each iteration, the nodes then color the edges of the current matching with a new color and remove them from consideration for future matchings.

Note that the algorithm due to [68] produces a maximal matching in $O(\log n)$ rounds with high probability in $n$. Therefore, to formally implement each iteration of Algorithm 6, we can run the MM algorithm for a fixed duration of $O(\log n)$ rounds. Then, at the end of at most $2\Delta - 1$ iterations, we will have computed a valid coloring over $T$ with high probability.

\textsuperscript{17} Again, for a more complete summary of distributed message passing models, refer to Section 4.1.

\textsuperscript{18} In Luby’s vertex coloring subroutine, for example, each node is responsible for simulating $\Theta(\Delta)$ virtual nodes during an execution of an MIS algorithm. Each node must send an MIS message on behalf of each of its virtual nodes, requiring at least $\Omega(\Delta)$ bits. The message size grows larger when this strategy is applied to the line graph of the original graph. More recent solutions require, at the very least, that nodes frequently describe their current palette of used or unused colors, which also requires $\Omega(\Delta)$ bits.

\textsuperscript{19} Note that the maximal matching algorithm due to Israel and Itai [68] which works in the broadcast-CONGEST model is easily adapted to work with the small advertisement tags in the MTM (it requires nodes to broadcast, at most, a constant number of identifiers per round). Incidentally, we also use this algorithm as a subroutine in a different context in Section 4.3.5, which describes the algorithm in much greater detail.
in $n$ (over a union bound of all $2\Delta - 1 = O(n)$ iterations).\footnote{The significance of executing $2\Delta - 1$ iterations of the algorithm from [68] is discussed in the proof of Lemma 3.5.6, where we show that $2\Delta - 1$ iterations are sufficient for Algorithm 6 to compute a $2\Delta - 1$ coloring of $T$ w.h.p. in $n$.} In the unlikely event that the algorithm fails to compute a valid coloring (due to one or more of its iterations failing to compute a maximal matching in the fixed duration of $O(\log n)$ rounds), we have all nodes with uncolored edges convergecast and broadcast the fact that the coloring is invalid. Recall that this is the same mechanism described previously to spread the value of $\Delta(T)$ after the distributed construction of $T$.

**Algorithm 6** Edge coloring strategy for the mobile telephone model.

```algorithm
1: function $\text{EdgeColorMTM}(\Delta)$
2:     for $i \in R(2\Delta - 1)$ do
3:         construct a maximal matching $M$ using algorithm from Israel and Itai [68]
4:     color edges in $M$ with color $i$
5:         (i.e., for each $e \in M$ set $\pi(e) \leftarrow i$)
6:     remove edges in $M$
7:     return $\pi$
```

Given a spanning tree $T$ of $G$, colored with at most $c \in \mathbb{N}$ colors, we now describe how we can simulate a round of the more powerful CONGEST model over $T$ in at most $c$ rounds of the MTM. Namely, for a fixed round $r > 0$ of the CONGEST model, let the rounds $cr, \ldots, c(r + 1) - 1$ be the sequence of $c$ rounds of the MTM which simulate round $r$. Next, let $m_u(r)$ be the message that $u$ would send to its neighborhood $N(u)$ in round $r$. To simulate round $r$ of the CONGEST model, in round $i \in cr, \ldots, c(r + 1) - 1$ simply sends $m_u(r)$ to node $v \in N(u)$ such that $\{u, v\} \in T$ is of color $c_r$ where $c_r = r \mod \Delta(T))$. Note then that each node $v \in N(u)$ then receives $m_u(r)$ by round at most $c(r + 1) - 1$. Therefore, if $c = 2\Delta(T) - 1 \in O(\Delta(T))$, the above simulation strategy can be used to simulate CONGEST in at most $O(\Delta(T))$ rounds of the MTM.
Next, we introduce an algorithm, which we call \textit{broadcast gossip}, designed to solve the one-shot gossip problem in the CONGEST model. The strategy works as follows.

Every node maintains a FIFO (First In, First Out) message queue initialized to hold its initial gossip message (if it starts with such a message), and a list of sent messages initialized to be empty. At the beginning of each round, each node $u \in V$ does the following. If its queue is non-empty, it dequeues a message, broadcasts it to all of its neighbors, and adds this message to its sent messages list. For each message $m$ received by $u$ in this round, if $m$ is on $u$’s sent messages list (i.e., $u$ already has already sent $m$), $u$ discards the message. Otherwise, $u$ enqueues message $m$ to its message queue.

Pulling together these pieces, given our spanning tree $T$ and CONGEST model simulator which uses $O(\Delta(T))$ rounds to simulate each round of CONGEST, each node executes several instances of the broadcast gossip strategy just described to spread all packets over $T$. Formally, each node $u \in V$ maintains several message queues where the $j$th queue for $j \in \mathbb{N}$ is used for all packets $(i, j)$ for any $i \in \mathbb{N}$ (i.e., the $j$th queue is responsible for the $j$th packet of every packet type $i$). In each simulated CONGEST round, each node $u$ executes the broadcast gossip strategy using the $j'$th message queue such that $j'$ is the smallest value $j$ for which the $j$th message queue is non-empty. A message is then dequeued from the selected queue and sent to all of $u$’s neighbors in $T$ over at most $O(\Delta(T))$ rounds of the MTM (one simulated CONGEST round). Note that, according to this construction, earlier packets are always prioritized over later packets.

\textbf{Complexity Analysis}

We now analyze the throughput achieved by the above algorithm with respect to the all-to-all capacity problem. Formally, we prove the following.
Theorem 3.5.4. With high probability in \( n \), the streaming gossip algorithm achieves an all-to-all throughput in \( \Omega\left(\frac{1}{\Delta_{mds}(G) \cdot n}\right) \) in the mobile telephone model, where \( n \) is the network size and \( \Delta_{mds}(G) \) is the maximum degree of the minimum-degree spanning tree of \( G \).

We first analyze the time complexity of our one-shot gossip strategy.

Lemma 3.5.5. Consider the broadcast gossip algorithm used to spread \( k \geq 1 \) messages in a connected \( G \) with diameter \( D \). All messages spread to all nodes by the end of round \( D + k \) of the \textsc{congest} model.

Proof. We begin by defining two useful pieces of notation: for a given gossip message \( m \), let \( u_m \) be the node that starts with \( m \), and for each \( v \in V \), let \( d_m(v) \) be the shortest path distance between \( v \) and \( u_m \). Fix a specific message \( m \). We first study the spread of this message through the network using induction on the round number. In particular, consider the following inductive hypothesis:

For every \( r \geq 1 \): for every \( v \) and \( \ell \) such that \( d_m(v) + \ell = r \), one of the following two properties must be true of \( v \) after round \( r \): (1) \( v \) has sent \( m \); (2) \( v \) has sent at least \( \ell \) distinct gossip messages.

We begin with the base case \((r = 1)\). There are only two relevant combinations of \( v \) and \( \ell \) values for \( r = 1 \). The first is when \( d_m(v) = 0 \) and \( \ell = 1 \). In this case, \( v = u_m \) so we know \( v \) starts with \( m \) and therefore has a message to broadcast during round 1, satisfying property (2) for \( \ell = 1 \). The second case is when \( d_m(v) = 1 \) and \( \ell = 0 \). In this case, property (2) is vacuously true.

We continue with the inductive step \((r > 1)\). Fix any \( v \) and \( \ell \) such that \( d_m(v) + \ell = r \). Let us consider what has happened by the end of round \( r - 1 \). If \( v \) has sent \( m \) by the end of round \( r - 1 \), then we are done. Moving forward, therefore, assume \( v \) has not sent \( m \) by the end of \( r - 1 \).
Fix some $w$ that is one hop closer to $u_m$ than $v$. Notice that $r-1 = d_m(v) + \ell - 1 = d_m(w) + \ell$. Therefore, by the inductive hypothesis, and our assumption that $v$ has not yet sent $m$, we know that after round $r - 1$, node $v$ has broadcast at least $\ell - 1$ messages and node $w$ has either broadcast $m$ or broadcast $\ell$ messages. Either way, $v$ has at least one new message to broadcast in $r$, meaning that by the end of this round it will have at least satisfied property (2) of the inductive hypothesis.

Stepping back, we can now pull together the pieces to prove the lemma statement. The inductive claim above establishes that for each $v$, $v$ has sent $m$ by round $d_m(v) + k \leq D + k$. This follows because by the above hypothesis, by the end of round $d_m(v) + k$, $v$ has either sent $m$ or at least $k$ other messages. Given that there are only $k$ total messages, the latter property also implies it has sent $m$.

The inductive claim applies for every message $m$. Therefore, every node has sent (and therefore received) every message by round $D + k$, as claimed. □

From our algorithm description, we have that given a spanning tree $T$ of $G$ colored with at most $c$ colors using our strategy in Algorithm 6, where $c = O(\Delta(T))$, such that each node knows $\Delta(T)$, we can simulate a single round of the CONGEST model using at most $c$ rounds in the MTM. This means that we can execute a single instance of broadcast gossip in at most $c \cdot (D + k) \leq O(\Delta(T) \cdot (D + k))$ rounds in the MTM. Therefore, we next derive an upper bound on the number of colors $c$.

In the following lemma, we also consider the time complexity of Algorithm 6, which will be important for deriving the convergence time of our all-to-all capacity algorithm.

**Lemma 3.5.6.** Our edge coloring algorithm for the mobile telephone model produces a $(2\Delta - 1)$-edge coloring in at most $O(\Delta \log n)$ rounds with high probability in $n$.

**Proof.** It follows directly from the definition of the strategy in Algorithm 6 that no two adjacent edges are colored the same color, as this would require two adjacent edges to be
included in the same matching. It is sufficient, therefore, to show that \(2\Delta - 1\) maximal matchings are sufficient to cover every edge in \(E\).

To see why this is true, fix some edge \(\{u, v\}\) in \(G\). The only event that can prevent \(\{u, v\}\) from being included in a given maximal matching is if at least one other edge adjacent to \(u\) or \(v\) is included in the matching. There are at most \(2(\Delta - 1)\) such other edges, so \(\{u, v\}\) must be matched after at most \(2(\Delta - 1) + 1 = 2\Delta - 1\) matchings.

Note that it is well-known that the MM algorithm from Israel and Itai [68] always computes a valid maximal matching, and requires at most \(O(\log n)\) rounds with probability at least \(1 - 1/n^c\) for any arbitrarily large constant \(c \geq 1\) (where \(c\) may influence constants hidden in the asymptotic time complexity).\(^{21}\) Therefore, if we set \(c = 2\), we have that the this algorithm terminates in \(O(\log n)\) rounds of the broadcast-CONGEST model with probability at least \(1 - 1/n^2\). Since we can simulate algorithms for this model using the advertisements of the MTM, the same guarantee follows for each instance invoked by our edge coloring algorithm.

This means that each instance of the algorithm in Algorithm 6 fails to compute a valid MM with probability at most \(1/n^2\). Since \(2\Delta - 1 < 2n\), we can take a union bound over all \(2\Delta - 1\) instances of the algorithm to show that all iterations of Algorithm 6 terminate in \(O(\log n)\) rounds each with probability at least \(1 - 2/n\). Therefore, the time complexity of the algorithm is \((2\Delta - 1) \cdot O(\log n) = O(\Delta \log n)\) w.h.p. in \(n\). \(\square\)

We now have everything we need to bound the throughput achieved by Algorithm 5.

\textit{Proof (of Theorem 3.5.4).} Let \(T\) be the spanning tree produced by the algorithm of Fürer and Raghavachari [51] which guarantees that \(\Delta(T) \leq \Delta_{mdst}(G) + 1\) where \(\Delta(T)\) is the maximum degree of \(T\). By Lemma 3.5.6 we then have that the algorithm produces a valid coloring over this tree with at most \(2\Delta(T) + 1 \leq 2(\Delta_{mdst}(G) + 1) + 1 = 2\Delta_{mdst}(G) + 3\) colors.

\(^{21}\)See, for example, the survey of MM algorithms in Table 2 of [8].
with high probability in \( n \). Note that, although there is a low probability of failure incurred by our coloring strategy, our algorithm construction where each node reports its uncolored edges, we are guaranteed to *eventually* compute a valid coloring over \( T \).\(^{22}\)

By our simulation construction, we then have that it takes at most \( 2\Delta_{mdst}(G) + 3 \) rounds of the MTM to simulate one round of CONGEST. Furthermore, by Lemma 3.5.5 we have that only \( D(T) + k \) simulated rounds of CONGEST are required to send all \( k \) packets over \( T \), where \( D(T) \) is the diameter of \( T \). Therefore, the broadcast gossip strategy delivers all \( k \) packets in at most \( (2\Delta_{mdst}(G) + 3) \cdot (D(T) + k) = O(\Delta_{mdst}(G) \cdot (D(T) + k)) \) rounds of the MTM. Since \( D(T), k = O(n) \), this yields a final all-to-all throughput of,

\[
\Omega\left(\frac{1}{\Delta_{mdst}(G)(D(T) + k)}\right) = \Omega\left(\frac{1}{\Delta_{mdst}(G) \cdot n}\right),
\]

as required by the theorem.

Note that the convergence time of our algorithm depends on the time to construct and color \( T \), as well as the time to flood the message queues of all nodes of the network. Note that the latter takes \( O(D(T)) \) simulated rounds of the CONGEST model where \( T \) is the spanning tree built by our algorithm. By our simulation, this takes \( O(D(T) \cdot \Delta(T)) \) rounds of the MTM.\(^{23}\) Furthermore, with high probability in \( n \), the edge coloring strategy of Algorithm 6 takes \( O(\Delta(T) \log n) \) rounds. Therefore, given a spanning tree \( T \), the total additional convergence time is \( O(\Delta(T) \cdot (D(T) + \log n)) \) rounds of the MTM w.h.p. in \( n \).

Now, using the centralized algorithm from Fürer and Raghavachari [51] to compute \( T \), we need to potentially spend \( O(n^2) \) rounds for all nodes to learn the entire graph topology.

---

\(^{22}\)Note that the costs of computing, coloring, and broadcasting the maximum degree of \( T \) are independent of the number of packets being broadcast. Since our definition of throughput (see Definition 3.5.1) is only concerned about performance in the limit after some convergence round, they have no effect on the final throughput.

\(^{23}\)Observe that this subsumes the \( O(D(T)) \) rounds needed to broadcast \( \Delta(T) \) throughout the network.
and compute $T$ locally. This cost subsumes all other factors in the convergence and makes it so that the convergence time to achieve the throughput specified by Theorem 3.5.4 is at most $O(n^2)$ rounds. Recall, however, that the work of Dinitz et al. [33] showed a distributed MDST approximation algorithm which produces a spanning tree $T$ with $\Delta(T)$ in $O(\Delta_{mdst}(G) + \log n)$ in at most $\tilde{O}(D + \sqrt{n})$ rounds. Using a tree constructed in this way in our above analysis yields an all-to-all throughput of,

$$\Omega\left(\frac{1}{\Delta(T) \cdot n}\right) = \Omega\left(\frac{1}{(\Delta_{mdst}(G) + \log n) \cdot n}\right). \tag{3.191}$$

Note that, since $\Delta(T) = O(\Delta_{mdst}(G) + \log n)$ the convergence time of this algorithm is

$$O(\Delta(T) \cdot (D(T) + \log n)) + \tilde{O}(D + \sqrt{n}) = \tilde{O}(\Delta(T) \cdot D(T)) + \tilde{O}(D + \sqrt{n}) \tag{3.192}$$

$$= \tilde{O}(D(T) \cdot (\Delta_{mdst}(G) + \log n)) + \tilde{O}(D + \sqrt{n}) \tag{3.193}$$

$$= \tilde{O}(D(T) \cdot \Delta_{mdst}(G)) + \tilde{O}(D + \sqrt{n}) \tag{3.194}$$

$$= \tilde{O}(D \cdot \Delta_{mdst}(G) + \sqrt{n}). \tag{3.195}$$

Therefore, while the convergence of this distributed solution can be more efficient, it does not enable optimal throughput for networks with small values for $\Delta_{mdst}(G)$.

For this reason, in practice, we can use a hybrid of the approximate MDST algorithms from [33, 51]. Namely, we can simultaneously execute two instances of Algorithm 5 with each using a different MDST algorithm. Once the instance using a tree computed via by [33] converges after at most $\tilde{O}(D \cdot \Delta_{mdst}(G) + \sqrt{n})$ rounds with high probability in $n$, we can use the resulting simulator until the instance using a tree computed via the algorithm from [51] converges after at most $O(n^2)$ rounds w.h.p. in $n$. This allows us to achieve a
throughput in $\Omega\left(\frac{1}{\Delta_{mdst}(G) \cdot n}\right)$ in $\tilde{O}(D \cdot \Delta_{mdst}(G) + \sqrt{n})$ rounds with high probability in $n$ that then improves to $\Omega\left(\frac{1}{\Delta_{mdst}(G) \cdot n}\right)$ rounds by $O(n^2)$ rounds with high probability in $n$.

**Implications for One-Shot Gossip**

Many results for one-shot gossip in the mobile telephone model like those in 3.3 are expressed with respect to the vertex expansion (denoted $\alpha$) of the graph topology. The best known upper bound is our result of Theorem 3.3.2 which requires $O((k/\alpha) \log^3 n)$ expected rounds to spread $k$ tokens, which is not tight in all graphs as vertex expansion does not necessarily characterize optimal gossip.\(^{24}\)

A key open question is whether it is possible to produce a gossip algorithm that is optimal (or within log factors of optimal) in all network topology graphs. The techniques used in the above capacity bounds help us prove the following, which largely resolves this open question:

**Theorem 3.5.7.** Fix a connected network topology $G = (V, E)$ with diameter $D$, size $n = |V|$, and MDST degree $\Delta_{mdst}(G)$. Every solution to one-shot gossip in $G$ requires $\Omega(\Delta_{mdst}(G) \cdot n)$ rounds in the MTM. There exists an algorithm that solves the problem, w.h.p. in $n$, in at most $\tilde{O}(\Delta_{mdst}(G) \cdot n)$ rounds in the MTM.

**Proof.** In the proof of Theorem 3.5.2, we showed that, for every schedule, at least $n \cdot (\Delta_{mdst}(G) - 3)$ rounds are required for all $n$ nodes to spread $i$ distinct packets. Therefore, at least $n \cdot (\Delta_{mdst}(G) - 3)$ rounds are required to spread a single packet to all nodes. Since this would contradict the existence of a one-shot gossip algorithm which spreads a token in only $o(\Delta_{mdst}(G) \cdot n)$ rounds, the lower bound of $\Omega(\Delta_{mdst}(G) \cdot n)$ then follows.

To derive the upper bound, we consider the streaming gossip strategy of Algorithm 5, run with the distributed approximate MDST algorithm from [33]. After initialization,\(^{24}\) Consider, for example, a path of length $n$, which has $\alpha = 2/n$. It is possible to pipeline $n$ messages through this network in $\Theta(n)$ rounds, which is much faster than $\tilde{O}(n/\alpha) = \tilde{O}(n^2)$.

---

\(^{24}\)Consider, for example, a path of length $n$, which has $\alpha = 2/n$. It is possible to pipeline $n$ messages through this network in $\Theta(n)$ rounds, which is much faster than $\tilde{O}(n/\alpha) = \tilde{O}(n^2)$.\(^{24}\)
this algorithm solves one-shot gossip in \( O((D + n) \cdot \Delta(T)) = O(n \cdot \Delta(T)) \) rounds, where \( T \) is the tree used by the algorithm. Using the distributed algorithm from [33], we can construct a tree \( T \) with \( \Delta(T) \in O(\Delta_{mdst}(G) + \log n) \) in only \( \tilde{O}(D + \sqrt{n}) \) rounds of the broadcast-CONGEST model. Since this time complexity holds for the MTM as well, the time complexity to solve one-shot gossip in the MTM is,

\[
\tilde{O}(D + \sqrt{n}) + O(n \cdot \Delta(T)) = \tilde{O}(n) + O(n \cdot (\Delta_{mdst}(G) + \log n))
\]

(3.196)

\[
= \tilde{O}(n) + \tilde{O}(\Delta_{mdst}(G) \cdot n)
\]

(3.197)

\[
= \tilde{O}(\Delta_{mdst}(G) \cdot n).
\]

(3.198)

Therefore, the theorem claim follows. □

Notice that, the solution described in Theorem 3.5.7 is asymptotically optimal in any graph with \( \Delta_{mdst}(G) \in \Omega(\log n) \) and \( D \in O(n/\log^x n) \) (where \( x \) is the constant from the polylog in the MDST construction time), which describes a large family of graphs. Note that this optimality comes from the fact that when \( \Delta_{mdst}(G) \in \Omega(\log n) \) and \( D \in O(n/\log^x n) \), the time complexity of the solution described in Theorem 3.5.7 reduces to,

\[
\tilde{O}(D + \sqrt{n}) + O(n \cdot \Delta(T)) = \tilde{O}(n/\log^x n + \sqrt{n}) + O(n \cdot (\Delta_{mdst}(G) + \log n))
\]

(3.199)

\[
= O(n) + O(\Delta_{mdst}(G) \cdot n)
\]

(3.200)

\[
= O(\Delta_{mdst}(G) \cdot n),
\]

(3.201)

rounds, matching the lower bound of \( \Omega(\Delta_{mdst}(G) \cdot n) \) rounds from Theorem 3.5.7.

Furthermore, for the subset of graphs with \( \Delta_{mdst}(G) \in o(\log n) \) or \( D \in \omega(n/\log^x n) \), the solution can be expressed as \( \tilde{O}(\Delta_{mdst}(G) \cdot n) \), which is at most a polylog factor slower
than optimal. This is the first known gossip solution to be optimal, or within log factors of optimal, in all graphs, regardless of vertex expansion.
The last layer we consider is the application layer, for which we provide robust distributed algorithms for two key networking primitives with the goal of supporting fault-tolerant end-user applications.

The application layer is the highest level of the OSI model, acting as the interface between network protocols and end-user programs. It provides subroutines which support many different end-user applications, such as web browsers and instant messengers. One class of these subroutines, of particular interest in the area of distributed algorithms research, is the distributed computation of graph structures. In many scenarios a particular graph structure can act as a critical backbone to an application, increasing its efficiency or robustness. For example, consider the task of aggregating information in a local sensor network to minimize the number expensive long-range transmissions required to collect field data. The problem of identifying which devices should be responsible for data collection is greatly simplified by identifying a set of participants which constitutes a maximal independent set over the underlying network topology. This is just one of many examples of how the knowledge of certain graph structures can prove useful to many different applications.

That being said, it is not always possible to have access to these structures at the outset of a given computational task. This is particularly true in the case of wireless networks which we study here, where the communication topology is not known ahead of time, and any required graph structures must be computed from scratch. To make matters worse, as we mentioned in Section 1, wireless devices (and consequently, wireless networks) are
highly susceptible to different types of failures. Therefore, if we hope to inspire practical protocols for the application layer of wireless networks, we must design distributed algorithms which are equally robust in the face of frequent faults. These algorithms must not only be capable of computing a particular graph structure in the presence of failures, but should also be able to repair these structures after they are damaged due to device failures which occur after construction.

In this chapter, we aim to accomplish this goal by devising fault-tolerant algorithms for two of the most fundamental graph structuring problems in distributed algorithms theory: the maximal independent set and maximal matching problems. As we discuss in Section 4.1, these problems are well-studied in standard distributed computation models. It comes as a surprise, therefore, how comparatively little is known about how these structures are impacted by common types of device failures. The results of this chapter, summarized at length in Section 4.2, are intended to help fill this gap and thus empower more robust application layer protocols which may exploit these graph structures. The formal presentation of our results begins in Section 4.3 where we study the time complexity of repairing maximal independent sets and maximal matchings in the presence of crash failures. To this end we introduce a model in Section 4.3.1 which allows a network to "prepare" in order to speed up the time required to recover from the crash failures once they occur. We then use this model extensively to evaluate the impact of the crashes, providing a number of upper and lower bounds for both general graphs and specialized graph classes. Finally, in Section 4.4 we turn to the significantly harder setting of byzantine failures, where participants (in addition to crashing) can also maliciously hinder the construction of a particular graph structure. As we will see, designing an appropriate byzantine model, and even properly defining the distributed problem, poses an intriguing theoretical challenge. We discuss this challenge in Section 4.4.1, which culminates in a computation model which allows us to study the impact of byzantine participants on the distributed construction of maximal independent
sets. Lastly, we quantify this impact with a new upper and lower bound. Note that at this time, our results in Section 4.4 for the byzantine MIS problem and our upper bound for repairing maximal independent sets with crash failures in general graphs in Section 4.3.4 are currently unpublished. All other results in this chapter first appeared in [88].

As in previous sections, before we can provide a detailed presentation of the results we achieve in this thesis, we once again must discussed the existing literature pertaining to existing models.

4.1 Existing Work on Distributed Graph Structures

We now explore the existing results from distributed algorithms theory regarding the maximal independent set and maximal matching problems. While for the previous layers we have explored, many of the abstractions have primarily focused on representing the communication limitations of these layers, this is not the case for the application layer. The reason is that when we consider problems related to the application layer, we assume that these considerations are mitigated by processes implemented for these lower layers. This allows us to design algorithms which abstract away these technical details, and focus on the task of coordinating multiple devices to accomplish some complex distributed task. Indeed this is the entire design philosophy behind the OSI model architecture, where low-level details regarding communication protocols are hidden from the high-level interface exposed by the application layer.

One of the most well-studied models in distributed algorithms theory at the level of the application layer is the LOCAL model of communication (e.g., [80, 1, 54, 93, 8]). This is again a synchronous model where time proceeds in rounds, but contrary the models we have seen thus far, participants in each round can perform an unbounded amount of communication with their neighbors. Furthermore, this communication is assumed to be completely
reliable, once again signifying this idea that any communication issues have already been ironed out in the lower layers of the network stack. The LOCAL model was theorized in order to ask the question of how far do nodes have to explore their local topology to solve a given distributed problem. This is evident in the way that the sheer power of the communication mechanism in this model allows nodes to completely learn their local topology up to distance $d$ in at most the same number of rounds.

However, pragmatically speaking of course unbounded communication is quite unrealistic (again, compare this assumption to any of the models discussed in the sections so far). For this reason, the CONGEST model was introduced, which mirrors the LOCAL model except that messages in each round are restricted to $O(\log n)$ bits per round (where recall $n$ is the number of network participants) (e.g., see [56, 55]). In the classic CONGEST model, what this means specifically is that a different message of $O(\log n)$ bits is potentially sent to each of a node’s neighbors in each round. There is also a version of this model which enforces even stricter communication limitations on the participants. This model, called broadcast-CONGEST, dictates that each node must choose a single message each round, such that the same $O(\log n)$ bits are sent to all of each node’s neighbors in a given round.

While the LOCAL model explores what is possible to compute by a distributed algorithm, these models in contrast explore what is practical given bandwidth limitations.

The maximal independent set (MIS) problem is one of the most fundamental problems in distributed graph algorithms research. A MIS $S$ is described as a set of nodes such that (1) no two nodes in $S$ are neighbors and (2) every node in the graph has an inclusive neighbor in $S$. Such sets of nodes have a large number of practical applications: such as labeling and indexing techniques [50, 70, 53], detecting collusion in voting pools [2], social network analysis [64, 44], discovering association rules in databases [105], among others [79, 52]. An MIS can also be used as a building block to empower solutions to many other graph problems of theoretical interest.
The seminal starting point for randomized distributed MIS research is the simple algorithm of Luby [80] which computes an MIS in $O(\log n)$ expected rounds in the LOCAL model (see also Alon, Babai, and Itai [1]). For deterministic algorithms in the same model, the classic result belongs to Panconesi and Srinivasan [91] which takes $O\left(2^{\sqrt{\log n}}\right)$ rounds. For decades, these algorithms remained the state-of-the-art algorithms for general graphs. However, extensive exploration of the MIS problem has produced many improved strategies. Currently, the best randomized and deterministic algorithms for graphs with "large" maximum degree, $\Delta$, come from a line of work by Ghaffari, Grunau, and Jin [55] (see also [54, 93, 56]). These solutions achieve the bounds of $O(\log \Delta + \text{polylog } \log n)$ rounds w.h.p. and $O(\text{polylog } n)$ rounds, respectively. For smaller $\Delta$, the best deterministic algorithm is that of Barenboim and Elkin [6] which takes $O(\Delta + \log^* n)$ rounds.

State-of-the-art randomized MIS algorithms with dependencies on $\Delta$ exploit the locality of the MIS problem, using the "shattering" framework first introduced by Barenboim et al. [8] and was later improved by Ghaffari [54]. This paradigm involves solving the MIS problem for the majority of the nodes in $O(f(\Delta))$ rounds w.h.p. for some function $f$ and then finishing off the remaining nodes with a deterministic network decomposition algorithm. Therefore, the MIS problem and the network decomposition problem are often related. The $O(\text{polylog } n)$-time deterministic decomposition algorithm of Rozhoň and Ghaffari [93] the first to improve over the aforementioned classic result from Panconesi and Srinivasan [91] of $O\left(2^{\sqrt{\log n}}\right)$ rounds. Currently, the best algorithm is that of Ghaffari, Grunau, and Rozhoň [56] which takes $O(\log^5 n)$ rounds.

The MIS problem has also been studied extensively in the CONGEST model when message sizes are restricted to only $O(\log n)$ bits each round. Worth mentioning, the classic algorithm from Luby [80] also works when we restrict the message sizes. The state-of-the-art randomized algorithm for this model stems from the work Ghaffari, Grunau, and Jin [55] which takes $O(\log \Delta \cdot \log \log n + \text{polylog } \log n)$ rounds in expectation. Again, this
latter $O(\text{polylog } \log n)$ term is a product of the usage of a deterministic network decomposition algorithm over the remaining nodes after applying the improved shattering framework of Ghaaffari [54]. While this framework can still be applied in the CONGEST model, the $O(\text{polylog } n)$-time deterministic decomposition algorithm of Rozhoň and Ghaaffari [93] which greatly improved MIS results for the LOCAL model relied on large messages. However, the improved algorithm of Ghaaffari, Grunau, and Rozhoň [56] for the CONGEST model eliminated this dependence.

Lower bounds for the MIS problem have also matured in the past few decades, beginning with the early work of Linial [77, 78] which showed that $\Omega(\log^* n)$ rounds are required to compute an MIS. This bound was elaborated upon by Kuhn, Moscibroda, and Wattenhofer [75] and Daum, Kuhn, and Maus [28] which proved a bound of $\Omega\left(\min\left\{\frac{\log \Delta}{\log \log \Delta}, \sqrt{\frac{\log n}{\log \log n}}\right\}\right)$ expected rounds. Recently, this bound was augmented further by the work of Balliu et al. [4] which showed that $\Omega\left(\min\left\{\Delta, \frac{\log \log n}{\log \log \log n}\right\}\right)$ rounds are required for any randomized MIS algorithm which holds w.h.p. and $\Omega\left(\min\left\{\Delta, \frac{\log n}{\log \log n}\right\}\right)$ rounds for any deterministic algorithm.

Due to extensive research into the MIS problem, many other discoveries have been made. One line of research beginning with Blelloch, Fineman, and Shun [10] studied the performance of the simple greedy MIS algorithm where nodes are added to the MIS according to a random permutation over all nodes. What is interesting about this algorithm is that the random choices by nodes need only occur in a single round, whereas in most probabilistic MIS algorithms re-randomization occurs in every round. While the work of Blelloch, Fineman, and Shun [10] showed this simple strategy computes an MIS in $O\left(\log^2 n\right)$ rounds w.h.p. in $n$, the work of Fischer and Noever [47] tightened this bound, showing that $\Theta(\log n)$ rounds are necessary and sufficient for this algorithm to succeed with high probability.
While the LOCAL and CONGEST models are the most well-studied, there are many others that can be used to describe the application layer setting. One such is the family of models inspired by the massively parallel MapReduce architecture (e.g., [58]). Another model is the node capacitance model where nodes represent electrical devices, limited to the number of bits they can receive each round. Lastly, there is also the family of models denoted knowledge til (KT) models (e.g., [63]). These models are parameterized with a distance value $r$ such that each node begins with knowledge of its $r$-hop neighborhood.

Also relevant to our study of this problem are self-stabilization the dynamic network models. In the self-stabilization model (e.g., [100, 24]), the network starts from an arbitrary configuration and must reach a correct state (w.r.t. the problem of interest) in a finite number of steps. Furthermore, algorithm executions are orchestrated via a scheduler which dictates which nodes may operate in a particular round. Common choices for this scheduler are centralized (a single node participates each round), distributed (all nodes participate each round), and unfair (where any nonempty subset of nodes are selected to participate each round). In the related dynamic network model (e.g., [16]), some number of transient failures occur during or even after the algorithm execution.

In all of the results presented so far, the emphasis is placed entirely on how fast an MIS can be computed in a stable network. Therefore despite the extensive research into this problem, very few works have considered the possibility of node failures during or after computation of the MIS. In this case, the most relevant results we can draw from are those which study the self-stabilizing and dynamic versions of the problem. In the self-stabilization model (e.g., [100, 24]), the network starts from an arbitrary configuration and must reach a correct state (w.r.t. the problem of interest) in a finite number of steps. Furthermore, algorithm executions are orchestrated via a scheduler which dictates which nodes may operate in a particular round. Common choices for this scheduler are centralized (a single node participates each round), distributed (all nodes participate each round), and
unfair (where any nonempty subset of nodes are selected to participate each round). In this model, the work of Turau [100] provides a self-stabilizing MIS algorithm which takes $O(n)$ rounds under the unfair scheduler. Cohen, Pilard, and Sénizergues [24] iterates on this result by providing a self-stabilizing MIS algorithm where the scheduler is fair but the failures can also be byzantine. This algorithm converges in time $O(\Delta \cdot n)$ with high probability.

In the related dynamic network model (e.g., [16]), some number of transient failures occur during or even after the algorithm execution. This work was the first to consider the MIS problem in a dynamic network model, where an MIS is initially computed and then must "recover" from a single topology change (e.g., crash failure) to ensure that a valid MIS exists over the resulting graph. The algorithm the authors provide can recover from any such individual failure in $O(1)$ expected rounds.

The problem of finding a maximal matching (MM) is another cornerstone of distributed graph algorithms research. A maximal matching is a set of edges such that for every node $u$ either (a) $u$ has an edge in the matching or (b) all of $u$’s neighbors have an edge in the matching. In addition to the MM problem being well-studied in its own right, it is also closely tied to the maximal independent set problem, resulting in similar solutions for the two problems. This is because a MM can be constructed by computing an MIS over the edge graph of a network. Here the classic algorithm belongs to Israeli and Itai [68] which achieves a maximal matching in $O(\log n)$ expected rounds. Again, as with the MIS problem, locality can be exploited to achieve $O(\log \Delta + \text{polylog log } n)$ rounds in expectation (see the algorithm of Barenboim et al. [8]). Note that the dependence here on the maximum degree, $\Delta$, is optimal.
4.2 Summary of New Results

Compared to how fundamental these problems are, very little work has been done to study the impact of different types of failures on their construction. We correct for this gap in the existing literature by considering two common definitions of node failures: crash failures and byzantine failures. For each, we define a new computation model and analyze the time needed to build graph structures in these models, as detailed below.

4.2.1 Crash Failures

Consider the following problem. You are provided an MIS in a large network. As might be expected in such a setting, at some point, one or more nodes crash, potentially leaving some of the correct nodes now uncovered by the remaining structure. The challenge is to repair the MIS in the surviving subgraph as efficiently as possible. A simple solution, of course, is to build a new MIS from scratch in this subgraph. Using something like Luby’s algorithm [80], this will require $O(\log n)$ rounds, in expectation, in a network of size of $n$. A somewhat smarter approach is to instead run the MIS algorithm only among the nodes left uncovered by the crashes, as it is easy to see that a new MIS computed on this subset will combine cleanly with the existing MIS. Given $k$ crashes and maximum degree $\Delta$, there are at most $k\Delta$ nodes left uncovered. Therefore, this improved repair strategy requires $O(\log (k\Delta)) = O(\log k + \log \Delta)$ total rounds in expectation.

In both cases, repairing $k$ crashes has a dependence on a global graph parameter (i.e., $n$ or $\Delta$), which might not be desirable when $k$ is small. Here we ask if it is possible to achieve a repair that depended only on $k$, as this would allow us, for sufficiently small $k$, to repair an MIS in time faster than what is required to build an MIS from scratch. As elaborated in the related work discussion above, existing work has already proved that it is possible, under certain circumstances, to repair a single failure (e.g., $k = 1$) in $O(1)$ rounds [73, 16].
The open question focused on what is possible in the more general case where \( k > 1 \) nodes crash concurrently.

To answer this question, in Section 4.3.1 we introduce a general framework that divides computation into two stages: setup and repair. The setup stage is initialized with the nodes already knowing an arbitrary graph structure computed over the initial graph (e.g., an MIS). They can then perform additional precomputation to help prepare for potential future crashes.\(^1\) At the end of this stage, an adversary (oblivious to the computation performed in the setup stage) selects up to \( k \) total nodes to crash. Next begins the repair stage, during which the algorithm attempts to compute a valid MIS w.r.t. the surviving subgraph.

An algorithm in this framework is said to be \((f(n), g(k))-resilient\), for functions \( f \) and \( g \), if its setup stage requires at most \( O(f(n)) \) rounds and its repair stage takes at most \( O(g(k)) \) rounds to terminate. (As formalized in Section 4.3.1, this definition generalizes naturally to probabilistic guarantees.) The key question from before can now be formalized as asking whether their exists such an algorithm for MIS repair that is \((f(n), g(k))-resilient\), for any functions \( f \) and \( g \).

In Section 4.3, we answer this question in the affirmative through a number of results summarized in Table 4.1. First, in Section 4.3.2 we handle the easier case of repairing graph structures in restricted classes of graphs. Then in Section 4.3.3 analyze algorithms for repairing these structures in a broader family of graphs known as bounded independence graphs. Finally, we tackle the most difficult case of general graphs in Section 4.3.4. Here, we succeed in achieving efficient MIS repair by describing and analyzing a simple algorithm which is \( O(\log n, \log k)\)-resilient with high probability. Notice, in the worst case, building an MIS from scratch using any existing MIS algorithm requires \( \Omega(\log n) \) rounds in expectation. This means that adding our setup stage to a standard MIS algorithm does

\(^1\)In some cases, this precomputation might involve discarding the initial MIS, and generating a new structure from scratch used a particular method. In other cases, it might instead mean preparing plans for repairing the provided structure.

163
not increase its expected asymptotic complexity in general. The setup stage replaces the provided MIS with one computed using the classical parallel greedy strategy. The repair stage then runs an additional instance of this simple algorithm on the nodes uncovered by the crashes. Our analysis focuses on the structure of the random MIS generated during the setup stage. It proves the general property that given $k$ failures, chosen independently of an MIS generated using a parallel greedy strategy,\footnote{Formally, we assume the adversary chooses its $k$ nodes that will later crash before the repair stage begins. It is also straightforward to adapt our result to an adaptive adversary under the assumption of a private source of shared randomness.} that the number of nodes uncovered in the MIS by these crashes is in $O(k)$ w.h.p. in $k$. The final bound then follows directly.

Next, in Section 4.3.5 we move beyond the MIS problem to study repairing a maximal matching. Lastly, we approach both problems from the perspective of lower bound results in Section 4.3.6.

4.2.2 Byzantine Failures

Having improved our understanding of the impact of crashes on an MIS, we turn our attention to an even more challenging fault-tolerance setting: byzantine failures. In particular, we assume $t$-local faults, in which each honest (non-byzantine) node has at most $t$ faulty neighbors, for some upper bound parameter $t$. There are many problems that malicious nodes can introduce into the construction of a graph structure. As we elaborate in Section 4.4.1, in this thesis we focus on the ability of byzantine nodes to delay termination.\footnote{For example, in classic algorithm of Luby\cite{luby1986}, nodes make random choices to join the MIS. If two neighboring nodes decide to join simultaneously, then neither is allowed to join in order to ensure independence. Therefore, if a byzantine node, $u$, claims in every round that it has randomly decided to join the MIS but has a neighbor that has done the same, $u$ never joins the MIS while simultaneously preventing any of its honest neighbors from joining either.} With this in mind, we provide, in
<table>
<thead>
<tr>
<th>MIS (general graphs)</th>
<th>(log $n$, log $k$)-resilient</th>
<th>$\Omega\left(\sqrt{\frac{\log log k}{\log log \log k}}\right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM (general graphs)</td>
<td>(1, log $k$)-resilient</td>
<td>$\Omega\left(\frac{\log log k}{\log log \log k}\right)$</td>
</tr>
<tr>
<td>MIS (rooted trees)</td>
<td>(log $^*$ $n$, 1)-resilient</td>
<td>$\Omega(1)$</td>
</tr>
<tr>
<td>MIS (planar graphs)</td>
<td>(log $n$, 1)-resilient</td>
<td>$\Omega(1)$</td>
</tr>
<tr>
<td>MIS (bipartite graphs)</td>
<td>(D, 1)-resilient</td>
<td>$\Omega(1)$</td>
</tr>
<tr>
<td>MIS (bounded independence)</td>
<td>(1, log $^*$ $k$)-resilient</td>
<td>$\Omega(1)$</td>
</tr>
</tbody>
</table>

Table 4.1: Summary of resiliency bounds achieved in this thesis. Note that the algorithms for all restricted graph classes are deterministic. The MM repair algorithm for general graphs consists of a deterministic setup stage followed by a randomized repair stage which holds in expectation, therefore its overall resiliency also holds in expectation. The resiliency of the MIS repair algorithm holds with high probability. Finally, our lower bound results bound the minimum number of rounds required to repair the graph structure in each setting (even if we allow for unlimited setup time).

Section 4.4.1, a natural generalization of the definition of the MIS problem that both prevents trivial independence violations (e.g., from byzantine nodes arbitrarily add themselves to the set), while still allowing for a natural notion of termination.

In Section 4.4.2, we then describe and analyze the random channel MIS algorithm, which has nodes simulate running Luby’s algorithm on $2t$ channels over phases of $2t$ rounds each, where each node can participate on only a single channel per phase. A key observation in analyzing this strategy is that although a byzantine node can choose its simulated channel arbitrarily in a given round, it cannot participate on multiple simulated channels in the round, as this would clearly mark its behavior as faulty. This allows a constant fraction of honest nodes (in expectation) to participate in a given round on a simulated channel free from any byzantine interference. We deploy a careful generalization of the standard analysis from [1] to stitch together the progress made by these honest nodes, ultimately...
proving they terminate with a correct MIS, with respect to our new byzantine-compatible definition (see Section 4.4.1), in $O(t \log n)$ rounds, in expectation.

This result requires three (admittedly, rather strong\(^4\)) restrictions placed on the byzantine nodes to render the MIS problem approachable: (1) a PKI that prevents dishonest nodes from forging messages from honest nodes; (2) a bound on the number of messages each node can send per round; and (3) a restriction on byzantine nodes’ ability to execute *rushing* attacks in which they base their message in a given round on what nearby honest nodes send earlier during that same round. We discuss and justify these decisions in more detail in Section 4.4.1. More generally, our goal with these results is to help initiate the formal study of byzantine-resilient graph structures, not provide the final word on the topic.

### 4.3 Repairing Graph Structures with Crash Failures

In this section we study the problem of repairing maximal independent sets and maximal matchings given crash failures. Once again, we begin by defining the computation model used in Section 4.3.1. We then use this model in Sections 4.3.2 and 4.3.3 to devise upper bounds for repairing these structures in restricted classes of graphs. Next, in Sections 4.3.4 and 4.3.5 we present our strongest upper bound results: providing upper bounds for repairing maximal independent sets and maximal matchings, respectively, in general graphs. Lastly, we round out our exploration by consider lower bounds for this problem in Section 4.3.6.

\(^4\)We thought it appropriate to start with strong restrictions on the byzantine nodes as these results initiate the study of this hard problem. Considering the problem under weaker byzantine constraints is obvious future work.
4.3.1 Model and Problem Definition

We assume the classic broadcast-CONGEST message passing model of communication, in which an undirected graph \( G = (V, E) \) is used to describe the underlying network topology, where the nodes in \( V \) correspond to the computational processes and the edges in \( E \) describe the links between them. In this model, time proceeds in synchronous rounds with integer labels. Each round, each node can broadcast a message of \( O(\log n) \) bits to its neighborhood where recall \( n = |V| \). We further assume communication to be strongly synchronous, meaning that nodes cannot wait to receive a message from their neighbors before sending their own message for the same round (relevant in the context of byzantine failures explored in Section 4.4).

In this section, we focus on solutions to the maximal independent set (MIS) and maximal matching (MM) problems. The standard version of the MIS problem for a given graph \( G = (V, E) \) is defined as follows:

**Definition 4.3.1** (Maximal Independent Set). A subset of nodes \( M \subseteq V \) is a maximal independent with respect to \( G \) if it satisfies the following two properties for every \( u \in V \):

1. \( N^+(u) \cap M \neq \emptyset \) (maximality), and

2. for every \( v \in N(u) \), if \( u \in M \) then \( v \notin M \) (independence).

Furthermore, the MM problem is defined as:

**Definition 4.3.2** (Maximal Matching). A subset of edges \( M \subseteq E \) is a maximal matching with respect to \( G \) if it satisfies one of the following two properties for every \( u \in V \):

1. \( \{u, v\} \in M \) for some \( v \in N(u) \), or

2. for every \( v \in N(u) \), \( \{v, w\} \in M \) for some \( w \in N(v) \).
The model which we introduce to study the impact of crash failures on the computation of graph structures is as follows. An algorithm’s execution is divided into two distinct stages, called setup and repair.

Prior to the setup stage, an initial (arbitrary) instance of the graph structure (e.g., an MIS) is provided to the algorithm. Then, in the setup stage, some initial amount of pre-computation is performed during a time which the graph is assumed to be stable. This computation can leverage the initial MIS provided or, depending on the algorithm, discard it completely (as is the case with our algorithm presented in Section 4.3.4). After the setup stage completes, some set of nodes \( C \) is crashed where \(|C| \leq k\). The set \( C \) is assumed to be chosen by an oblivious adversary which has knowledge of the graph topology but no insight into the computation performed as part of the setup algorithm. One way to think of this setting is one in which the adversary selects the nodes to be crashed (without crashing them) prior to the setup stage’s execution.

After the crashes occur, each node learns which of its neighbors have crashed. Next, the repair stage begins with the goal of computing a valid MIS over the subgraph induced by the surviving nodes (i.e., \( V \setminus C \)). For functions \( f \) and \( g \), we say an algorithm is \((f(n), g(k))-resilient\) if the setup stage takes at most \( O(f(n)) \) rounds and the repair stage takes at most \( O(g(k)) \) rounds. Similarly, we say an algorithm is \((f(n), g(k))-resilient in expectation\) if the expected number of rounds required for the setup stage is \( f(n) \), and the expected number of rounds required for the repair stage is \( g(k) \). Similarly, we say an algorithm is \((f(n), g(k))-resilient with high probability\) if the setup stage terminates in at most \( O(f(n)) \) rounds with high probability in \( n \), and the repair stage terminates in at most \( O(g(k)) \) rounds with high probability in \( k \).
4.3.2 Upper Bounds for Trees, Planar, and Bipartite Graphs

Here we study graph classes in which efficient repair is straightforward to obtain. Our approach focuses on coloring. If a graph can be colored with $c$ colors during the setup phase, then it can be subsequently repaired in $c$ rounds using the following strategy: rotate through the $c$ colors in a fixed order, dedicating one round to each color; during the round for color $i$, if a node $u$ is both uncovered and colored with $i$, it joins the repaired MIS and broadcasts to inform its neighbors that it has joined (allowing its uncovered neighbors to learn they are now covered). Clearly, all uncovered nodes will end up covered by the time the rotation gets through their color. Furthermore, the coloring prevents conflicts among the nodes that join the MIS in a given repair round.

Leveraging this insight we produce the following collection of results for graph classes in which any number of failures can be repaired in $O(1)$ rounds, which is clearly optimal.

The efficiency of the respective setup phases, by contrast, depend on the state of the art coloring strategies for the class in question:

**Theorem 4.3.3.** There exists a MIS algorithm that is:

- $(\log^* n, 1)$-resilient for **rooted trees**.
- $(\log n, 1)$-resilient for **planar graphs**.
- $(D, 1)$-resilient for **bipartite graphs**.

**Proof.** Our solution for all three graph classes deployed the coloring-based repair strategy summarized above. The setup stage complexities therefore depend on the time required to not only build an initial MIS but also obtain a constant coloring.

For rooted trees, the coloring algorithm of Cole and Vishkin [25] provides a constant coloring in $O(\log^* n)$ rounds. This coloring can be used to establish the initial MIS during the setup phase and enable a constant round repair phase.
For both general trees and planar graphs, we turn to a distributed algorithm due to Barenboim and Elkin [7] that can produce a \(\lfloor (2 + \varepsilon) \cdot a(G) \rfloor + 1\)-coloring in \(O(a(G) \log n)\) rounds, where \(a(G)\) is the arboricity of the graph \(G\) in question, and \(\varepsilon > 0\) is an arbitrarily small parameter. Both trees and planar graphs feature constant arboricity, enabling a constant coloring in \(O(\log n)\) rounds. We can therefore run this algorithm along a standard MIS algorithm during the setup phase, yielding both an initial MIS and a constant coloring. The latter can be leveraged in the repair phase to enable efficient repair.

Finally, we turn our attention to bipartite graphs. Though these graphs can be two-colored (by definition), enabling an efficient repair, it is unclear how to obtain such a color efficiently in a distributed manner. In \(O(D)\) rounds, however, it is possible to easily obtain a constant coloring in a bipartite graph by electing a leader, and then using the leader to build a BFS tree, alternating colors by tree level (see [81] for more on standard techniques for combining leader election and a terminating BFS construction).

\[\square\]

### 4.3.3 Upper Bounds for Bounded Independence Graphs

A graph \(G = (V, E)\) is considered to satisfy *bounded independence* if there exists a growth function \(f\), such that for every \(u \in V\) and hop count \(r \geq 1\), the maximum size independent (MaxIS) set of \(u\)'s \(r\)-hop neighborhood (including \(u\) itself) has size upper bounded by \(f(r)\). This property generalizes many practical graph constraints, including unit disk graphs, quasi-unit disk graphs, and unit ball graphs. An important result due to Schneider and Wattenhofer [94] identifies a deterministic algorithm that can compute an MIS in a bounded independence graph of size \(|V| = n\) in only \(O(\log^* n)\) rounds.

Here we explore whether this natural constraint, which is known to speed up the time to construct an MIS from scratch, can also support more efficient repairs. We answer this question affirmatively by describing and analyzing what we call the isolated component MIS algorithm, a deterministic algorithm for which we prove the following resiliency result:
Theorem 4.3.4. The isolated component MIS algorithm is \( (1, \log^* k) \)-resilient.

Given the initial MIS, no additional precomputation time is required. In the repair stage, we deploy a clustering strategy in which uncovered nodes join with uncovered neighbors to form clusters. The nodes in each cluster coordinate to simulate a virtual node associated with the cluster. We then color the resulting virtual node graph in an efficient distributed manner, subsequently cycling through the colors, allowing virtual nodes of that color to safely repair the MIS among its constituent nodes.

Below we start by detailing and analyzing the formation of the clusters at the beginning of the repair stage, before then detailing and analyzing the color-driven set repair that leverages these clusters.

Clustering

Let \( G = (V, E) \) be the original graph. We assume that \( G \) satisfies the bounded independence property, and let \( f \) be the growth function for \( G \) required by the property. We use \( M \) to indicate the initial MIS computed in the setup stage. Let \( C \) be the nodes that crash at the beginning of the repair stage in question where \( |C| \leq k \). Finally, let \( U \) be the nodes uncovered by these crashes (i.e., the nodes not covered by the set \( M \setminus C \)).

By definition, each node in \( U \) neighbors at least one node in \( M \cap C \). The repair stage begins by having each \( u \in U \) arbitrarily select a single leader \( \ell_u \) from its one or more neighbors in \( M \cap C \). We now analyze the properties of this simple distributed clustering.

To do so, we need some additional notation to describe the resulting clusters. We define a subgraph over \( U \) in which we only connect nodes that share a leader. Formally, let \( G' = (U, E') \), where \( E' = \{\{u, v\} \mid \{u, v\} \in E, \ell_u = \ell_v\} \). Next, let \( \mathcal{F} = \{F_1, F_2, \ldots, F_m\} \) be the set of \( m \) connected components of \( G' \). We call each such component a cluster and define \( G_{\mathcal{F}} \).
to be cluster graph induced by $\mathcal{F}$. Formally, $G_{\mathcal{F}} = (V_{\mathcal{F}}, E_{\mathcal{F}})$, where $V_{\mathcal{F}} = \{u_1, u_2, \ldots, u_m\}$, and $E_{\mathcal{F}} = \{\{u_i, u_j\} \mid \{s, t\} \in E \text{ for some } s \in F_i \text{ and } t \in F_j \text{ where } i \neq j\}$.

We can now leverage the bounded independence property of $G$ to argue that the connected components in $G'$ have small diameter:

**Lemma 4.3.5.** For each $F_i \in \mathcal{F}$: the diameter of $F_i$ in $G'$ is no more than $2 f(1) = O(1)$.

*Proof.* Assume for contradiction that some cluster $F_i$ associated with leader $u \in M \cap C$ has a diameter of at least $2 f(1) + 1$. It follows that there are two nodes $s, t \in F_i$ such that the shortest path $p$ between $s$ and $t$ in $F_i$ is of length at least $2 f(1) + 1$. Notice, however, if we consider every other node in $p$, the result is an independent set of size at least $f(1) + 1$. Every node in $F_i$, by definition, neighbors $u$. Therefore, we have identified an independent set of size greater than $f(1)$ in the 1-hop neighborhood of $u$, contradicting our bounded independence assumption. □

Having established each cluster in $\mathcal{F}$ has small diameter, we now prove a pair of useful properties about the cluster graph $G_{\mathcal{F}}$ induced by $\mathcal{F}$.

**Lemma 4.3.6.** The cluster graph $G_{\mathcal{F}}$ contains at most $f(1) \cdot k = O(k)$ nodes and has a maximum degree at most $f(3) = O(1)$.

*Proof.* We begin by using the bounded independence property to prove that the size of $G_{\mathcal{F}}$ is bounded as $O(k)$; formally: $|V_{\mathcal{F}}| \leq f(1) \cdot k$. To do so, fix a cluster leader $v \in M \cap C$. Consider the subset $S_v \subseteq V_{\mathcal{F}}$ of components associated with cluster leader $v$ (i.e., connected components of nodes which neighbor $v$ and selected it as their leader). It follows that $|S_v| \leq f(1)$. If this was not true, we could create an independent set of size strictly greater than $f(1)$ among $v$’s neighbors in $G$ by selecting one node per cluster in $S_v$. This set would be independent since, by definition, if two nodes with the same leader $v$ are in different
connected components in $G'$, then they are not neighbors in $G$. Because there are at most $k$
such leaders, there are at most $f(1) \cdot k = O(k)$ total clusters.

Next, let $\delta$ be the maximum degree of the component graph $G_F$. We aim to bound
$\delta \leq f(3) = O(1)$. To do so, we first note that if two components are neighbors in $G_F$, then
the two leaders of these components are within three hops of each other in $G$. (Observe
that the two components must have different leaders, since two neighboring components
with the same leader would be represented as a single component in $G_F$.) We also note that
no two cluster leaders are neighbors in $G$, as all cluster leaders came from an independent
set $M$ defined on $G$. If some cluster with leader $u$ has $\delta$ neighbors in $G_F$, then there is an
independent set of size $\delta$ in the 3-hop neighborhood of $u$.

From this follows the claim that $\delta \leq f(3)$. \hfill $\square$

**Coloring and Covering**

As described above, the repair stage begins with each uncovered node in $U$ choosing a
leader from $M \cap C$. This implicitly places each node $u \in U$ is some cluster $F_i \in \mathcal{F}$. We
now leverage these clusters to grow the covering set $M \setminus C$ to cover all nodes in $U$.

By Lemma 4.3.5, each cluster $F_i$ has a constant diameter, and by Lemma 4.3.6, $F_i$ is
adjacent to at most a constant number of other clusters. It follows that the nodes in $F_i$ can
work together to simulate one round of the corresponding node $u_i$ in $G_F$ by using a constant
number of real rounds; i.e., by flooding to all nodes in the cluster corresponding to $u_i$, every
message received by $u_i$ (of which there can be at most a constant).\footnote{We are eliding some minor details in this description. Namely, though nodes in $F_i$ can receive
at most $O(1)$ unique messages from neighboring clusters, they might receive each message many
different times. Standard pipeline flood algorithms, however, have each node send each unique mes-
sage exactly once, and will therefore complete the flood of the constant number of messages in a
constant number of rounds, no matter how many times each message shows up in the cluster at first.}

173
The nodes in $U$ can therefore simulate $G_F$ with only a constant-round overhead. The repair stage has the nodes simulate the deterministic coloring algorithm from [94] on $G_F$. This algorithm guarantees a coloring of size $O(\delta)$ in $\log^* n'$ in a bounded independence graph of size $n'$ with maximum degree $\delta$. In our case, $G_F$ has a maximum degree $\delta = O(1)$, which means it satisfies bounded independence (since, of course, the size of the MaxIS over a set of $\delta$ nodes is at most $\delta$). By Lemma 4.3.6, $V_F$ is also of size $O(k)$. It follows that this algorithm, when simulated in $G_F$, will color $G_F$ with $O(1)$ colors in $O(\log^* k)$ rounds.

Let $x$ be a constant size upper bound on the number of colors required to color $G_F$. (It is sufficient, for example, to set $x = f(3) + 1$.) To conclude, the repair stage the nodes cycle through $x$ phases, one for each possible color. During the phase for color $c$, all nodes associated with a cluster $F_i$ assigned color $c$ work together to cover any nodes in $F_i$ that remain uncovered at the start of this phase. This can be accomplished in a constant number of rounds by simply deploying a standard greedy parallel strategy where any node in $F_i$ that is uncovered, and has the highest id among its uncovered neighbors in $F_i$, joins the independent set. The cluster $F_i$ cannot support more than $f(1) = O(1)$ independent set nodes (as this would violate the bounded independence property of the 1-hop neighborhood of the leader of $F_i$), so a constant number of rounds is sufficient to cover the nodes in $F_i$. Notice, adding nodes in $F_i$ cannot create an independence violation with nodes in neighbor components, as by definition neighbor components have a different color and are therefore not participating in the same phase.

Note that while this presentation of the above algorithm is provided to maximize clarity, a more local implementation is possible. Instead of synchronized phases for each color, nodes of a given color can begin executing the greedy parallel strategy described above as soon as they have no neighbors of a preceding color. This implementation provides the same guarantees as the above presentation since we can consider nodes of the color associated with the first phase in isolation until all nodes of this color are covered. We then
consider the nodes of the color associated with the second phase, and so on, noting that any progress these nodes may have made in the meantime can only reduce the number of rounds of the algorithm. The advantage of this presentation is ease of implementation, the formal construction of which is provided in Algorithm 7.

**Algorithm 7** Isolated component MIS repair algorithm (for node \( u \in U \))

1: \( \ell_u \leftarrow v \in N(u) \cap (M \cap C) \) (chosen arbitrarily)
2: 
3: simulate the algorithm from [94] over the cluster graph \( G_C \)
4: let color(\( u \)) be \( u \)'s assigned color given its choice of \( \ell_u \)
5: 
6: while \( u \in U \) do
7: \hspace{1em} if color(\( u \)) \leq \min\{\text{color}(v) \mid v \in N(u) \cap U\} \) then
8: \hspace{2em} if ID(\( u \)) \geq \max\{ID(v) \mid v \in N(u) \cap U, \text{color}(u) = \text{color}(v)\} \) then
9: \hspace{3em} \( M \leftarrow M \cup \{u\} \)
10: \hspace{3em} \( U \leftarrow U \setminus \{u\} \)
11: \hspace{1em} if \( N(u) \cap M \neq \emptyset \) or \( u \in M \) then \( U \leftarrow U \setminus \{u\} \)

At this point, we have covered every node in \( U \) in the \( O(\log^* k + x) = O(\log^* k) \) rounds required to complete the simulated coloring and subsequent repair steps of the repair stage—establishing Theorem 4.3.4.

### 4.3.4 Upper Bound for MIS Repair in General Graphs

In this section, we provide an algorithm for repairing an MIS in general graphs.

**Algorithm**

Here we discuss the greedy MIS repair algorithm, which we show to be \((\log n, \log k)\)-resilient with high probability. A critical piece of the algorithm is the parallel greedy MIS algorithm analyzed in [10, 47]. We begin by providing a description of this algorithm.

In the parallel greedy MIS algorithm, each node begins by choosing a value uniformly at random from the range \( R(n^c) \) for some constant \( c \geq 3 \). We then use these values to induce a
permutation \( \pi \) over \( V \), with ties broken by arbitrary node identifiers. Note that, when \( c \geq 3 \), this process yields a permutation over \( V \) chosen uniformly at random with high probability in \( n \).\(^6\) In each round, for each uncovered node \( u \) with no uncovered neighbor \( v \) for which \( v \) selected a value smaller than \( u \)'s value, \( u \) joins the MIS. This node and all of its neighbors are now considered covered. This continues until eventually every node is covered.

In the setup stage of our algorithm, we execute an instance of the parallel greedy MIS algorithm as described above. For the following, let \( M_\pi \) denote the MIS created by this algorithm. Then, in the repair stage, each node in \( M_\pi \setminus C \) forms the basis of the MIS, where again \( M_\pi \) is the MIS computed in the setup stage. Then, any nodes which are left uncovered execute an additional instance of the parallel greedy MIS algorithm over the uncovered subgraph.

**Complexity Analysis**

In the following theorem, we formalize our claim regarding the time complexity of the above algorithm. We then introduce some useful notation before outlining the analytical strategy deployed in our proof.

**Theorem 4.3.7.** The greedy MIS repair algorithm is \((\log n, \log k)\)-resilient with high probability, where \( n \) is the network size and \( k \) is an upper bound on the number of crash failures.

Let \( \pi \) be the fixed uniform random permutation over \( V \) induced by the values selected by the nodes in the setup instance of the parallel greedy MIS algorithm. Let \( M_{\pi,i} \) for round \( i > 0 \) be the nodes added to the MIS via the algorithm using \( \pi \) at the beginning of round \( i \). Furthermore, let \( U_{\pi,i} = \{ u \in V \mid N^+(u) \cap M_{\pi,i} = \emptyset \} \) be the set of uncovered nodes according to \( \pi \).\(^6\) Fix \( c = 3 \). Note that the probability that a pair of nodes choose the same value is at most \( 1/n^c = 1/n^3 \). A union bound over all at most \( n^2 \) pairs of nodes yields that all nodes choose unique values with probability at least \( 1 - 1/n \). Conditioned on this event, note that all resulting permutations of \( V \) are equally likely. Therefore, this process is equivalent to choosing a unique permutation \( \pi \) over \( V \) uniformly at random with probability at least \( 1 - 1/n \).
to \( \pi \) at the beginning of round \( i \). Note that, in each round \( i > 0 \) of the algorithm, each node \( u \in U_{\pi,i} \) adds itself to \( M_{\pi,i+1} \) if \( \pi(u) = \min\{\pi(v) \mid v \in N^+(u) \cap U_{\pi,i}\} \).

Fix the set of crashed nodes \( C \subseteq V \) where \( |C| \leq k \). We now analyze the MIS, \( M_\pi \), created by the sequential greedy MIS algorithm according to \( \pi \). Note the sequential variant of this algorithm simply considers each node one at a time instead of each node in parallel which is the local minima of its uncovered neighborhood according to \( \pi \). In greater detail, if \( \pi^{-1}(i) \in V \) is the node considered in step \( i > 0 \) of the sequential greedy algorithm, \( \pi^{-1}(i) \) is added to \( M_\pi \) if and only if it has no neighbors in \( M_\pi \). This continues for \( n \) steps until each node is considered. Despite this distinction, both the parallel and sequential greedy algorithms compute identical sets, \( M_\pi \), given the same permutation, \( \pi \) [10, 47].

For a node \( u \in V \), let \( N_{\pi,i}(u) = N(u) \cap U_{\pi,i} \) be \( u \)'s uncovered neighbors according to \( \pi \) at the beginning of step \( i \in \mathbb{N} \) where \( d_{\pi,i} = |N_{\pi,i}(u)| \). Next, define the shorthand notation \( d_{\pi}(u) = d_{\pi,\pi(u)}(u) \) and recall that \( M_\pi \) is the MIS produced by the sequential algorithm according to \( \pi \). We can then define

\[
d_{\pi}^*(u) = \begin{cases} 
  d_{\pi}(u) & \text{if } u \in M_\pi \\
  0 & \text{if } u \notin M_\pi.
\end{cases} \quad (4.1)
\]

While the significance of capturing \( d_{\pi}^*(u) \) apart from \( d_{\pi}(u) \) will be made clear in Lemma 4.3.8, intuitively, it signifies the number of \( u \)'s neighbors which are covered for the first time in step \( \pi(u) \) by the sequential greedy algorithm given permutation \( \pi \). Lastly, let

\[
V_\pi(C) = \{ u \in V \setminus C \mid (N^+(u) \cap M_\pi) \subseteq C \} \quad (4.2)
\]

be the surviving node set left uncovered by the crash failures in \( C \) and let \( n_\pi(C) = |V_\pi(C)| \).
The fact that the setup stage of our algorithm completes in $O(\log n)$ expected rounds follows directly from the main result of Fischer and Noever [47]. Therefore, our main interest lies in bounding the time of the repair algorithm defined over the random choice of the permutation $\pi$ given any set of crash failures $C \subseteq V$.

We begin by bounding the number of uncovered nodes as a function of the crash failures. Since if every node is covered by $M_\pi$, any node which is uncovered must only neighbor nodes in $M_\pi$ that are also in $C$. In Lemma 4.3.8 we show that the only nodes that are uncovered are those $u \in V$ such that for some node $v \in C$, $u$ was uncovered in step $\pi(v)$ and covered in step $\pi(v) + 1$. Note that the number of such nodes $u \in V$ for which this is true is given by the value of $d^*_\pi(v)$.

The rest of the proof is then devoted to the much more difficult task of upper bounding $d^*_\pi(u)$ for some $u \in V$ given a random permutation $\pi$. We start in Lemma 4.3.10 by adapting a result from [10] to better fit our setting. Namely, we show that given $\pi(u) = i$ and some $d \geq 1$, the probability that $d^*_\pi(u) \geq d$ is $O(\exp(-di/n))$. This bound is then simplified in Lemma 4.3.11 which removes the bound’s dependencies on $i$ and $n$, showing that the probability a node $u \in V$ is responsible for covering at least $d$ nodes is at most $1/d$.

Pulling the pieces together thus far, we then show in Lemma 4.3.12 that the probability that more than $n_0$ nodes are left uncovered is at most $k^2/n_0$ for any $n_0 \geq k$. We then prove our main theorem by leveraging the fact that the backup algorithm takes at most $O(\log n_\pi(C))$ rounds with high probability in $k$.

**Lemma 4.3.8.** For any $C \subseteq V$, $n_\pi(C) \leq \sum_{u \in C} d^*_\pi(u)$.

*Proof.* Consider the formation of $M_\pi$ under the sequential greedy MIS algorithm. When a node $u$ is added to $M_\pi$, it covers all nodes in $N_{\pi,\pi(u)}(u)$, as these are $u$’s neighbors which are not currently covered by the set $M_{\pi,\pi(u)}$. If $u$ fails, these are the only nodes that can become uncovered by $u$ itself, as every other node in its neighborhood is covered by a node earlier
in $\pi$ and would need that node to also crash in order for it to be uncovered. Therefore, the number of nodes that can be uncovered by only crashing node $u$ is at most $d^*_\pi(u)$.

Iterate through each node in $M_\pi \cap C$ in order, according to $\pi$, and consider the first such node $u_1$. Again, following the reasoning above, at most $d^*_\pi(u_1)$ nodes can become uncovered as a result. Considering the next node in $M_\pi \cap C$, which we denote $u_2$, note that each neighbor of $u_2$ which becomes uncovered by this crash must either have been first covered by $u_1$ or first covered by $u_2$. Therefore, each uncovered node accumulated thus far is either included in the count $d^*_\pi(u_1)$ or $d^*_\pi(u_2)$. Continuing this reasoning for all nodes in $M_\pi \cap C$ then yields $n_\pi(M_\pi \cap C) = n_\pi(C) \leq \sum_{u \in C} d^*_\pi(u)$. □

Since we now know that $n_\pi(C) \leq \sum_{u \in C} d^*_\pi(u)$, we turn our attention to bounding this sum $\sum_{u \in C} d^*_\pi(u)$. We begin by bounding the probability that $d^*_\pi(u)$ for some node $u \in V$ is "too" large. We first argue this claim informally before proving it in Lemma 4.3.11 using a generalization of a result first proven in [10].

**Claim 4.3.9.** For any $u \in V$ and $d \geq 1$, $\Pr[d^*_\pi(u) \geq d] \leq 1/d$.

**Proof.** Consider an execution of the sequential variant of the greedy MIS algorithm according to our random permutation $\pi$ and fix a node $u \in V$ such that $d^*_\pi(u) \geq d$ for some $d \geq 1$. Note that, not only does this require $u$ to be added to $M_\pi$ in some step $i > 0$ of the algorithm, by definition of $d^*_\pi(u)$, this means that $u$ has at least $d$ neighbors uncovered by $M_\pi$ at the beginning of this step.

Since these neighbors are all eligible to join $M_\pi$ themselves, the probability that $u$ in particular is selected to join $M_\pi$ in step $i$ is at most $1/d$. Therefore, the claim follows. □

We next provide a more formal proof of Claim 4.3.9.

**Lemma 4.3.10.** For any $u \in V$, $d \geq 1$, and $i \in R(n)$, $\Pr[d^*_\pi(u) \geq d \mid \pi(u) = i] \leq \exp(-di/n)$. 179
**Proof.** This proof follows a similar strategy to that of Lemma 3.1 of [10] but is generalized for our setting. Fix a node \( u \in V \). For the sake of our analysis, we iterate through each index \( i' \leq i \) of the sequential variant of the greedy MIS algorithm according to our random permutation \( \pi \), recognizing that each step is equivalent to selecting a random node \( v \) (without replacement) and adding \( v \) to \( M_{\pi,i'} \) if \( v \in U_{\pi,i'} \).

First assume \( d_\pi(u) \geq d \), as \( d_\pi(u) < d \) would imply that \( d'_\pi(u) < d \) since \( d'_\pi(u) \leq d_\pi(u) \). This would mean that \( \Pr[d'_\pi(u) \geq d] = 0 \), satisfying the lemma trivially. Given that \( u \) has \( d_{\pi,i'}(u) \) uncovered neighbors at step \( i' \), the probability that \( u \) has a neighbor join \( M_{\pi,i'} \) is at least \( d_{\pi,i'}(u)/(n-i') \geq d_{\pi,i'}(u)/n \geq d_\pi(u)/n \geq d/n \). Therefore, the probability that no neighbor of \( u \) joins in step \( i' \) is at most \( 1 - d/n \). The probability that this occurs in every step \( i' \leq i \) is then at most \( (1 - d/n)^i \leq \exp(-di/n) \). Since \( d'_\pi(u) \geq d \) implies that \( u \in M_{\pi} \) (since we assume \( d > 0 \)) and \( u \in M_{\pi} \) implies that \( u \) never has a neighbor join \( M_{\pi} \) in steps \( i' \leq i \), we have that \( \Pr[d'_\pi(u) \geq d \mid \pi(u) = i] \leq \exp(-di/n) \). \( \square \)

The above bound is useful, but the dependencies on \( n \) and \( i \) are undesirable, as we are only really interested in the dependency on \( d \). Therefore, we next isolate this dependency.

**Lemma 4.3.11.** For any \( u \in V \) and \( d \geq 1 \), \( \Pr[d'_\pi(u) \geq d] \leq 1/d \).

**Proof.** Fix \( u \in V \) and \( d \geq 1 \). Since \( \Pr[d'_\pi(u) \geq d] \leq \sum_{i=1}^{n} \Pr[d'_\pi(u) \geq d \mid \pi(u) = i] \cdot \Pr[\pi(u) = i] \) and from Lemma 4.3.10 we have that \( \Pr[d'_\pi(u) \geq d \mid \pi(u) = i] \leq \exp(-di/n) \) we get that \( \Pr[d'_\pi(u) \geq d] \leq \sum_{i=1}^{n} \exp(-di/n) \cdot \Pr[\pi(u) = i] \). Since \( \Pr[\pi(u) = i] = 1/n \) for all \( i \in R(n) \) we have \( \Pr[d'_\pi(u) \geq d] \leq (1/n) \sum_{i=1}^{n} \exp(-di/n) \). We can then treat the sum \( \sum_{i=1}^{n} \exp(-di/n) \) as the sum a geometric series with a common ratio of \( r = \exp(-d/n) \)
(noting that $\sum_{i=0}^n r^i \leq 1/(1-r)$ for $r < 1$),

\[
\sum_{i=1}^n \exp(-di/n) = \sum_{i=0}^n \exp(-di/n) - 1 \tag{4.3}
\]

\[
\leq \frac{1}{1 - \frac{1}{\exp(d/n)}} - 1 \tag{4.4}
\]

\[
= \frac{\exp(d/n)}{\exp(d/n) - 1} - 1 \tag{4.5}
\]

\[
= \frac{1}{\exp(d/n) - 1}. \tag{4.6}
\]

If we then note that from Lemma 1.3.7 that $\exp(d/n) \geq 1 + d/n$, we can simplify further:

\[
\Pr[d^*_\pi(u) \geq d] \leq (1/n) \sum_{i=1}^n \exp(-di/n) \tag{4.7}
\]

\[
\leq \frac{1}{n} \cdot \frac{1}{\exp(d/n) - 1} \tag{4.8}
\]

\[
\leq \frac{1}{n} \cdot \frac{1}{1 + d/n - 1} \tag{4.9}
\]

\[
= (1/n) \cdot (n/d) \tag{4.10}
\]

\[
= 1/d. \tag{4.11}
\]

Therefore, $\Pr[d^*_\pi(u) \geq d] \leq 1/d$. \hfill \Box

Now that we have derived a simplified upper bound for $d^*_\pi(u)$ for any $u \in V$, we use this to derive an upper bound on the number of uncovered nodes $n_\pi(C)$.

**Lemma 4.3.12.** For any $n_0$ where $n_0 \geq k$, $\Pr[n_\pi(C) > n_0] \leq k^2/n_0$.

**Proof.** From Lemma 4.3.11 we have that for any $u \in V$ and $d \geq 1$ that $\Pr[d^*_\pi(u) \geq d] \leq 1/d$. Since $n_0 \geq k$ means $n_0/k \geq 1$, we can substitute $d = n_0/k$ into the result of Lemma 4.3.11 to get that $\Pr[d^*_\pi(u) \geq n_0/k] \leq k/n_0$. Via union bound over the at most $k$ nodes in $C$, we then have that the probability $d^*_\pi(u) \geq n_0/k$ for any $u \in C$ is at most $|C| \cdot (k/n_0) \leq k^2/n_0$. Therefore
$d^*_u \leq n_0/k$ for all $u \in C$ with probability at least $1 - k^2/n_0$. Note that if $d^*_u \leq n_0/k$ for all $u \in C$ then $\sum_{u \in C} d^*_u \leq \sum_{u \in C} n_0/k = |C| \cdot (n_0/k) \leq n_0$.

Furthermore, from Lemma 4.3.8 we then have that $\sum_{u \in C} d^*_u \leq n_0$ implies that $n_\pi(C) \leq n_0$. Therefore we have that $\Pr[n_\pi(C) > n_0] \leq k^2/n_0$, completing the lemma. □

We now have the required support to prove our main theorem.

Proof. (of Theorem 4.3.7) Again, the $O(\log n)$ time complexity of the setup stage, which holds with high probability in $n$, is given by the main result of [47]. Therefore, we now turn our attention to the repair stage, showing that its time complexity is in $O(\log k)$ with high probability in $k$.

First, Lemma 4.3.12 gives us that $\Pr[n_\pi(C) \geq k^3] \leq k^2/k^3 = 1/k$. Therefore, assume $n_\pi(C) < k^3$. Again, from [47] we have that the parallel greedy MIS algorithm executed over a network, with size upper bounded by $n' > 0$, terminates in $O(\log n')$ rounds with high probability in $n'$. Therefore, given that $n_\pi(C) < k^3$, this algorithm computes a valid MIS over the uncovered subgraph in at most $O(\log k)$ rounds with probability at least $1 - c/k^d$ for some constants $c, d \geq 1$. Therefore, the probability that the backup algorithm fails to compute an MIS in at most $O(\log k)$ rounds, given that $n_\pi(C) < k^3$, is at most $c/k^d$.

Now, via union bound over the events that either $n_\pi(C) \geq k^3$ or that the backup algorithm takes longer than $O(\log k)$ rounds given that $n_\pi(C) < k^3$, the probability that the repair takes longer than $O(\log k)$ rounds is at most $1/k + c/k^d \leq 2c/k$. Therefore, with probability at least $1 - 2c/k$, the repair algorithm computes a valid MIS after at most $O(\log k)$ rounds. □

4.3.5 Upper Bound for MM Repair in General Graphs

We now turn our attention to computing resilient maximal matchings. Unlike the case for maximal independent sets in the previous section, we show that we can repair matching solutions in time $O(\log k)$ without any additional precomputation time beyond the time.
required to compute the original matching. Also the repair time holds in expectation, making our algorithm \((1, \log k)\)-resilient in expectation.

**Algorithm**

Let \(M \subseteq E\) denote the maximal matching computed as part of the setup stage, with node set \(V_M = \bigcup_{(u,v) \in M} \{u, v\}\) defined with respect to our original graph \(G\). Again let \(C\) be the set of crashed nodes and \(V' = V \setminus C\) be the surviving nodes. Let \(G' = G(V') = (V', E')\) be the subgraph these nodes induce.

We denote the surviving edges from the original matching \(M' = M \cap E'\) and let \(V'_M = \bigcup_{(u,v) \in M'} \{u, v\}\) be the nodes incident to \(M'\). Let \(U = \{u \in V' \mid \{u, v\} \in M \setminus M' \text{ (for some node } v \in C\}\}\) be the set of surviving nodes incident to the matching \(M\) whose neighbor in the matching has failed. Notice that \(|U| \leq k\) as well. Finally, let \(E'' = \{\{u, v\} \in E' \mid u \in U, u, v \notin V'_M\}\) be the set surviving edges with at least one endpoint in \(U\) and no endpoints in the surviving matching. Let \(G'' = (V'', E'')\) be the bipartite subgraph (with partitions \(U\) and \(V'_M\)) of \(G'\) induced by these edges. During the repair stage, we execute the maximal matching algorithm from [68] only over the edges of \(G''\). For completeness, we briefly describe this existing algorithm.

In each round, each node chooses an edge uniformly at random and directs it outward. Then, each node chooses an adjacent incoming edge uniformly at random. Let \(S''\) be the undirected subgraph induced by edges chosen in both of the preceding steps. Finally, each node selects a random adjacent edge from \(S''\) and all edges \(\{u, v\}\) chosen by both endpoints \(u\) and \(v\) are added to the set \(M''\). Nodes \(u\) and \(v\) are then removed from \(U\) and \(V''\) prior to the next round of the algorithm. The algorithm continues until no more edges in \(E''\) remain. The final solution is the union of these edges with the surviving edges from the original matching (i.e., \(M'' \cup M'\)) which comprises a maximal matching over \(G'\).
Note that the algorithm from [68] is designed for the broadcast-CONGEST model and therefore can be implemented directly in our crash failure model.

**Correctness Analysis**

Clearly $M''$ generated by the algorithm from [68] is a maximal matching over $G''$. What remains to be proven is that adding this matching to the edges in $M'$ creates an MM over the entire graph $G'$. We prove this first.

**Lemma 4.3.13.** $M' \cup M''$ is a maximal matching over $G'$.

**Proof.** Assume for the sake of contradiction that $M' \cup M''$ is not an MM with respect to $G'$ such that an edge $\{u,v\} \in E'$ can be added. That is to say that neither $u$ nor $v$ are the endpoint of an edge in $M' \cup M''$.

First consider the case that $\{u,v\} \in M$ where $M$ is the original matching over $G$. If $\{u,v\} \not\in M'$, then by definition either $u \not\in V'$ or $v \not\in V'$. Therefore, $\{u,v\} \not\in E'$ which means $\{u,v\}$ is not eligible to be added, a contradiction. Therefore, assume $\{u,v\} \not\in M$. Since $\{u,v\} \in E' \subseteq E$, either $u$ or $v$ (but not both) must be the endpoint of an edge in $M$ (otherwise this would contradict the fact that $M$ is maximal). Without loss of generality, let $u$ be this endpoint and let $\{u,w\}$ be the edge incident to $u$ in $M$. Since by assumption $\{u,w\} \not\in M'$ and $u \in V'$, it must be the case that $w \not\in V'$. This implies $u \in U$.

Now, if $v \in V'_M$ then of course $\{u,v\}$ is not eligible to be added to the matching. Therefore, assume $v \not\in V'_M$ which by definition implies that $\{u,v\} \in E''$. However, if neither $u$ nor $v$ are an endpoint of an edge in $M''$, this contradicts the maximality of the matching $M''$ over $G''$. Therefore, $\{u,v\}$ cannot be added to the matching and $M' \cup M''$ must be maximal with respect to $G'$.
Complexity Analysis

We now analyze the running time of the above algorithm, proving that it produces a maximal matching over $G'$ in $O(\log k)$ rounds in expectation.

**Theorem 4.3.14.** The maximal matching algorithm is $(1, \log k)$-resilient in expectation.

We first show that any node in $U$ with $\Omega(k^2)$ neighbors is likely to be added to the new matching in a single round, allowing us to later bound the number of edges remaining in $G''$ after $\log k$ rounds.

**Lemma 4.3.15.** Fix any node $u \in U$ with more than $4k^2$ neighbors in $G''$. With probability at least $1/64$, an edge incident to $u$ is added to $M''$ in one round.

**Proof.** Let $u \in U$ be a node with more than $4k^2$ neighbors in $G''$. First, in the algorithm from [68] over $G''$, each of $u$’s neighbors chooses a random neighbor in $U$ to propose an edge to. Since there are at most $k$ nodes in $U$, the probability that any neighbor of $u$ selects $u$ is at least $1/k$.

Since $u$ has more than $4k^2$ neighbors, the expected number of incoming edges on $u$ is at least $4k$. We can therefore apply the Chernoff bound from Theorem 1.3.4 for $\mu = 4k$ and $\varepsilon = 1/2$ to show that the probability that $u$ is incident to at least $2k$ incoming edges is at least $1 - \exp(-k/2)$ (which is at least $1/16$ for any $k \geq 1$). Next, as part of the algorithm, $u$ chooses an incoming edge uniformly at random from those directed towards it. Notice that out of all the neighbors of $u$ that direct an edge towards $u$, at most $k$ of these neighbors can have an incoming edge from any other node in $U$ (since $|U| \leq k$).

Therefore, conditioned on the event that at least $2k$ neighbors choose $u$, the probability that $u$ chooses an incoming edge from a neighbor which was not selected by any node in $U$ (in the first step of the algorithm) is at least $1/2$. Conditioned on this event occurring, denote this neighbor $v$. Since $v$ has no incoming edges, it chooses $\{u, v\}$ with probability
1. Now, per our algorithm, \( u \) has at most 2 incident edges in \( S'' \) (the edge \( \{u, v\} \) which we just identified and potentially some other edge \( \{u, w\} \) for \( w \neq v \), initiated by \( u \) in the first step of the algorithm). Therefore, the probability that \( u \) selects \( \{u, v\} \) in the final step of the algorithm is at least \( 1/2 \).

From the above, we have that at least \( 2k \) neighbors of \( u \) choose \( u \) in the first step of the algorithm, \( u \) chooses a neighbor \( v \) in the second step of the algorithm which has not been selected by any node in \( U \) in the first step of the algorithm, and \( u \) and \( v \) choose edge \( \{u, v\} \) in the third step with probability at least \( (1/16) \cdot (1/2) \cdot (1/2) = 1/64 \). Since this sequence of events is sufficient for \( u \) to be incident to an edge added to the matching, this event also occurs with probability at least \( 1/64 \). Therefore, in any round of the algorithm from [68], if \( u \) has at least \( 4k^2 \) neighbors, it is removed with probability at least \( 1/64 \). □

\textbf{Proof.} (of Theorem 4.3.14) From Lemma 4.3.15, we have that any node with more than \( 4k^2 \) neighbors in \( G'' \) is added to the matching \( M'' \) in one round with at least constant probability. Therefore, the expected number of such nodes in \( U \) removed is at least a constant fraction of these nodes. Since \( |U| \leq k \) it follows that \( O(\log k) \) expected rounds suffice until all remaining nodes in \( U \) have fewer than \( 4k^2 \) neighbors in \( G'' \).

Finally, once no node in \( U \) has more than \( 4k^2 \) neighbors in \( G'' \), the total number of edges in \( G'' \) is at most \( 4k^4 \). This follows because there at most \( k \) nodes in \( U \), each such node has at most \( 4k^2 \) neighbors, and each of these neighbors can have at most \( k \) edges back to \( U \). Since the running time of the matching algorithm from [68] is \( O(\log |E|) \) in expectation, it follows that an additional \( O(\log k^4) = O(\log k) \) expected rounds suffice to produce a maximal matching.

This yields \( O(\log k) \) expected rounds for the repair algorithm as required. □
4.3.6 Lower Bounds for Repairing Graph Structures

In Sections 4.3.4 and 4.3.5, we proved the existence of algorithms that can repair an MIS or MM with a time complexity that depends only on the number of crashes, $k$. A natural follow up question is whether we can do even better. Perhaps, for example, if given enough time for precomputation we can repair the structures in $O(1)$ rounds, for at least some super-constant values of $k$. Here we dash these hopes by proving that no algorithm can guarantee to repair an MIS or MM in $o\left(\frac{\log \log k}{\log \log \log k}\right)$ rounds. Put another way, any repair time expressed with respect to $k$ must grow with $k$.

Our argument, which we formalize below, is based on reduction. Given an efficient repair solution, we use it to create a novel MIS or MM algorithm that works in some graph $G$ by simulating the repair on a universal topology, that begins as a very large graph (with respect to $G$), but then, after many simulated crashes, enter the repair stage with a remaining subgraph that is isomorphic to $G$. A very fast repair solution therefore yields a fast MIS or MM algorithm.

The Universal Topology

Fix a vertex set $V = \{u_1, u_2, ..., u_n\}$ of size $n > 1$. We define a universal topology for $V$, denoted $\mathcal{G}(V) = (V^*, E^*)$, that embeds every possible graph over $V$ in a manner useful to the lower bound argument that follows. To do so, for every $u_i \in V$, and every possible adjacency list $L \subseteq V \setminus \{u_i\}$ for $u_i$, we add a node $u_{i,L}$ to $V^*$. Slightly more tricky is the construction of $E^*$. For each pair $u_{i,L}, u_{i',L'} \in V^*$, we add the edge $\{u_{i,L}, u_{i',L'}\}$ to $E^*$ if and only if: $u_i \in L'$ and $u_{i'} \in L$.

The key property of $\mathcal{G}(V)$ is that it embeds every graph defined over $V$. Formally, fix any such graph $G = (V, E)$. For each $u_i \in V$, keep $u_{i,N_G(u_i)}$ in $V^*$, but remove every $u_{i,L \not\in N_G(u_i)}$, where we define $N_G(u_i)$ in the standard way to be the neighbors of $u_i$ in $G$. We use the
notation $\mathcal{G}(V, E)$ to describe the subgraph of $\mathcal{G}(V)$ the remains after these node removals.

We argue the following:

**Lemma 4.3.16.** Fix any node set $V$ with $|V| > 1$ and $G = (V, E)$. It follows that $\mathcal{G}(V, E)$ is isomorphic to $G$.

**Proof.** Fix any $u_i \in V$. By the definition of $\mathcal{G}(V, E)$, the only node of the form $u_{i,*}$ that remains in this graph is $u_{i,N_G(u_i)}$. To establish a bijection $b$ between $G$ and $\mathcal{G}(V, E)$, therefore, we can define for each $u_i \in V$ that $b(u_i) = u_{i,N_G(u_i)}$. To prove that this bijection satisfies the adjacency requirements, first fix some $u_i, u_j \in V$ such that $\{u_i, u_j\} \in E$. By assumption, $u_i \in N_G(u_j)$ and $u_j \in N_G(u_i)$, therefore, by construction, $\mathcal{G}(V)$ contains an edge between $b(u_i) = u_{i,N_G(u_i)}$ and $b(u_j) = u_{j,N_G(u_j)}$. Next, fix some $u_i, u_j \in V$ such that $\{u_i, u_j\} \notin E$. In this case, there cannot be an edge between $b(u_i) = u_{i,N_G(u_i)}$ and $b(u_j) = u_{j,N_G(u_j)}$, as by assumption, $u_i$ and $u_j$ are not neighbors in $G$. □

**Reduction Argument**

We now deploy a reduction argument to connect repairing $\mathcal{G}(V, E)$ to building a structure from scratch in $G = (V, E)$. When combined with existing lower bounds we arrive at lower bounds for repairs.

**Lemma 4.3.17.** Assume $\mathcal{A}$ is an $(f(n), g(k))$-resilient repair algorithm. It follows that there exists an algorithm $\mathcal{A}'$ that can construct this same structure in $O\left(g(2^n + \log n)\right)$ rounds in graphs of size $n$. If the resiliency of $\mathcal{A}$ holds with probabilistic constraints, then the same constraints hold for the correctness of $\mathcal{A}'$.

**Proof.** Fix some algorithm $\mathcal{A}$ and complexity function $g(k)$ as specified by the lemma statement. We use $\mathcal{A}$ to construct $\mathcal{A}'$. In more detail, let $G = (V, E)$ be the graph topology on which we are executing $\mathcal{A}'$. The main strategy of this algorithm is to simulate running
$\mathcal{A}$ in $\mathcal{G}(V)$, where the repair stage starts with exactly the crashes that reduce the universal topology to $\mathcal{G}(V, E)$. Because $\mathcal{G}(V, E)$ is the same for all edge sets that $\mathcal{A}'$ might encounter, the simulation of the setup stage of $\mathcal{A}$ can be done offline and in advance, and have the results hardwired into the definition of $\mathcal{A}'$. The magnitude of $f(n)$, therefore, is immaterial to the time complexity of $\mathcal{A}'$.

Simulating the repair is slightly more involved. By Lemma 4.3.16, we know $\mathcal{G}(V, E)$ is isomorphic to $G = (V, E)$. We can have each $u_i \in V$ simulate $b(u_i)$, where $b$ is the bijection provided by the isomorphism. Each $u_i$ knows can learn the identify to the node it is simulating by simply learning its neighbors in $G$, as we earlier defined $b(u_i) = u_{i, Ng(u_i)}$. Accordingly, we put aside one setup round for the simulation in which each node in $G$ broadcasts its id so that each node can learn which node from $\mathcal{G}(V, E)$ it is simulating.

We must also, however, simulate each $b(u_i)$ receiving a crash notification from every neighbor in $\mathcal{G}(V)$ that was crashed to create $\mathcal{G}(V, E)$. To accommodate this behavior, we introduce a second setup round in which each $u_i \in V$ broadcasts its simulated identity $b(u_i)$ to its neighbors. Fix two neighbors $u_i$ and $u_j$ in $G$. By the end of the second round $u_i$ has learned $u_j$ and $Ng(u_j)$. It can now simulate $b(u_i) = u_{i,Ng(u_j)}$ receiving a crash notification from every node of the form $u_{j,L}$, where $L \neq Ng(u_j)$ and $u_i \in L$. If $u_i$ does this for every neighbor $u_j$, it has successfully simulated all failure notifications that $b(u_i)$ would have received as the universal topology was reduced to $\mathcal{G}(V, E)$ due to crashes.

We now proceed with our simulation of $\mathcal{A}$ by the nodes of $G$. By Lemma 4.3.16, the graph in which they are repairing the graph structure is isomorphic to $G$. Therefore, when the simulated repair stage completes, there is a valid version of the structure in $G$.

To arrive at the stated time complexity, we note that $\mathcal{A}$ has a time complexity in $O(g(k))$, where $k$ is the number of node failures. If $n = |V|$, then our simulation of $\mathcal{A}$ crashes slightly less than $2^n$ nodes in the universal topology for each of the $n$ nodes in $V$, leading to a failure.
bound of \( k < n2^n = 2^{n+\log n}. \) Therefore, \( \mathcal{A} \) requires \( O(g(k)) = O(g(n2^n = 2^{n+\log n})) \) rounds, as claimed. \( \square \)

To arrive at concrete lower bounds for the fastest achievable repair time for a given graph structuring problem, it is now sufficient to identify a sufficiently small \( g(k) \) such that \( g(2^{n+\log n}) \) violates an existing lower bound for that problem in graphs where \( \Delta = \Omega(n) \) (e.g., \( G(V) \)). For example, Kuhn et al. [75] prove that randomized solutions to both the MIS and MM problems require \( \Omega\left(\sqrt{\frac{\log n}{\log \log n}}\right) \) rounds to solve with non-trivial probability. Combined with Lemma 4.3.17, we get the following corollary:

**Corollary 4.3.18.** Fix a randomized \((f(n), g(k))\)-resilient MIS or MM algorithm which succeeds with non-trivial probability. It follows that

\[
g(k) = \Omega\left(\sqrt{\frac{\log \log k}{\log \log \log k}}\right). \tag{4.12}
\]

Similarly, the work of Balliu et al. [4] shows that for deterministic algorithms, \( \Omega\left(\frac{\log n}{\log \log n}\right) \) rounds are required. This lets us derive a repair lower bound for deterministic algorithms as well. Namely:

**Corollary 4.3.19.** Fix a deterministic \((f(n), g(k))\)-resilient MIS or MM algorithm. It follows that

\[
g(k) = \Omega\left(\frac{\log \log k}{\log \log \log k}\right). \tag{4.13}
\]
4.4 The Byzantine Maximal Independent Set Problem

We now introduce the MIS problem with byzantine failures, which we refer to as the byzantine MIS problem. First, in Section 4.4.1 we discuss and formulate the computation model and problem definition. We then use this model in Section 4.4.2 to construct an algorithm which succeeds in the task we define. Lastly, in Section 4.4.3 we compare this bound against what is possible in this model by proving a lower bound.

4.4.1 Model and Problem Definition

In the setting in which we consider byzantine failures, we will call a node suffering from a byzantine failure dishonest, and non-faulty nodes honest. For a given execution, we denote the set of honest nodes $V_h \subseteq V$, and the set of dishonest nodes $V_f = V \setminus V_h$.

In this thesis, we study the so-called $t$-local (or, simply local) model for some positive integer $t \in \mathbb{N}$, which assumes that for every $u \in V_h$: $|N(u) \cap V_f| \leq t$. For simplicity, we assume all nodes have prior knowledge of $t$. That being said, our algorithm can be easily generalized to remove this assumption at the cost of some additional time complexity (e.g., by executing several instances with increasing guesses at $t$).

The Power of Byzantine Failures

We assume the same broadcast-CONGEST message passing model of communication as we do in Section 4.3.1. Recall that we assume in this model that communication is strongly synchronous. In the context of byzantine failures, this prevents so-called rushing attacks in which faulty nodes wait to observe the behavior of their honest neighbors in a round before deciding on their own. For simplicity, we also assume that each node can broadcast at most one message per round. Finally, we assume strong signatures. Formally, each node

---

7This restriction prevents byzantine nodes from directly simulating many nodes at once. A more general assumption is that each byzantine node can instead send up to $m \geq 1$ messages per round.
is provided a public/private key pair \((p_u, k_u)\), that allows \(u\) to identify itself with \(p_u\), and sign every message in a manner that allows other nodes to verify the signature comes from the node that knows \(k_u\). We assume these signatures cannot be forged. As we detail below, byzantine nodes can generate as many of these pairs as they want, but cannot forge the signatures of honest nodes.

We assume nodes begin with no advance information about the network topology. Without loss of generality, however, we can assume that every algorithm begins with a preliminary round in which each node broadcasts a signed message announcing its public key and the fact that it is in the neighborhood. This fixes each node’s knowledge of their 1-hop neighborhood. Each byzantine node is forced to essentially commit to a single identity in this first round, or be silent. Honest nodes can ignore subsequent messages directly received from “neighbors” that did not announce themselves in this first round.

We also assume that all dishonest nodes share full knowledge of each other’s state at the beginning of each round, enabling full coordination of their behavior. The model described above restricts byzantine nodes in three important ways: (1) they cannot forge messages from honest nodes; (2) they can only send one message per round; and (3) and they cannot rush the channel by waiting to hear honest messages before sending their own in a given round. Furthermore, the assumption made above about nodes broadcasting their identity in the first round effectively limits byzantine nodes to choosing an identity to stick with for the duration of an execution. The main disruptive power of these failures, therefore, is less about directly pretending to be multiple nodes, and more about executing their algorithm in a malicious manner; e.g., replacing random choices with worst-case choices, coordinating behavior with other byzantine nodes, lying about what they hear from other neighbors, crashing at inopportune times, and so on. As also mentioned above, byzantine

In this case, each such node can directly simulate \(m\) unique nodes, sending one message on behalf of each simulated node in each round. In such a setup, however, it is arguably easier to consider the bound \(t\) to represent \(m\) times the actual number of byzantine nodes.
nodes can generate multiple public/private key pairs. Therefore, even though they must declare a single identity to their direct neighbors in the first round, they can indirectly simulate the existence of any number of other nodes which they can pretend to exist elsewhere in the network.

**The Byzantine MIS Problem**

The standard definition of an MIS (Definition 4.3.1) requires the set to satisfy two properties: maximality and independence. It is unclear, however, what it means to solve this problem in the presence of byzantine nodes.

How, for example, do we now determine what it even means for a node to join $M$? We may require that each node make an internal decision whether or not be in the set. The issue here, however, is that it is not reasonable to consider the internal state of a dishonest node, as we can make no assumption about the algorithm it is running. We might instead require honest nodes to form *opinions* about which of their neighbors are in $M$, and require that all of the opinions held by these nodes are consistent. We encounter problems here too, however, as the dishonest nodes might cause disagreements between honest neighbors which are not directly connected in the network. Indeed, such consistent opinion sets might not be possible.

Here, we settle on a simpler alternative. Each node now has the ability to publicly declare its intention to irrevocably join $M$ by broadcasting a special signed *join* message. If a node broadcasts this message it is considered, from the perspective of our analysis, to be included in $M$, whether it is honest or not. We now have a clean method to define $M$, the question then becomes what maximality and independence constraints to place on this set. We start with maximality. It would be unreasonable to require honest maximality (i.e., every honest node must be covered by an honest node), as dishonest nodes can *act* honest. Consider a clique where half the nodes are dishonest but all faithfully run the MIS.
algorithm in question. How can we ever determine if the single node eventually elected to the set is honest or not? We therefore place no restriction on this property. So long as a node is in $M$ or covered by a node in $M$, it is considered covered.

More interesting is independence. We cannot require independence between all nodes in $M$ because there is no restriction on byzantine nodes declaring themselves in the set. Returning to the clique example, assume an honest node $u$ elects itself to join the MIS and cover the entire clique. Nothing stops a byzantine node from later also declaring itself in $M$, violating independence. Accordingly, we require so-called honest independence, which requires that no two honest neighbors are both in $M$. We formalize this definition as follows:

**Definition 4.4.1.** A set $M \subseteq V$ is a correct maximal independent set if for every $u \in V_h$:

1. $N^+(u) \cap M \neq \emptyset$ (maximality), and

2. for every $v \in N(u) \cap V_h$, if $u \in M$ then $v \notin M$ (honest independence).

While we arrive at the above definition, note that there is no perfect single description of the MIS problem with byzantine failures. A shortcoming of our above definition, for example, is that it allows byzantine nodes to aggressively announce themselves to be included in the set, potentially terminating an MIS algorithm before any useful computation is completed by the honest nodes. This issue, however, is less relevant to our primary focus on time complexity, where such an attack would help the honest nodes terminate even faster. In our setting, the simplicity of the definition provides clarity, allowing for crisper algorithms and analysis. For other metrics, however, the ability to unilaterally declare yourself in the set might be problematic, and therefore motivate more involved definitions. We further note that this definition is not achieved by existing algorithms in the presence of byzantine faults. For example, with Luby's algorithm, it is not hard to show that a single byzantine node can prevent termination indefinitely by lying about its random choices.
Consider the version of Luby’s where in a given round, a participating node $u$ choose a random value, and then joins the set if its value is the smallest out of all its participating neighbors in that round (with ties broken using the unique node identifiers). Assume $u$’s only neighbor is a byzantine node $v$. Consider the strategy where in every round $v$ announces that it is still participating and that its random value is the smallest possible value. Clearly, $u$ will never terminate. Similar attacks can be easily extended to prevent termination in other existing algorithms, such as those in [10, 54]. Satisfying our definition of the MIS problem in the presence of byzantine nodes will require some additional effort, which we apply in Section 4.4.2.

4.4.2 Upper Bound for the Byzantine MIS Problem

In this section, we prove an upper bound for the problem defined in Section 4.4.1 by providing an algorithm which computes a correct MIS in this model w.r.t. Definition 4.4.1.

Algorithm

We now introduce the random channel MIS algorithm. This algorithm aims to address the vulnerability we saw above with existing algorithms by only allowing a subset of nodes to participate at a time. Namely, the algorithm takes place over phases of $2t$ rounds each. In each phase, we use these rounds to simulate $2t$ channels; where each node which is eligible to join the MIS chooses exactly one channel to participate in (uniformly at random).

The full pseudocode of the algorithm can be found in Algorithm 8. We use the method \texttt{VerifiedSendReceive} for all communication in our algorithm. Inherent to this method is the use of an established PKI to authenticate all received communications. The special join event for this algorithm is the call \texttt{VerifiedSendReceive}($j_v$) on Line 22 (in accordance with our model definition outlined in Section 4.4.1).
Algorithm 8 Random channel MIS algorithm for node $u \in V_h$

1: $M \leftarrow \emptyset$  
\hspace*{1.5em} $\triangleright$ the MIS under construction, initially empty
2: 
3: function $\text{RandomChannelLuby}(t, N^+(u))$
4: 
\hspace*{1.5em} $c_v, s_v, m_v, j_v \leftarrow 0$ for all $v \in N^+(u)$  
\hspace*{1.5em} $\triangleright$ initialize variables for all $v \in N^+(u)$
5: $i \leftarrow 0$
6: 
7: while $N^+(u) \cap M = \emptyset$ do
8: \hspace*{1.5em} if $i \equiv 0 \pmod{2^t}$ then
9: \hspace*{2em} $c_u \leftarrow$ uniform random value from $R(2^t - 1)$  
\hspace*{2em} $\triangleright$ select random channel
10: \hspace*{2em} $\text{VerifiedSendReceive}(c_u)$  
\hspace*{2em} $\triangleright$ send $c_u$ and receive $c_v$ for all $v \in N(u)$
11: 
12: \hspace*{1.5em} $m_u \leftarrow |N^+(u) \cap M|$  
\hspace*{1.5em} $\triangleright$ if $m_u > 0$ then $u$ is covered by $M$
13: \hspace*{2em} $\text{VerifiedSendReceive}(m_u)$  
\hspace*{2em} $\triangleright$ send $m_u$ and receive $m_v$ for all $v \in N(u)$
14: 
15: \hspace*{1.5em} $P \leftarrow \{v \in N^+(u) \mid [m_v = 0] \land [i \equiv c_v \pmod{2t}]\}$
16: \hspace*{1.5em} $\triangleright$ uncovered nodes which selected current channel
17: 
18: \hspace*{1.5em} $s_u \leftarrow$ uniform random value from $R(n^\ell - 1)$
19: \hspace*{2em} $\text{VerifiedSendReceive}(s_u)$  
\hspace*{2em} $\triangleright$ send $s_u$ and receive $s_v$ for all $v \in N(u)$
20: 
21: \hspace*{1.5em} $j_u \leftarrow \begin{cases} 
1 & \text{if } u \in P \land s_u = \min \{s_v \mid v \in P\} \land u = \min \{v \in P \mid s_v = s_u\} \\
0 & \text{otherwise} 
\end{cases}$
22: \hspace*{2em} $\text{VerifiedSendReceive}(j_u)$  
\hspace*{2em} $\triangleright$ send $j_u$ and receive $j_v$ for all $v \in N(u)$
23: 
24: $M \leftarrow M \cup \{v \in N^+(u) \mid j_v = 1\}$
25: $i \leftarrow i + 1$

A node $u$’s choice of channel $c_u$ is broadcast at the beginning of the phase and recorded
by all its neighbors. The algorithm then executes $2^t$ modified Luby steps (Lines 21-25). For
each step, nodes broadcast whether or not they are covered by the current MIS. In the $\ell$th
Luby step of the current phase, each node which selected channel $\ell$ in this phase and is still
uncovered at the beginning of this step participates in this step.
Correctness Analysis

We begin by quickly arguing that Algorithm 8 computes a correct MIS $M$ (agreeing with Definition 4.4.1). Fix some honest node $u \in V_h$ and note that as long as $N^+(u) \cap M = \emptyset$, the algorithm never terminates for node $u$. We will later prove that the algorithm does indeed terminate but for now simply note that if it does terminate then for every $u \in V_h$ we have that $N^+(u) \cap M \neq \emptyset$.

We now argue honest independence. Note, however, that this property is held in a similar way to Luby's original algorithm. For two eligible, honest, neighboring nodes $u$ and $v$ participating in the same Luby step; the only way either node joins the MIS is if one has the minimum value rolled in its neighborhood, with ties broken by unique node identifiers. This can clearly only hold for at most one of these nodes and so both cannot join the MIS in the same round. The honest independence property then follows since if either $u$ or $v$ join $M$, then in all future rounds both nodes will be ineligible to join the MIS.

Complexity Analysis

We now analyze the time complexity of the random channel MIS algorithm when $|N(u) \cap V_f| \leq t$ for all $u \in V_h$, proving the following claim,

**Theorem 4.4.2.** The Random Channel MIS algorithm computes a correct MIS (where correctness is defined with respect to Definition 4.4.1) in at most $O(t \log(n))$ expected rounds.

We begin by defining some additional notation. Similar to our definitions of honest and dishonest nodes, call any edge $\{u, v\} \in E$ honest if both endpoints are honest (i.e., $\{u, v\} \subseteq V_h$), otherwise $\{u, v\}$ is dishonest. Let $E_h \subseteq E$ be the honest edge set (i.e., the edges of $G_h$ where $G_h$ is the subgraph of $G$ induced by $V_h$).
Fix a step $i \in \mathbb{N}$ of our algorithm and node $u \in V$. Let $c_i(u)$, $m_i(u)$, and $s_i(u)$ denote the values of $c_u$, $m_u$, and $s_u$ at the beginning of step $i$, respectively.\(^8\) Let $M_i$ be the MIS prior to step $i$ and let $V_{h,i} = \{u \in V_h \mid N^*(u) \cap M_i = \emptyset\}$ denote the set of honest nodes eligible to join the MIS (i.e., those that have no neighbors in $M_i$). Let $E_{h,i}$ be the edge set of the subgraph, $G_{h,i}$, of $G$ induced by $V_{h,i}$. For a fixed step $i$ where $i \equiv 0 \pmod{2t}$, let $C_\ell \subseteq V_{h,i}$ be the subset of honest nodes which selected channel $\ell$. Next let $H_i(u) = N(u) \cap V_{h,i}$ be a node $u$’s honest neighborhood where $d_i(u) = |H_i(u)|$ is its honest degree. Note that if $u$ is not eligible itself (i.e., $u \notin V_{h,i}$) then we define $d_i(u) = 0$.

Next, recall Definition 1.3.17 from [1] which describes the concept of good nodes with respect to a graph. Let $V_{g,h,i} \subseteq V_{h,i}$ denote the vertices of $V_{h,i}$ that are good with respect to $G_{h,i}$ which is the subgraph of $G$ induced by $V_{h,i}$. Good nodes are important for two reasons. The first is that many of the edges of any graph are incident on its good nodes, as formalized in Theorem 1.3.18.

The second reason good nodes are significant to our analysis here is that their high density of uncovered neighbors which have lower relative degree makes them relatively more likely to join the MIS. These are often called a node $u$’s light neighbors, which we will denote $L_i(u) = \{v \in H_i(u) \mid d(v) \leq d(u)\}$. In the analysis of [1], the degree requirement was sufficient for a light neighbor to be somewhat likely to join the MIS. However, this is of course not the case in our model.

First of all, we can only rely on honest neighbors of $u$ to join the MIS at any given time. For this reason we defined $L_i(u)$ such that $L_i(u) \subseteq H_i(u)$. Secondly, we also cannot rely on honest nodes to join the MIS if they have selected the same channel as one of their dishonest neighbors. For this reason we further filter $u$’s light neighbors into a set which contains only those which select a channel in step $i$ without any of its own dishonest neighbors.

\(^8\)Note, of course if node $u$ ceases running the algorithm due to being covered, the values of $c_u$, $m_u$, and $s_u$ are no longer defined. For simplicity of our notation, however, let $c_i(u)$, $m_i(u)$, and $s_i(u)$ denote what $u$ would set in step $i$ for the corresponding value even if $u$ does not participate in step $i$. 198
neighbors. We call these nodes selectors of $u$ and formally denote the set of $u$'s selectors $S_i(u) = \{v \in L_i(u) \mid c_i(v) \neq c_i(w) \text{ for all } w \in N(v) \cap V_f\}$.

One way to express the main goal of our analysis is to find a step $i \in \mathbb{N}$ such that $V_{h,i} = \emptyset$. In other words, our goal would be to identify a step $i > 0$ such that each node $u \in V_h$ is "removed" where we say $u$ is removed in step $i$ if $u \in V_{h,i}$ and $u \notin V_{h,i+1}$. However, instead of attempting this directly, conventional wisdom has shown it is conceptually simpler to focus on removing the edges of the graph (i.e., find a step $i$ such that $E_{h,i} = \emptyset$). We therefore express our high level analytical strategy as an attempt to lower bound the expected number of edges "resolved" from $E_{h,i}$ after executing some number of steps of the algorithm where we say an honest edge $\{u, v\} \in E_{h,i}$ such that $\{u, v\} \notin E_{h,i+1}$ is resolved in step $i$ (namely at least one endpoint became ineligible in this step). This definition allows us to transform our goal from removing honest, eligible nodes to the (almost\(^9\)) equivalent goal of resolving honest edges.

To do so, we would like to follow a similar strategy to the analysis of [1] and show that any good node with respect to the graph at some particular point in time has at least a constant probability of being removed. This, however, is not the case in our model. It may be that a node chooses a high-priority random value and is likely to join the MIS given the chance, however, it participates in a later channel and therefore is prematurely covered. We therefore cannot bound a good node’s chance to be removed as in [1]. Instead we build towards the weaker result in Lemma 4.4.4 which states only that these nodes have a constant fraction of their incident edges resolved in a given phase. With some extra arguing in Lemma 4.4.5 we then achieve a result akin to the main insight of [1]. Namely,

\(^9\)There is a small subtlety here. It is not actually sufficient for every edge between two honest endpoints to be removed to ensure that every honest node has decided its status, as an undecided honest node can neighbor undecided, dishonest neighbors exclusively (violating maximality despite technically having no unresolved honest edges). We will, however, deal with this edge case later in our analysis.
we show that even with our weaker result regarding the fate of the good nodes, still at most a constant fraction of the honest edges remain unresolved by the end of the phase.

The rest of the analysis to resolve all edges in $E_h$ then follows the same as in Luby’s. There is, however, the extra point about what to do when $E_h = \emptyset$ yet some number of honest vertices still have not decided their status. Since this stipulation is not too cumbersome, we save it for the proof of our main theorem.

We now execute the above analytical strategy which begins by building towards our result in Lemma 4.4.4. To this end, we first show that the number of a good node’s light neighbors that avoid their own dishonest neighbors (i.e., its selectors) is at least a constant fraction of its degree (with at least constant probability).

**Lemma 4.4.3.** For step $i \in \mathbb{N}$ where $i \equiv 0 \pmod{2t}$ and node $u \in V_{h,i}^g$,

$$\Pr\left[ |S_i(u)| \geq d_i(u)/2^4 \right] \geq 1/2^7.$$  \hfill (4.14)

**Proof.** First, note that if $d_i(u) = 0$ then the lemma statement holds trivially.

Therefore, assume $d_i(u) \geq 1$. Since $u \in V_{h,i}^g$, we have from Definition 1.3.17 that $|L_i(u)| \geq \frac{1}{3} \cdot d_i(u)$. Given a fixed set of channel choices of all dishonest nodes, it should be clear that for each $v \in L_i(u)$, that $\Pr[v \in S_i(u) \mid v \in L_i(u)] \geq \frac{1}{2}$. The reason is that there are $2t$ total channels for $v$ to choose from and $|N(v) \cap V_f| \leq t$; therefore at most $t$ channels contain a dishonest neighbor of $v$. We then get that

$$\mathbb{E}[|S_i(u)|] \geq \frac{1}{2} \cdot \frac{1}{3} \cdot d_i(u) \geq \frac{d_i(u)}{6} \geq \frac{d_i(u)}{2^3}.$$  \hfill (4.15)  \hfill (4.16)  \hfill (4.17)
Furthermore, note that the events that any two nodes \( v, w \in L_i(u) \) are independent of one another given the channel choices of all dishonest nodes neighboring \( v \) and \( w \). We can therefore apply the Chernoff bound (where \( \varepsilon = \frac{1}{2} \) and \( \mu \geq \frac{d_i(u)}{2^4} \)) from Theorem 1.3.4 as follows:

\[
\Pr[|S_i(u)| \geq d_i(u)/2^4] \geq 1 - \Pr[|S_i(u)| \leq d_i(u)/2^4]
\geq 1 - \exp(-d_i(u)/2^6) \tag{4.18}
\]

Then, since we assume \( d_i(u) \geq 1 \), we have that

\[
1 - \exp(-d_i(u)/2^6) \geq 1/2^7 \tag{4.20}
\]

Therefore, \( \Pr[|S_i(u)| \geq d_i(u)/2^4] \geq 1/2^7 \) and the lemma is satisfied. \( \square \)

As we mentioned earlier, the above is unfortunately not enough to guarantee that a good node \( u \in V_g^i \) is removed with sufficient probability (since many nodes in \( S_i(u) \) may have a neighbor join the MIS before they have a chance to participate in their channel). We therefore still cannot apply the same techniques from [1].

However, in the event that many nodes \( S_i(u) \) have a neighbor join the MIS, intuitively this seems fine as it means many honest edges are still being resolved in the current phase. We formalize this intuition below and show that at least a constant fraction of edges incident to \( u \) are resolved in expectation in the current phase.

**Lemma 4.4.4.** For step \( i \in \mathbb{N} \) where \( i \equiv 0 \pmod{2t} \) and node \( u \in V_{h,i}^s \)

\[
\mathbb{E}[d_{i+2i}(u)] \leq (1 - 1/2^{14}) \cdot d_i(u). \tag{4.21}
\]
Proof. Fix our step \( i \geq 0 \) where \( i \equiv 0 \) (mod \( 2t \)) and node \( u \in V_{h,i}^{g} \) and for now assume that \( |S_{i}(u)| \geq \frac{d_{i}(u)}{2^{t}} \). By definition, each node \( v \in C_{\ell} \cap S_{i}(u) \cap V_{h,i+\ell} \) for channel \( \ell \in R(2t - 1) \), joins the MIS in channel \( \ell \) with probability at least \( \frac{1}{d_{i}(v)} \). That is to say \( v \) joins with this probability if: \( v \) chooses channel \( \ell \), none of \( v \)'s dishonest neighbors choose channel \( \ell \), and if \( v \) is not covered by the MIS at the start of step \( i + \ell \). This follows from the fact that an eligible node \( v \in V_{h,i+\ell} \cap C_{\ell} \) where \( N(v) \cap V_{\ell} \cap C_{\ell} = \emptyset \) joins with probability at least

\[
\frac{1}{|H_{i+\ell}(v) \cap C_{\ell}|} \leq \frac{1}{|H_{i}(v)|} \geq \frac{1}{d_{i}(v)}. \tag{4.22}
\]

We now go channel by channel, recognizing that by the time we reach a certain channel \( \ell \), an arbitrary number of nodes in \( S_{i}(u) \cap C_{\ell} \) may have already had a neighbor join the MIS. Each node which is made ineligible in this way hurts \( u \)'s chances to have a neighbor join the MIS. However, in a way this is a good thing since it means \( u \)'s incident edges in \( E_{h,i} \) are still being resolved. We use this intuition to explore two disjoint cases.

For our fixed step \( i \) and node \( u \), let \( X_{\ell} = S_{i}(u) \cap C_{\ell} \) be those selectors of \( u \) which choose channel \( \ell \) and let \( Y_{\ell} = X_{\ell} \setminus V_{h,i+\ell} \) be those that had a neighbor join the MIS before channel \( \ell \) of step \( i \). Let \( x_{\ell} = |X_{\ell}| \) and \( y_{\ell} = |Y_{\ell}| \) and denote their sums as \( x = \sum_{\ell \in R(2t-1)} x_{\ell} \) and \( y = \sum_{\ell \in R(2t-1)} y_{\ell} \). Note that \( x = |S_{i}(u)| \). Once again, recall that for now we assume \( x \geq \frac{d_{i}(u)}{2^{t}} \).

We now show that given this assumption, \( E[d_{i+2}(u)] \leq (1 - \frac{1}{2^{t}}) \cdot d_{i}(u) \). Note that if \( y \geq \frac{d_{i}(u)}{2^{t}} \), then clearly \( d_{i+2}(u) \leq (1 - \frac{1}{2^{t}}) \cdot d_{i}(u) \leq (1 - \frac{1}{2^{t}}) \cdot d_{i}(u) \) which satisfies the inequality. Therefore we further assume that \( y \leq \frac{d_{i}(u)}{2^{t}} \).

Further note that for any \( v \in X_{\ell} \setminus Y_{\ell} \), the probability that \( v \) selects the lowest value \( s_{v} \) in \( H_{i+\ell}(u) \cup H_{i+\ell}(v) \) is at least \( \frac{1}{d_{i+\ell}(u) + d_{i+\ell}(v)} \geq \frac{1}{d_{i}(u) + d_{i}(v)} \). Note that the events that two nodes \( v_{1}, v_{2} \in S_{i}(u) \) select such a value are disjoint, as both cannot select the lowest value in \( H_{i+\ell}(u) \) to begin with. Since each such event is sufficient for \( u \) to be removed in channel
ℓ, if we assume that u has not been removed in any channel 0, . . . , ℓ − 1, the probability that u is removed in channel ℓ is at least \( \sum_{v \in X_\ell \setminus Y_\ell} \frac{1}{d_i(u) + d_i(v)} \). Therefore, u is removed with probability at least,

\[
\sum_{\ell \in R(2t-1)} \sum_{v \in X_\ell \setminus Y_\ell} \frac{1}{d_i(u) + d_i(v)} \geq \sum_{\ell \in R(2t-1)} \sum_{v \in X_\ell \setminus Y_\ell} \frac{1}{2d_i(u)}
\]

(4.24)

\[
= \sum_{\ell \in R(2t-1)} \frac{|X_\ell \setminus Y_\ell|}{2d_i(u)}
\]

(4.25)

\[
= \sum_{\ell \in R(2t-1)} \frac{x_\ell - y_\ell}{2d_i(u)}
\]

(4.26)

\[
= \frac{x - y}{2d_i(u)}
\]

(4.27)

\[
\geq \frac{1}{2d_i(u)} \cdot \left( \frac{d_i(u)}{2^4} - y \right)
\]

(4.28)

\[
\geq \frac{1}{2d_i(u)} \cdot \left( \frac{d_i(u)}{2^4} - \frac{d_i(u)}{2^5} \right)
\]

(4.29)

\[
= \frac{1}{2d_i(u)} \cdot \frac{d_i(u)}{2^5}
\]

(4.30)

\[
= 1/2^6
\]

(4.31)

\[
\geq 1/2^7.
\]

(4.32)

(4.33)

In the event this occurs, then all edges incident to u are resolved, and so \( d_{i+2t}(u) = 0 \). Therefore, \( E[d_{i+2t}(u)] \leq (1 - \frac{1}{2^7}) \cdot d_i(u) \). This satisfies our lemma for the case that \( y \leq \frac{d_i(u)}{2^5} \).

We have now shown that given \( |S_i(u)| \geq \frac{d_i(u)}{2^5} \), \( E[d_{i+2t}(u)] \leq (1 - \frac{1}{2^7}) \cdot d_i(u) \). Denote the event \( |S_i(u)| \geq \frac{d_i(u)}{2^5} \) by the symbol \( \mathcal{A} \) and let \( \neg \mathcal{A} \) be its complement. Using this notation, we can
bound the final expectation of $d_{i+2}(u),$

$$E[d_{i+2}(u)] \leq E[d_{i+2}(u) \mid \mathcal{A}] \cdot \Pr[\mathcal{A}] + E[d_{i+2}(u) \mid \neg\mathcal{A}] \cdot \Pr[\neg\mathcal{A}] \quad (4.34)$$

$$\leq (1 - 1/2^7) \cdot d_i(u) \cdot \Pr[\mathcal{A}] + d_i(u) \cdot \Pr[\neg\mathcal{A}] \quad (4.35)$$

Note that since $\Pr[\neg\mathcal{A}]$ is multiplied by $d_i(u)$ and $\Pr[\mathcal{A}]$ is multiplied by $(1 - 1/2^7) \cdot d_i(u) < d_i(u)$, this expression is monotonically increasing with respect to $\Pr[\neg\mathcal{A}]$. Since from Lemma 4.4.3 we have that $\Pr[\mathcal{A}] \geq 1/2^7$ and, equivalently, $\Pr[\neg\mathcal{A}] \leq 1 - 1/2^7$, we get:

$$E[d_{i+2}(u)] \leq (1 - 1/2^7) \cdot d_i(u) \cdot \Pr[\mathcal{A}] + d_i(u) \cdot \Pr[\neg\mathcal{A}] \quad (4.36)$$

$$\leq (1 - 1/2^7) \cdot d_i(u) \cdot (1/2^7) + d_i(u) \cdot (1 - 1/2^7) \quad (4.37)$$

$$= (1 - 1/2^7) \cdot (1 + 1/2^7) \cdot d_i(u) \quad (4.38)$$

$$= (1 - 1/2^{14}) \cdot d_i(u) \quad (4.39)$$

Thus satisfying the claim made by the lemma statement. $\square$

The equivalent result from [1] states that a good node $u \in V_{h,i}^g$ would actually be removed with at least constant probability. Our result is weaker and so we need to continue to deviate somewhat from the original proof to achieve our goal (which is that at least a constant fraction of edges are removed each phase). We accomplish this goal in following lemma.

**Lemma 4.4.5.** For a step $i \in \mathbb{N}$ where $i \equiv 0 \pmod{2t}$, $E[|E_{h,i+2t}|] \leq (1 - 1/2^{16}) \cdot |E_{h,i}|$. 

204
Proof. For fixed step \( i \equiv 0 \pmod{2t} \), let \( D_i = \sum_{u \in V_{h,i}^g} d_i(u) \) be the degree sum of \( G_{h,i} \)'s good nodes. Note

\[
E[D_{i+2t}] = \sum_{u \in V_{h,i+2t}^g} E[d_{i+2t}(u)] \quad (4.40)
\]

\[
\leq \sum_{u \in V_{h,i}^g} E[d_{i+2t}(u)] \quad (4.41)
\]

Applying our result from Lemma 4.4.4 regarding the expected drop in good node degree, we get that

\[
E[D_{i+2t}] \leq \sum_{u \in V_{h,i}^g} E[d_{i+2t}(u)] \quad (4.42)
\]

\[
\leq \sum_{u \in V_{h,i}^g} (1 - 1/2^{14}) \cdot d_i(u) \quad (4.43)
\]

\[
= (1 - 1/2^{14}) \sum_{u \in V_{h,i}^g} d_i(u) \quad (4.44)
\]

\[
= (1 - 1/2^{14}) \cdot D_i. \quad (4.45)
\]

Another way to express this result is in terms of the expected difference between the two degree sums represented above. Namely we have that \( E[D_i - D_{i+2t}] = D_i - E[D_{i+2t}] \geq 1/2^{14} \cdot D_i \). If we then look at the edges which are resolved in steps \( \langle i, \ldots, i + 2t - 1 \rangle \) to cause this drop in degree sum, observe that each edge resolved from \( E_{h,i} \) accounts for exactly 2 of this difference. Therefore,

\[
|E_{h,i} \setminus E_{h,i+2t}| \geq \frac{1}{2} \cdot (D_i - D_{i+2t}) \quad (4.46)
\]

\[
\Rightarrow E[|E_{h,i} \setminus E_{h,i+2t}|] \geq \frac{1}{2} \cdot E[D_i - D_{i+2t}] \quad (4.47)
\]

\[
\geq \frac{1}{2^{15}} \cdot D_i. \quad (4.48)
\]
Let $E^g_{h,i} \subseteq E_{h,i}$ be the edges incident on nodes in $V^g_{h,i}$ and note two things. The first is that since the degree sum of a set of nodes is at least the number of edges incident on these nodes, $D_i \geq |E^g_{h,i}|$. Furthermore, from Theorem 1.3.18 we have that $|E^g_{h,i}| \geq \frac{1}{2} \cdot |E_{h,i}|$. We therefore have that the expectation of $E_{h,i+2t}$ is at most

\[
\mathbb{E}[|E_{h,i+2t}|] = |E_{h,i}|-\mathbb{E}[|E_{h,i} \setminus E_{h,i+2t}|]
\]

\[
\leq |E_{h,i}|-\frac{1}{2^{15}} \cdot D_i \tag{4.49}
\]

\[
\leq |E_{h,i}|-\frac{1}{2^{15}} \cdot |E^g_{h,i}| \tag{4.50}
\]

\[
\leq |E_{h,i}|-\frac{1}{2^{16}} \cdot |E_{h,i}| \tag{4.51}
\]

\[
= \left(1-\frac{1}{2^{16}}\right) \cdot |E_{h,i}|. \tag{4.52}
\]

Therefore, $\mathbb{E}[|E_{h,i+2t}|] \leq \left(1-1/2^{16}\right) \cdot |E_{h,i}|$ as claimed by the lemma statement. \hfill \Box

We now have the required support to prove Theorem 4.4.2.

Proof. (of Theorem 4.4.2) First, for random variable $X$, let the notation $\mathbb{E}_i[X]$ denote the expectation of $X$ given the randomness up to and including step $i > 0$. From Lemma 4.4.5 we then have that for a step $i \equiv 0 \pmod{2t}$ it follows that $\mathbb{E}_i[|E_{h,i+2t}|] \leq \left(1-1/2^{16}\right) \cdot |E_{h,i}|$.

Therefore if we consider $T$ phases of $2t$ steps each, we can recursively apply this result $T$
times. Namely, we have that,
\[
E_{(T-1)\cdot 2t}[|E_{h,T\cdot 2t}|] \leq (1 - 1/2^{14}) \cdot |E_{h,(T-1)\cdot 2t}|
\]  
(4.54)
\[
\Rightarrow E_{(T-2)\cdot 2t}[E_{(T-1)\cdot 2t}[|E_{h,T\cdot 2t}|]] \leq (1 - 1/2^{14}) \cdot E_{(T-2)\cdot 2t}[|E_{h,(T-1)\cdot 2t}|]
\]  
(4.55)
\[
\Rightarrow E_{(T-1)\cdot 2t}[|E_{h,T\cdot 2t}|] \leq (1 - 1/2^{14})^T \cdot E_{(T-2)\cdot 2t}[|E_{h,(T-1)\cdot 2t}|]
\]  
(4.56)
\[
\leq (1 - 1/2^{14})^T \cdot |E_{h,(T-2)\cdot 2t}|
\]  
(4.57)
\[
\ldots
\]  
(4.58)
\[
\leq (1 - 1/2^{14})^T \cdot |E_h|.
\]  
(4.59)

Since $|E_h| \leq n^2$, if we set $T = \Theta(\log n)$, we then get that the expected number of phases until all honest edges are resolved is in $O(\log n)$. Therefore, only $O(t \log n)$ total steps are required in expectation to resolve all honest edges. However, as we noted at the beginning of our analysis our goal is not find a step $i$ such that $E_{h,i} = \emptyset$; but rather to find a step $i$ such that $V_{h,i} = \emptyset$. Again, the possible case that separates these two criteria is that an honest node $u \in V_{h,i}$ can potentially only neighbor dishonest nodes and still be undecided. However, in this case, note that if an honest node has $d_i(u) = 0$ for some step $i \equiv 0 \pmod{2t}$ then $\Pr[u \in V_{h,i+2t}] \leq \frac{1}{2}$. The reason being that all $u$ must do in this phase is to select a channel with no dishonest neighbors, leading to $u$ having no neighbors in its selected channel. If this occurs, only one of two events are possible. Either $u$ is removed before participating in its selected channel or it is guaranteed to join the MIS in its channel (either way, $u \not \in V_{h,i+2t}$).

Therefore, if we then let $i$ be the first round for which $E_{h,i} = \emptyset$, for every step $j \geq i$ where $j \equiv 0 \pmod{2t}$ we have that $E[|V_{h,j+2t}|] \leq \frac{1}{2} \cdot |V_{h,j}|$.

Again, recursively applying this inequality for $O(\log n)$ phases of $2t$ steps each gives us that if $i + \delta$ is the first round for which $V_{h,i+\delta} = \emptyset$ then $E[\delta] = O(t \log n)$ (i.e., only $O(t \log n)$
additional steps are required in expectation). Since $E[i] = O(t \log n)$ as well, the theorem then follows.

4.4.3 **Lower Bound for the Byzantine MIS Problem**

With the result we achieve in Section 4.4.2, a natural next question to ask is whether or not our upper bound must contain a dependence on $t$ at all? Is there perhaps a way to completely nullify the presence of the dishonest nodes when constructing an MIS? In this section we prove that the answer to this latter question is “no”: any algorithm which produces a correct MIS must take $\Omega(t)$ rounds in expectation.

Our lower bound requires the assumption of a *uniform* algorithm: that is, every node in the graph executes the exact same algorithm. The assumption of uniformity simplifies lower bound arguments as it allows us to describe the distribution of fates for a given node based solely on the algorithm and the graph topology.

**Theorem 4.4.6.** Every uniform algorithm requires $\Omega(t)$ expected rounds to compute a correct MIS.

**Proof.** Fix some $t > 0$. The graph $G = (V, E)$ we use in this proof consists of $n = t + 1$ nodes, arranged in a $(t + 1)$-clique. Label the node set $V = \{u_1, \ldots, u_{t+1}\}$. For any MIS algorithm $A$, the strategy for each dishonest node $u$ will be to behave honestly with one exception: if it arrives at a round in which it is supposed to join the MIS by broadcasting a join message, it instead permanently crashes itself at this point.

To simplify our analysis, we can assume without loss of generality that each node is assigned its random bits as input at the beginning of an execution. For a given bit assignment, we can define a permutation of the nodes, $\pi = \langle \pi_1, \ldots, \pi_{t+1} \rangle$, that we call the *dynamic crash sequence*, as follows: Run $A$ in $G$ until the first round $r$ in which some node $u_i$ is set to broadcast a join message. Set $\pi_1 = u_i$. Instead, of having $u_i$ broadcast the message in $r$,
permanently crash it. Now keep running \( \mathcal{A} \) until some later round \( r' \) during which some \( u_j \) is prepared to broadcast join. Set \( \pi_2 = u_j \). As before, crash \( u_j \) and keep running \( \mathcal{A} \).

A key observation about this process is that, in any given crash round, there can be at most one node poised to broadcast join. Assume, without loss of generality, that in some crash round \( r \), both \( u_i \) and \( u_j \) are poised to broadcast join. It is possible that we are in an execution in which the nodes that previously crashed were byzantine, but both \( u_i \) and \( u_j \) are honest. If they are left to broadcast join in \( r \), honest independence would be violated. It follows that each crash used to generate the crash sequence occurs in a unique round.

Moving forward, we now use the dynamic crash sequence to analyze our byzantine strategy in a setting in which the single honest node is selected with uniform randomness. In particular, fix a randomness assignment and corresponding dynamic crash sequence. Let \( \pi_i \) be the randomly selected honest node. It follows that the actual execution that unfolds will be indistinguishable from the execution used to generate the dynamic crash sequence up until the round in which \( \pi_i \) is supposed to crash. It follows that termination is delayed until at least the round corresponding to \( \pi_i \) in the sequence, which, as argued above, is at least round \( i \). Because we selected the honest node with uniform randomness independent of the dynamic crash sequence, the expected value of \( i \) (i.e., the expected position of \( \pi_i \) in the sequence) is \( (t + 2)/2 \).

Therefore, if the honest node is selected randomly, and the byzantine nodes execute the simple strategy from above, the expected time complexity is \( \Omega(t) \) rounds. \( \square \)


