(c) 2017 Shaileshh Bojja Venkatakrishnan

## ALGORITHMS FOR INTERACTIVE, DISTRIBUTED AND NETWORKED SYSTEMS

BY<br>SHAILESHH BOJJA VENKATAKRISHNAN

## DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:
Professor Pramod Viswanath, Chair
Professor Bruce Hajek
Professor R. Srikant
Assistant Professor Mohammad Alizadeh, Massachusetts Institute of Technology

## ABSTRACT

In recent years, massive growth in internet usage has spurred the emergence of complex large-scale networking systems to serve growing user bases, bandwidth and computation requirements. For example, data center facilities workhorses of today's internet - have evolved to house upward of several hundreds of thousands of servers; content distribution networks with high capacity and wide coverage have emerged as a de facto content dissemination modality, and peer-to-peer applications with hundreds of thousands of users are increasingly becoming popular. At these scales, it becomes critical to operate at high efficiencies as the price of idling resources can be significant. In particular, the interaction between agents (servers, peers etc.) is a defining factor of efficiency in these systems - applications are often communication intensive, whereas agents share links of only limited bandwidth. This necessitates the use of principled algorithms, as efficient communication to a large extent depends on the interaction protocols.

We study data center networks and peer-to-peer networks as canonical examples of modern-day large-scale networking systems. Server-to-server interaction is an integral part of the data center's operation. The latency of these interactions is often a significant bottleneck toward overall job completion times. We study complementary approaches toward reducing this latency: (i) design of computation algorithms that minimize interaction and (ii) optimal scheduling algorithms to maximally utilize the network fabric. We also consider peer-to-peer networks as an emerging mode of content distribution and sharing. Unlike data centers, these networks are flexible in their network structure and also scale well, but require decentralized algorithms for control. Of central importance here is the design of a network topology that enables efficient peer interactions for optimal application performance. We propose novel topology designs for two popular applications: (i) multimedia streaming and (ii) anonymity in Bitcoin's peer-to-peer network.

To my family.

## ACKNOWLEDGMENTS

Foremost I would like to thank Prof. Pramod Viswanath for his valuable advice and mentoring throughout my PhD. His deep insights, knowledge and critical thinking have been tremendously useful not only during the research process, but also in shaping me into a more mature researcher. I have thoroughly enjoyed working with him, and offer my sincere gratitude for making this dissertation possible.

I also thank my committee members Prof. Mohammad Alizadeh, Prof. Bruce Hajek and Prof. R. Srikant for their valuable feedback and helpful discussions. My foray into the world of networking would not have been possible without the interactions with Prof. R. Srikant. I thank Prof. Mohammad Alizadeh for introducing me to the pleasures of working at the intersection of systems and theory, and Prof. Chandra Chekuri for teaching me how to do principled algorithmic thinking. Each interaction with them has broadened my perspective.

I am also fortunate to have worked with my collaborators Prof. Giulia Fanti, Prof. Himanshu Tyagi and Prof. Shun Watanabe. Our projects together could not have been successful without our teamwork.

Life in grad school can be overwhelming, and for me it was never more so than during those years at the beginning. Wading through that could not have been possible without the help of my labmates and colleagues at CSL: Quan Geng, Xun Gong, Abhishek Gupta, Sachin Kadloor, Peter Kairouz, Sreeram Kannan, Siva Theja Maguluri and Jiaming Xu. They have been prompt in sharing their wisdom and experience whenever I needed them. Thanks are also due to Avhishek Chatterjee, Weihao Gao, Hongyu Gong, Hyeji Kim, Ashok Makkuva, Jiaqi Mu, Tarek Sakakini and Ge Yu for creating a cordial and intellectually stimulating environment, that was a pleasure to work from. I am also grateful to my many friends here in Champaign-Urbana, and my roommates Aravind Murali and Aditya Sarathy in particular. They
have kept my life balanced and have been there for me since day one of grad school.

Much of what I have learned, I owe to my wonderful teachers. I take this opportunity to sincerely thank my teachers from high school for making me believe in hard work, my professors from IIT Madras for making electrical engineering fun, and my professors here at Illinois for providing me with a wealth of knowledge and useful tools.

Last but not least, I would like to thank my incredible parents for their countless sacrifices and being wholeheartedly supportive of whatever I do. Looking back, I realize what I am today is purely because of their wisdom, foresight and motivation. I am also grateful to my sister-in-law for her constant support, and my lovely nephew for being a source of inspiration for all of us. Finally I would like to thank my brother, who has been my guiding light throughout, for teaching how to find roots of a quadratic equation which made me see elegance in mathematics and started this journey.

The research in this dissertation was supported in part by NSF grants CCF 1617745, CCF 1422278, CCF 1409016 and Army grant W911NF-14-1-0220.

## TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION ..... 1
1.1 Two-Party Communication ..... 2
1.2 Multi-Party Communication ..... 3
1.3 Outline and Contributions ..... 5
CHAPTER 2 INTERACTIVE COMMUNICATION ..... 8
2.1 Protocol Simulation ..... 14
2.2 Lower, Upper Bounds and Asymptotic Optimality ..... 17
2.3 Conclusion ..... 24
CHAPTER 3 DATA CENTER NETWORKS ..... 25
3.1 Switch System and Traffic Model ..... 28
3.2 Direct Routing ..... 33
3.3 Indirect Routing ..... 41
3.4 Evaluation ..... 50
3.5 Conclusion ..... 56
CHAPTER 4 PEER-TO-PEER NETWORKS ..... 57
4.1 Network Model and Steady-State Topology ..... 62
4.2 Properties to Be Maintained ..... 66
4.3 Repairing Algorithm ..... 72
4.4 Balancing Algorithm ..... 78
4.5 Reducing Redundancy ..... 83
4.6 Conclusion ..... 85
CHAPTER 5 ANONYMITY IN NETWORKS ..... 87
5.1 Bitcoin Network and Adversarial Model ..... 93
5.2 Anonymity Metric Properties ..... 101
5.3 Baseline Algorithms ..... 105
5.4 Dandelion ..... 109
5.5 Systems Issues ..... 116
5.6 Conclusion ..... 120
CHAPTER 6 CONCLUSION ..... 121
6.1 Open Problems and Future Directions ..... 121
APPENDIX A PROOFS FOR CHAPTER 2 ..... 124
A. 1 Background: Secret Key Agreement and Data Exchange ..... 124
A. 2 Section 2.2: Lower Bound ..... 130
A. 3 Section 2.2: Upper Bound ..... 140
A. 4 Section 2.2: Asymptotic Optimality ..... 154
APPENDIX B PROOFS FOR CHAPTER 3 ..... 157
B. 1 Section 3.2: Direct Routing ..... 157
APPENDIX C PROOFS FOR CHAPTER 4 ..... 162
C. 1 Section 4.5.1: Converse ..... 162
APPENDIX D PROOFS FOR CHAPTER 5 ..... 168
D. 1 Section 5.2: Anonymity Metric Properties ..... 168
D. 2 Section 5.3: Baseline Algorithms ..... 173
D. 3 Section 5.4: Main Result - Dandelion ..... 177
D. 4 Section 5.5: Systems Issues ..... 184
REFERENCES ..... 185

## CHAPTER 1

## INTRODUCTION

This past decade has seen several transformational changes in content generation, dissemination and consumption over the internet. Proliferation of numerous applications such as Google, Facebook, Netflix, Twitter etc. into the mainstream, together with rapid advancements in internet infrastructure have led to an explosive growth in end-user traffic. Services such as data storage, analytics, cloud computing offered by such applications, have also resulted in the emergence of data centers with massive scale and complexity. These are facilities housing in upward of one-hundred thousand servers under a single roof $[1,2]$. Multimedia distribution (video, music etc.) has also seen a sharp increase in the number of content distribution networks [3, 4]. In yet other applications, such as cryptocurrencies [5], decentralized modalities based on peer-to-peer ( p 2 p ) networking are used. These networks offer the distinct advantage of not having to rely on any one centralized server [6].

For a continued sustenance of growth and demand in the future, in addition to infrastructural innovations, strong and principled algorithms are required that can optimally operate the network. However many of the recent developments mentioned previously have, at their core, stemmed as basic engineering solutions with an emphasis on execution rather than optimality. This has been necessary, in part due to the fast pace of developments and the corporate race. The end result is that networks today suffer several problems, such as over-provisioning, and inadequate reliability. For example, it is reported that over $30 \%$ of data center servers worldwide remain idle [7]. On the other hand, even major providers experience capacity overload and outages during popular television events [8]. With an unceasing growth in scale, such events emphasize a stark need for more principled research in order to (a) understand and optimize existing infrastructure and (b) offer solutions that are better scalable.

Toward these objectives, in this dissertation we study the fundamental
limits and propose optimal algorithms across a canonical set of modern-day networking systems. While such a study is useful within the context of each of various sub-disciplines, such as power management, pricing, storage, computing, etc., ${ }^{1}$ our focus will specifically be on the communication protocols and the networking stack in these systems. This is an important avenue that is often algorithmically challenging even within simplified models of real-world networks. Indeed many practical algorithms deployed today are heuristics and may not be optimal. Moreover in several applications these systems are essentially communication limited, which further necessitates the development of better protocols and infrastructure.

Our results naturally categorize into algorithms for two-party and multiparty communication systems. In two-party communication, we investigate a setting where two interacting parties seek to jointly evaluate a function; whereas in the multi-party setting we study large-scale systems with potentially thousands of agents that interact with each other.

### 1.1 Two-Party Communication

Computing architectures today predominantly achieve performance scaling by aggregating multiple distinct processing entities and then interconnecting them. For example, data centers are built by networking several thousand servers each with dedicated compute and storage; whereas in highperformance micro-processor chips there could be multiple cores that are interconnected. Computation in these distributed processing frameworks often occurs in multiple interleaving stages of computation, and communication between individual processing units (e.g. map and reduce stages in MapReduce [9]). In general such an interactive exchange is necessary, however the communication phase is often slow and constitutes a key bottleneck in the overall execution latency. Hence it becomes imperative to design protocols such that communication exchange is minimized.

One-way communication, in which a transmitter seeks to send bits over a noisy channel, has been traditionally studied under the purview of information theory $[10,11]$. Abstracting out the semantics of the data, from the transmission problem itself, is a fundamental tenet of this subject. An

[^0]analogous theory for interactive communication is studied under the topic of communication complexity, where a more general problem of (i) what to communicate (protocols, semantics etc.) and (ii) how to communicate (coding etc.) are considered.

The most basic setting here is the two-party communication model introduced by Yao [12] in 1978. In this model, there are two players, Alice and Bob, who observe inputs $x$ and $y$ respectively. Their goal is to compute a function $f(x, y)$ by sending messages back and forth to each other over a noiseless channel while using the fewest bits. This leads to the notion of a communication protocol, which is a formal specification of how the messages should be exchanged. Since its introduction in [12], a lot of research has been done by the theoretical computer science community on developing techniques to lower bound the communication complexity of problems [13]. However from an algorithms perspective, optimal solutions for a general function computation problem in the two-party communication model still remains an unsolved problem [13]. Understanding this basic twoparty problem is essential for designing communication-efficient protocols in multi-party distributed computing frameworks. Toward this goal, we study the closely related interactive protocol compression problem [14, 15, 16, 17] in Chapter 2, that asks: Given an interactive communication protocol, can we find another protocol that "simulates" the given protocol while consuming fewer communication rounds?

### 1.2 Multi-Party Communication

While the two-party problem is a useful baseline, practical systems on the internet often consist of a significantly greater number of interacting agents - data centers house several 100,000 s of servers [1, 2], content distribution networks have upward of 60,000 servers [18], and popular p2p networks such as the Bitcoin network have more than 8000 nodes [19]. The interaction between such agents occurs over a network (e.g. internet, LAN), with the rules of interaction specified by protocols and algorithms (e.g. TCP/IP). Protocols may be required to coordinate a variety of networking operations, such as scheduling, rate-control etc., across agents. However, practical networks have often strict resource constraints (e.g. bandwidth, number of connec-
tions) that tightly couples the network-states as seen by different agents, making algorithm design a non-trivial problem. This is particularly true of modern large-scale systems, which despite having structural similarities to networking-systems of the past, exhibit combinatorics where prior-works often cannot be directly applied. We focus on two such systems - the data center network and p2p network - as discussed below.

Data center networks. Server-to-server interaction plays a crucial role in today's data centers. A vast majority of the jobs are structured to utilize services across multiple different servers, leading to heavy cross-flow of traffic. To facilitate such interactions, a fast, dynamic and economic interconnect is required at the physical level. Typically, data center networks use multi-rooted tree designs in which the servers are arranged in racks and an Ethernet switch at the top of the rack (ToR) connects the racks to one or more aggregation layers [1]. The designs use multiple-paths between the ToRs to deliver a uniform high bisection bandwidth and consist of a large number of electronic switches, but with poor speed to cost ratios. Recently there have been proposals to use high-speed circuit switches [20], to provide a higher aggregate capacity at a reduced cost, power and cabling complexity. However, circuit switches suffer from a key drawback of having a much slower switching time than their electronic counterpart. Depending on the switching technology, this delay can range from 10s of microseconds to a few milliseconds. As such, the scheduling algorithms currently used for packet switching do not directly apply to circuit switches and new algorithms are needed. In Chapter 3 we study this scheduling problem, and propose efficient near-optimal algorithms.

P2P networks. In p2p networks participating clients (peers) directly interact with each other, for content downloads and uploads, in a dynamic fashion rather than through a central server. This makes them attractive as a scalable content distribution modality, in which the available network capacity proportionately increases with the number of users. Since the late 1990s, p2p networks have been widely used over the internet, albeit predominantly for file-sharing applications [6]. However, today they are also increasingly being used for other applications such as multi-media streaming (live TV, video-on-demand etc.) and cryptocurrencies (Bitcoin, Ethereum etc.). Increased availability of bandwidth in the last mile is a key contributer
to this; cryptocurrencies such as Bitcoin also explicitly rely on decentralized networks for their operation. In these applications, the underlying topology over which the p2p protocol works is of crucial importance [21]. For example in p2p streaming, the underlying topology must be such that users receive a high streaming rate, low delay and minimal stream interruptions. Whereas in Bitcoin's p2p, the network must also provide good anonymity in addition to a low latency. How to construct p2p networks having these properties forms the subject matter of Chapters 4 and 5 respectively. Unlike data center networks discussed previously, the nodes (peers) in a p2p network can be geographically far apart necessitating protocols that are decentralized. This makes their design and implementation challenging.

### 1.3 Outline and Contributions

We present a summary of problems and results in the following.
Chapter 2. We consider a protocol compression problem, in which two parties - Alice and Bob - with inputs $X$ and $Y$ respectively, seek to simulate a given interactive protocol $\pi$ to within a fixed statistical distance $\varepsilon$ while using the fewest number of bits for exchange. This problem has received significant attention in recent years, where an information theoretic quantity called the information complexity $\operatorname{IC}(\pi)$ is used to bound communication complexity. This quantity plays a similar role in the simulation of protocols as $H(X)$ plays in the compression of $X^{n}$ or $H(X \mid Y)$ in compression with side information $Y$. Perhaps motivated by this analogy, recent works have proposed simulation protocols requiring communication of length depending on $\operatorname{IC}(\pi)$. For example, a protocol requiring $\tilde{O}(\sqrt{\operatorname{IC}(\pi)\|\pi\|})$ bits was proposed in [16] where $\|\pi\|$ denotes the length of the protocol tree of $\pi$. In another work [17]), a compression scheme requiring $2^{O(\operatorname{IC}(\pi))}$ bits has been proposed. However, there are still several questions that remain unanswered such as: (a) Can we find a lower bound for $D(\pi)$ ? or (b) Does the information complexity $\operatorname{IC}(\pi)$ have an operational role in simulating $\pi^{n}$ besides being the leading asymptotic term?

In Chapter 2 we answer these questions in the affirmative. Our first main result presents a general lower bound for the distributional communication complexity of an $\varepsilon$-simulation of a protocol $\pi$. The key quantity here is the
$\varepsilon$-tail of the information complexity density of the protocol $\pi$. The main idea here is to reduce the $\varepsilon$-simulation protocol to generate an information theoretically secure secret key for $X$ and $Y$. Then using an upper bound in $[22,23]$ for the maximum possible length of the secret key that can be generated using interactive communication, we can get our desired lower bound on the communication complexity.

The next main result is a new simulation protocol that simulates the given protocol $\pi$ one round at a time. In each round, we use an interactive version of the classical Slepian-Wolf compression, followed by further reduction in the message size using public randomness. This is possible since we are not required to recover the random variable $\Pi$ but rather only simulate it to within a fixed statistical distance.

Using the above lower and upper bounds, we also derive a tight (up to the second-order term) characterization of the distributional communication complexity of simulating an $n$-fold product protocol. This refers to a protocol $\pi^{n}$ obtained by applying $\pi$ to each coordinate ( $X_{i}, Y_{i}$ ) for inputs $X^{n}$ and $Y^{n}$ which are independently and identically generated from a joint distribution $P_{X Y}$.

Chapter 3. In Chapter 3 we consider the problem of scheduling circuit switches in data center networks. Commodity circuit switches (based on either wireless or optics) make for a faster and more economic cross-connect (compared to traditional packet switches), but have a very slow reconfiguration time. This makes their scheduling challenging, and conventional algorithms (such as MaxWeight scheduling) cannot be directly applied. Under a crossbar model for the circuit switch, our main result here is an approximately optimal, fast and efficient algorithm for computing the circuit switch schedule. Equivalently, the algorithm can also be seen as a fast method for computing a sparse representation of a point on the Birkhoff polytope. The main technical contribution here is the identification of a submodularity structure in the problem, the use of which enables us to make a connection to submodular function maximization.

We also consider a different routing paradigm, in which packets are allowed to reach their destinations after (potentially) transiting through intermediate nodes. Such an indirect routing can be particularly useful for switches with large reconfiguration delay. We again identify submodularity, but the
constraints in the maximization are no longer linear and efficient solutions are hard to find. However for a special case, where the schedule is partially fixed, we provide a fast greedy algorithm that is near optimal.

Chapter 4. Next, in Chapter 4 we study live-streaming of multimedia (audio or video) over a p2p network. In this setting, a low-capacity server uploads content in an online-fashion to a small number of clients, who then share it with other clients in the network. The key algorithmic challenge here is to distributedly construct a p2p overlay that (i) effectively utilizes bandwidth to guarantee an optimal streaming rate and (ii) minimizes the diameter of the network to reduce delay. Peers can also arrive or depart from the system at will (churn), and hence the network needs to be self-stabilizing. We propose a novel distribution structure and algorithm that is fully deterministic and yet requires only constant repair time for peer arrivals and departures. The main idea here is to allocate a small portion of the upload capacity to add redundancy to the network. This leads to a loss in the maximum possible streaming rate and delay, but we show that the penalties suffered are small. We also show that for a class of algorithms with redundancies such as in our algorithm, the above delay bound is essentially tight.

Chapter 5. This chapter considers protocols for providing anonymity in Bitcoin's p2p network. Bitcoin and other cryptocurrencies have surged in popularity over the last decade. Although Bitcoin does not claim to provide anonymity for its users, it enjoys a public perception of being a "privacypreserving" financial system. In reality, cryptocurrencies publish users' entire transaction histories in plaintext, albeit under a pseudonym; this is required for transaction validation. Therefore, if a user's pseudonym can be linked to their human identity, the privacy fallout can be significant.

Recently, researchers have demonstrated deanonymization attacks that exploit weaknesses in the Bitcoin network's p2p networking protocols. In particular, the p2p network currently forwards content in a structured way that allows observers to deanonymize users. In this chapter, we redesign the p2p network from first principles with the goal of providing strong, provable anonymity guarantees. We propose a simple networking policy which provides quasi-optimal, network-wide anonymity, with minimal cost to the network's utility. We also discuss practical implementation challenges and propose heuristic solutions.

## CHAPTER 2

## INTERACTIVE COMMUNICATION

Two parties observing random variables $X$ and $Y$ seek to run an interactive protocol $\pi$ with inputs $X$ and $Y$. The parties have access to private as well as shared public randomness. What is the minimum number of bits that they must exchange in order to simulate $\pi$ to within a fixed statistical distance $\epsilon$ ? This question is of importance to the theoretical computer science and information theory communities. On the one hand, it is related closely to the communication complexity problem [24], which in turn is an important tool for deriving lower bounds for computational complexity [25] and for space complexity of streaming algorithms [26]. On the other hand, it is a significant generalization of the classical information theoretic problem of distributed data compression [27], replacing data to be compressed with an interactive protocol and allowing interactive communication as opposed to the usual one-sided communication.

In recent years, it has been argued that the distributional communication complexity for simulating a protocol $\pi$ is related closely to its information complexity ${ }^{1} \mathrm{IC}(\pi)$ defined as follows:

$$
\operatorname{IC}(\pi) \stackrel{\text { def }}{=} I(\Pi \wedge X \mid Y)+I(\Pi \wedge Y \mid X)
$$

where $I(X \wedge Y \mid Z)$ denotes the conditional mutual information between $X$ and $Y$ given $Z(c f$. [10, 28]). For a protocol $\pi$ with communication complexity $\|\pi\|$ (the depth of the binary protocol tree), a simulation protocol requiring $\tilde{\mathcal{O}}(\sqrt{\operatorname{IC}(\pi)\|\pi\|})$ bits of communication was given in [29] and one requiring $2^{\mathcal{O}(\operatorname{IC}(\pi))}$ bits of communication was given in [17]. A general version of the simulation problem was considered in [30], but only bounded round simulation protocols were considered. Interestingly, it was shown in [15] that the

[^1]amortized distributional communication complexity of simulating $n$ copies of a protocol $\pi$ for vanishing simulation error is bounded above by ${ }^{2} \operatorname{IC}(\pi)$. While a matching lower bound was also derived in [15], it is not valid in our context - [15] considered function computation and used a coordinatewise error criterion. Nevertheless, we can readily modify the lower bound argument in [15] and use the continuity of conditional mutual information to formally obtain the required lower bound and thereby a characterization of the amortized distributional communication complexity for vanishing simulation error. Specifically, denoting by $D\left(\pi^{n}\right)$ the distributional communication complexity of simulating $n$ copies of a protocol $\pi$ with vanishing simulation error, we have
$$
\lim _{n \rightarrow \infty} \frac{1}{n} D\left(\pi^{n}\right)=\operatorname{IC}(\pi)
$$

Perhaps motivated by this characterization, or a folklore version of it, the research in this area has focused on designing simulation protocols for $\pi$ requiring communication of length depending on $\operatorname{IC}(\pi)$; the results cited above belong to this category as well. However, the central role of $\operatorname{IC}(\pi)$ in the distributional communication complexity of protocol simulation is far from settled and many important questions remain unanswered. For instance: (a) Does $\operatorname{IC}(\pi)$ suffice to capture the dependence of distributional communication complexity on the simulation error $\varepsilon$ ? (b) Does information complexity have an operational role in simulating $\pi^{n}$ besides being the leading asymptotic term?

The quantity $\operatorname{IC}(\pi)$ plays the same role in the simulation of protocols as $H(X)$ in the compression of $X^{n}[10]$ and $H(X \mid Y)$ in the transmission of $X^{n}$ by the first to the second party with access to $Y^{n}[27]$. The questions raised above have been addressed for these classical problems (cf. [32]). In this chapter, we answer these questions for simulation of interactive protocols. In particular, we answer all these questions in the negative by exhibiting another quantity that plays such a fundamental role and can differ from information complexity significantly. To this end, we introduce the notion of information complexity density of a protocol $\pi$ with inputs $X$ and $Y$ generated from a

[^2]fixed distribution $\mathrm{P}_{X Y}$.
Definition 2.1. Information complexity density. The information complexity density of a private coin protocol $\pi$ is given by the function
$$
\mathrm{ic}(\tau ; x, y)=\log \frac{\mathrm{P}_{\Pi \mid X Y}(\tau \mid x, y)}{\mathrm{P}_{\Pi \mid X}(\tau \mid x)}+\log \frac{\mathrm{P}_{\Pi \mid X Y}(\tau \mid x, y)}{\mathrm{P}_{\Pi \mid Y}(\tau \mid y)}
$$
for all observations $x$ and $y$ of the two parties and all transcripts $\tau$, where $\mathrm{P}_{\Pi X Y}$ denotes the joint distribution of the observation of the two parties and the random transcript $\Pi$ generated by $\pi$.

Note that $\operatorname{IC}(\pi)=\mathbb{E}[\operatorname{ic}(\Pi ; X, Y)]$. We show that it is the $\varepsilon$-tail of the information complexity density ic $(\Pi ; X, Y)$, i.e., the supremum ${ }^{3}$ over values of $\lambda$ such that $\operatorname{Pr}(\mathrm{ic}(\Pi ; X, Y)>\lambda)>\varepsilon$, which governs the communication complexity of simulating a protocol with simulation error less than $\varepsilon$ and not the information complexity of the protocol. The information complexity $\mathrm{IC}(\pi)$ becomes the leading term in communication complexity for simulating $\pi$ only when roughly

$$
\operatorname{IC}(\pi) \gg \sqrt{\operatorname{Var}(\mathrm{ic}(\Pi ; X, Y)) \log (1 / \varepsilon)}
$$

This condition holds, for instance, in the amortized regime considered in [15]. However, the $\varepsilon$-tail of ic $(\Pi ; X, Y)$ can differ significantly from $\operatorname{IC}(\pi)$, the mean of ic $(\Pi ; X, Y)$. The results in this chapter are based on our work [33, 34].

## Summary of Results

Our main results are bounds for distributional communication complexity $D_{\varepsilon}(\pi)$ for $\varepsilon$-simulating a protocol $\pi$. The key quantity in our bounds is the $\varepsilon$-tail $\lambda_{\varepsilon}$ of ic $(\Pi ; X, Y)$.

Lower bound. Our main contribution is a general lower bound for $D_{\varepsilon}(\pi)$. We show that for every private coin protocol $\pi, D_{\varepsilon}(\pi) \gtrsim \lambda_{\varepsilon}$. In fact, this

[^3]bound does not rely on the structure of random variable $\Pi$ and is valid for the more general problem of simulating a correlated random variable.

Prior to this work, there was no lower bound that captured both the dependence on simulation error $\varepsilon$ as well as the underlying probability distribution. On the one hand, the lower bound above yields many sharp results in the amortized regime. It gives the leading asymptotic term in the communication complexity for simulating any sequence of protocols, and not just product protocols. For product protocols, it yields the precise dependence of communication complexity on $\varepsilon$ as well as the exact second-order asymptotic term. On the other hand, it sheds light on the dependence of $D_{\varepsilon}(\pi)$ on $\varepsilon$ even in the single-shot regime. For instance, our lower bound can be used to exhibit an arbitrary separation between $D_{\varepsilon}(\pi)$ and $\operatorname{IC}(\pi)$ when $\varepsilon$ is not fixed. Specifically, evaluating our lower bound for the example protocol in [34] for $\varepsilon=1 / n^{3}$ we get $D_{\varepsilon}(\pi)=\Omega(n)$, which is far more than $2^{\operatorname{IC}(\pi)}$ since $\operatorname{IC}(\pi)=\tilde{\mathcal{O}}\left(n^{-2}\right)$. Remarkably, $[35,36]$ exhibited exponential separation between the distributional communication complexity of computing a function and the information complexity of that function even for a fixed $\varepsilon$, thereby establishing the optimality of the upper bound $D_{\varepsilon}(\pi) \leq \mathcal{O}\left(2^{\text {IC }}(\pi)\right)$ given in [17]. Our simple example shows a much stronger separation between $D_{\varepsilon}(\pi)$ and $\operatorname{IC}(\pi)$, albeit for a vanishing $\varepsilon$.

Upper bound. To establish our asymptotic results, we propose a new simulation protocol, which is of independent interest. For a protocol $\pi$ with bounded rounds of interaction, using our proposed protocol we can show that $D_{\varepsilon}(\pi) \lesssim \lambda_{\varepsilon}$. Much as the protocol of [15], our simulation protocol simulates one round at a time, and thus, the slack in our upper bound does depend on the number of rounds.

Note that while the operative term in the lower bound and the upper bound is the $\varepsilon$-tail of ic $(\Pi ; X, Y)$, the lower bound approaches it from below and the upper bound approaches it from above. It is often the case that these two limits match and the leading term in our bounds coincide. See Figure 2.1 for an illustration of our bounds.

Amortized regime: Second-order asymptotics. Denote by $\pi^{n}$ the $n$ fold product protocol obtained by applying $\pi$ to each coordinate ( $X_{i}, Y_{i}$ ) for inputs $X^{n}$ and $Y^{n}$. Consider the communication complexity $D_{\varepsilon}\left(\pi^{n}\right)$ of $\varepsilon$-simulating $\pi^{n}$ for independent and identically distributed (IID) ( $X^{n}, Y^{n}$ )


Figure 2.1: Illustration of lower and upper bounds for $D_{\varepsilon}(\pi)$.
generated from $\mathrm{P}_{X Y}^{n}$. Using the bounds above, we can obtain the following sharpening of the results of [15]: With $\mathrm{V}(\pi)$ denoting the variance of ic $(\Pi ; X, Y)$,

$$
D_{\varepsilon}\left(\pi^{n}\right)=n \mathrm{IC}(\pi)+\sqrt{n \mathrm{~V}(\pi)} Q^{-1}(\varepsilon)+o(\sqrt{n})
$$

where $Q(x)$ is equal to the probability that a standard normal random variable exceeds $x$ and $Q^{-1}(\varepsilon) \approx \sqrt{\log (1 / \varepsilon)}$. On the other hand, the arguments in $[15]^{4}$ or [30] give us

$$
D_{\varepsilon}\left(\pi^{n}\right) \geq n \operatorname{IC}(\pi)-n \varepsilon[\|\pi\|+\log |\mathcal{X} \| \mathcal{Y}|]-\varepsilon \log (1 / \varepsilon) .
$$

But the precise communication requirement is not less but $\sqrt{n \mathrm{~V}(\pi) \log (1 / \varepsilon)}$ more than $n \mathrm{IC}(\pi)$.

## Proof Techniques

Proof for the lower bound. We present a new method for deriving lower bounds on distributional communication complexity. Our proof relies on a reduction argument that utilizes an $\varepsilon$-simulation to generate an information theoretically secure secret key for $X$ and $Y$ (for a definition of the latter, see $[39,40])$. Heuristically, a protocol can be simulated using fewer bits of communication than its length because of the correlation in the observations $X$ and $Y$. Due to this correlation, when simulating the protocol, the parties agree on more bits (generate more common randomness) than what they communicate. These extra bits can be extracted as an information theoretically secure secret key for the two parties using the leftover hash lemma

[^4](cf. [41, 42]). A lower bound on the number of bits communicated can be derived using an upper bound for the maximum possible length of a secret key that can be generated using interactive communication; the latter was derived recently in [22, 23].

Protocol for the upper bound. We simulate a given protocol one round at a time. Simulation of each round consists of two subroutines: Interactive Slepian-Wolf compression and message reduction by public randomness. The first subroutine is an interactive version of the classical Slepian-Wolf compression [27] for sending $X$ to an observer of $Y$ which is of optimal instantaneous rate. The second subroutine uses an idea that appeared first in [43] (see, also, $[44,45]$ ) and reduces the number of bits communicated in the first by realizing a portion of the required communication by the shared public randomness. This is possible since we are not required to recover a given random variable $\Pi$, but only simulate it to within a fixed statistical distance.

The proposed protocol is closely related to that proposed in [15]. However, there are some crucial differences. The protocol in [15] also uses public randomness to sample each round of the protocol, before transmitting it using an interactive communication of size incremented in steps. However, our information theoretic approach provides a systematic method for choosing this step size. Furthermore, our protocol for sampling the protocol from public randomness is significantly different from that in [15] and relies on randomness extraction techniques. In particular, the protocol in [15] does not attain the asymptotically optimal bounds achieved by our protocol.

Technical approach. While we utilize new, bespoke techniques for deriving our lower and upper bounds, casting our problem in an information theoretic framework allows us to build upon the developments in this classic field. In particular, we rely on the information spectrum approach of Han and Verdú, introduced in the seminal paper [46] (see the textbook [32] for a detailed account). In this approach, the classical measures of information such as entropy and mutual information are viewed as expectations of the corresponding information densities, and the notion of "typical sets" is replaced by sets where these information densities are bounded uniformly. The set of values taken by an information density ( $\operatorname{such}$ as $h(x)=-\log \mathrm{P}_{X}(x)$ ) is called its spectrum. Coding theorems of classical information theory consider IID
repetitions and rely on the so-called asymptotic equipartition property [11] which essentially corresponds to the concentration of spectrums on small intervals. For single-shot problems such concentrations are not available and we have to work with the whole span of the spectrum.

Our main technical contribution is the extension of the information spectrum method to handle interactive communication. Our results rely on the analysis of appropriately chosen information densities and, in particular, will rely on the spectrum of the information complexity density ic $(\Pi ; X, Y)$. As is usually the case, different components of our analysis require bounds on these information densities in different directions, which in turn renders our bounds loose and incurs a gap equal to the length of the corresponding information spectrum. To overcome this shortcoming, we use the spectrum slicing technique of Han $[32]^{5}$ to divide the information spectrum into small portions with information densities closely bounded from both sides. While in our upper bounds spectrum slicing is used to carefully choose the parameters of the protocol, it is required in our lower bounds to identify a set of inputs where a given simulation will require a large number of bits to be communicated.

### 2.1 Protocol Simulation

Two parties observe correlated random variables $X$ and $Y$, with Party 1 observing $X$ and Party 2 observing $Y$, generated from a fixed distribution $\mathrm{P}_{X Y}$ and taking values in finite sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. An interactive protocol $\pi$ (for these two parties) consists of shared public randomness $U$, private randomness ${ }^{6} U_{\mathcal{X}}$ and $U_{\mathcal{Y}}$, and interactive communication $\Pi_{1}, \ldots, \Pi_{r}$. The parties communicate alternatively with Party 1 transmitting in the odd rounds and Party 2 in the even rounds. Specifically, $\Pi_{i}$ is a string of bits determined by the previous transmissions $\Pi_{1}, \ldots, \Pi_{i-1}$ together with ( $X, U_{\mathcal{X}}, U$ ) for odd $i$ and $\left(Y, U_{\mathcal{Y}}, U\right)$ for even $i$. For simplicity, we assume that the realizations of $\Pi_{i}$ constitute a prefix-free code, i.e., no realizations of $\Pi_{i}$ is a prefix of another realization of $\Pi_{i}$. The number of rounds of communication $r$ is a random stopping-time such that the event $\{r=t\}$ is determined by

[^5]the transcript $\Pi_{1}, \ldots, \Pi_{t}$; we denote the overall transcript of the protocol ${ }^{7}$ by $\Pi$. The length of a protocol $\pi,\|\pi\|$, is the maximum number of bits that are communicated in any execution of the protocol.

A random variable $F$ is said to be recoverable by $\pi$ for Party 1 (or Party 2) if $F$ is function of $\left(X, U, U_{\mathcal{X}}, \Pi\right)$ (or $\left(Y, U, U_{\mathcal{Y}}, \Pi\right)$ ).

A protocol with a constant $U$ is called a private coin protocol, with a constant $\left(U_{\mathcal{X}}, U_{\mathcal{Y}}\right)$ is called a public coin protocol, and with $\left(U, U_{\mathcal{X}}, U_{\mathcal{Y}}\right)$ constant is called a deterministic protocol.

When we execute the protocol $\pi$ above, the overall view of the parties consists of random variables ( $Х Ү \Pi \Pi$ ), where the two $\Pi$ s correspond to the transcript of the protocol seen by the two parties. A simulation of the protocol consists of another protocol which generates almost the same view as that of the original protocol. We are interested in the simulation of private coin protocols, using arbitrary ${ }^{8}$ protocols; public coin protocols can be simulated by simulating for each fixed value of public randomness the resulting private coin protocol.

Definition 2.2. $\varepsilon$-Simulation of a protocol. Let $\pi$ be a private coin protocol. Given $0 \leq \varepsilon<1$, a protocol $\pi_{\text {sim }}$ constitutes an $\varepsilon$-simulation of $\pi$ if there exist $\Pi_{\mathcal{X}}$ and $\Pi_{\mathcal{Y}}$, respectively, recoverable by $\pi_{\text {sim }}$ for Party 1 and Party 2 such that

$$
\begin{equation*}
d_{\mathrm{var}}\left(\mathrm{P}_{\Pi \Pi X Y}, \mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} X Y}\right) \leq \varepsilon \tag{2.1}
\end{equation*}
$$

where $d_{\mathrm{var}}(\mathrm{P}, \mathrm{Q})=\frac{1}{2} \sum_{x}\left|\mathrm{P}_{x}-\mathrm{Q}_{x}\right|$ denotes the variational or the statistical distance between P and Q .

Definition 2.3. Distributional communication complexity. The $\varepsilon$ error distributional communication complexity $D_{\varepsilon}\left(\pi \mid \mathrm{P}_{X Y}\right)$ of simulating a private coin protocol $\pi$ is the minimum length of an $\varepsilon$-simulation of $\pi$. The distribution $\mathrm{P}_{X Y}$ remains fixed throughout our analysis; for brevity, we shall abbreviate $D_{\varepsilon}\left(\pi \mid \mathrm{P}_{X Y}\right)$ by $D_{\varepsilon}(\pi)$.

[^6]Problem. Given a protocol $\pi$ and a joint distribution $\mathrm{P}_{X Y}$ for the observations of the two parties, we seek to characterize $D_{\varepsilon}(\pi)$.

Remark 1. Deterministic protocols. Note that a deterministic protocol corresponds to an interactive function, and for such protocols,

$$
d_{\mathrm{var}}\left(\mathrm{P}_{\Pi \Pi X Y}, \mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y} X Y}}\right)=1-\operatorname{Pr}\left(\Pi=\Pi_{\mathcal{X}}=\Pi_{\mathcal{Y}}\right) .
$$

Therefore, a protocol is an $\varepsilon$-simulation of a deterministic protocol if and only if it computes the corresponding interactive function with probability of error less than $\varepsilon$. Furthermore, randomization does not help in this case, and it suffices to use deterministic simulation protocols. Thus, our results below provide tight bounds for distributional communication complexity of interactive functions and, in fact, of all functions which are information theoretically securely computable for the distribution $\mathrm{P}_{X Y}$, since computing these functions is tantamount to computing an interactive function [47] (see, also, [48, 49]).

Remark 2. Compression of protocols. A protocol $\pi_{\text {com }}$ constitutes an $\varepsilon$-compression of a given protocol $\pi$ if it recovers $\Pi_{\mathcal{X}}$ and $\Pi_{\mathcal{Y}}$ for Party 1 and Party 2 such that

$$
\operatorname{Pr}\left(\Pi=\Pi_{\mathcal{X}}=\Pi_{\mathcal{Y}}\right) \geq 1-\varepsilon
$$

Note that randomization does not help in this case either. In fact, for deterministic protocols simulation and compression coincide. In general, however, compression is a more demanding task than simulation and our results show that in many cases, (such as the amortized regime), compression requires strictly more communication than simulation. Specifically, our results for $\varepsilon$-simulation can be modified to get corresponding results for $\varepsilon$-compression by replacing the information complexity density ic $(\tau ; x, y)$ by

$$
h(\tau \mid x)+h(\tau \mid y)=-\log \mathrm{P}_{\Pi \mid X}(\tau \mid x) \mathrm{P}_{\Pi \mid Y}(\tau \mid y)
$$

The proofs remain essentially the same and, in fact, simplify significantly.

### 2.2 Lower, Upper Bounds and Asymptotic Optimality

We derive a lower bound for $D_{\varepsilon}(\pi)$ which applies to all private coin protocols $\pi$ and, in fact, applies to the more general problem of communication complexity of sampling a correlated random variable. For protocols with bounded number of rounds of interaction, i.e., protocols with $r=r\left(X, Y, U, U_{\mathcal{X}}, U_{\mathcal{Y}}\right) \leq r_{\text {max }}$ with probability 1 , we present a simulation protocol which yields upper bounds for $D_{\varepsilon}(\pi)$ of similar form as our lower bounds. In particular, in the asymptotic regime our bounds improve over previously known bounds and are tight. The proofs for results in this section have been presented in Appendix A.

### 2.2.1 Lower Bound

We prove the following lower bound.
Theorem 2.1. Given $0 \leq \varepsilon<1$ and a protocol $\pi$, for arbitrary $0<\eta<1 / 3$

$$
\begin{equation*}
D_{\varepsilon}(\pi) \geq \sup \left\{\lambda: \operatorname{Pr}(\mathrm{ic}(\Pi ; X, Y)>\lambda) \geq \varepsilon+\varepsilon^{\prime}\right\}-\lambda^{\prime} \tag{2.2}
\end{equation*}
$$

where the fudge parameters $\varepsilon^{\prime}$ and $\lambda^{\prime}$ depend on $\eta$ as well as appropriately chosen information spectrums and will be described below in Equations (2.4) and (2.5).
(Proof in Section A.2)
The appearance of fudge parameters such as $\varepsilon^{\prime}$ and $\lambda^{\prime}$ in the bound above is not surprising since the techniques to bound the tail probability of random variables invariably entail such parameters, which are tuned based on the specific scenario being studied. For instance, the Chernoff bound has a parameter that is tuned with respect to the moment generating function of the random variable of interest. More relevant to the problem studied here, such fudge parameters also show up in the evaluation of error probability of single-party non-interactive compression problems (cf. [46, 32]).

When the fudge parameters $\varepsilon^{\prime}$ and $\lambda^{\prime}$ are negligible, the right-side of the bound above is close to $\varepsilon$-tail of $\operatorname{ic}(\Pi ; X, Y)$. Indeed, the fudge parameters turn out to be negligible in many cases of interest. For instance, for the amortized case $\varepsilon^{\prime}$ can be chosen to be arbitrarily small. The parameter $\lambda^{\prime}$ is related
to the length of the interval in which the underlying information densities lie with probability greater than $1-\varepsilon^{\prime}$, the essential length of spectrums. For the amortized case with product protocols, by the central limit theorem the related essential spectrums are of length $\Lambda=\mathcal{O}(\sqrt{n})$ and $\lambda^{\prime}=\log \Lambda$. On the other hand, $\lambda_{\varepsilon}$ is $\mathcal{O}(n)$. Thus, the $\log n$ order fudge parameter $\lambda^{\prime}$ is negligible in this case. Finally, it should be noted that similar fudge parameters are ubiquitous in single-shot bounds; for instance, see [32, Lemma 1.3.2].

Remark 3. The result above does not rely on the interactive nature of $\Pi$ and is valid for simulation of any random variable $\Pi$. Specifically, for any joint distribution $\mathrm{P}_{\Pi X Y}$, an $\varepsilon$-simulation satisfying Equation (2.1) must communicate at least as many bits as the right-side of Equation (2.2), which is roughly equal to the largest value $\lambda_{\varepsilon}$ of $\lambda$ such that $\operatorname{Pr}(\mathrm{ic}(\Pi ; X, Y)>\lambda)>\varepsilon$.

The fudge parameters. The fudge parameters $\varepsilon^{\prime}$ and $\lambda^{\prime}$ in Theorem 2.1 depend on the spectrums of the following information densities:
(i) Information complexity density: This density is described in Definition 2.1 and will play a pivotal role in our results.
(ii) Entropy density of $(X, Y)$ : This density, given by $h(X, Y)=$ $-\log \mathrm{P}_{X Y}(X, Y)$, captures the randomness in the data and plays a fundamental role in the compression of the collective data of the two parties (cf. [32]).
(iii) Conditional entropy density of $X$ given $Y \Pi$ : The conditional entropy density $h(X \mid Y)=-\log \mathrm{P}_{X \mid Y}(X \mid Y)$ plays a fundamental role in the compression of $X$ for an observer of $Y$ [50,32]. We shall use the conditional entropy density $h(X \mid Y \Pi)$ in our bounds.
(iv) Sum conditional entropy density of $(X \Pi, Y \Pi)$ : The sum conditional entropy density is given by $h(X \triangle Y)=-\log \mathrm{P}_{X \mid Y}(X \mid Y) \mathrm{P}_{Y \mid X}(Y \mid X)$ has been shown recently to play a fundamental role in the communication complexity of the data exchange problem [51]. We shall use the sum conditional entropy density $h(X \Pi \triangle Y \Pi)$.
(v) Information density of $X$ and $Y$ is given by $i(X \wedge Y) \stackrel{\text { def }}{=} h(X)-h(X \mid Y)$.

Let $\left[\lambda_{\min }^{(1)}, \lambda_{\max }^{(1)}\right],\left[\lambda_{\min }^{(2)}, \lambda_{\max }^{(2)}\right]$, and $\left[\lambda_{\min }^{(3)}, \lambda_{\max }^{(3)}\right]$ denote the "essential" spectrums of information densities $\zeta_{1}=h(X, Y), \zeta_{2}=h(X \mid Y \Pi)$, and $\zeta_{3}=$
$h(X \Pi \triangle Y \Pi)$, respectively. Concretely, let the tail events $\mathcal{E}_{i}=$ $\left\{\zeta_{i} \notin\left[\lambda_{\min }^{(i)}, \lambda_{\text {max }}^{(i)}\right]\right\}, i=1,2,3$, satisfy

$$
\begin{equation*}
\operatorname{Pr}\left(\mathcal{E}_{1}\right)+\operatorname{Pr}\left(\mathcal{E}_{2}\right)+\operatorname{Pr}\left(\mathcal{E}_{3}\right) \leq \varepsilon_{\text {tail }}, \tag{2.3}
\end{equation*}
$$

where $\varepsilon_{\text {tail }}$ can be chosen to be appropriately small. Further, let $\Lambda_{i}=$ $\lambda_{\text {max }}^{(i)}-\lambda_{\text {min }}^{(i)}, i=1,2,3$, denote the corresponding effective spectrum lengths. The parameters $\varepsilon^{\prime}$ and $\lambda^{\prime}$ in Theorem 2.1 are given by

$$
\begin{align*}
& \varepsilon^{\prime}=\varepsilon_{\text {tail }}+2 \eta, \quad \text { and }  \tag{2.4}\\
& \lambda^{\prime}=2 \log \Lambda_{1} \Lambda_{3}+\log \Lambda_{2}-\log (1-3 \eta)+9 \log 1 / \eta+3 \tag{2.5}
\end{align*}
$$

where $0<\eta<1 / 3$ is arbitrary. If $\Lambda_{i}=0, i=1,2,3$, we can replace it with 1 in the bound above. Thus, our spectrum slicing approach allows us to reduce the dependence of $\lambda^{\prime}$ on spectrum lengths $\Lambda_{i}^{\prime}$ 's from linear to logarithmic.

### 2.2.2 Upper Bound

We prove the following upper bound.
Theorem 2.2. For every $0 \leq \varepsilon<1$ and every protocol $\pi$,

$$
D_{\varepsilon}(\pi) \leq \inf \left\{\lambda: \operatorname{Pr}(\operatorname{ic}(\Pi ; X, Y)>\lambda) \leq \varepsilon-\varepsilon^{\prime}\right\}+\lambda^{\prime}
$$

where the fudge parameters $\varepsilon^{\prime}$ and $\lambda^{\prime}$ depend on the maximum number of rounds of interaction in $\pi$ and on appropriately chosen information spectrums.
(Proof in Section A.3)
Remark 4. In contrast to the lower bound given in the previous section, the upper bound above relies on the interactive nature of $\pi$. Furthermore, the fudge parameters $\varepsilon^{\prime}$ and $\lambda^{\prime}$ depend on the number of rounds, and the upper bound may not be useful when the number of rounds is not negligible compared to $\varepsilon$-tail of the information complexity density. However, we will see that the above upper bound is tight for the amortized regime, even up to the second-order asymptotic term.

The simulation protocol. Our simulation protocol simulates the given
protocol $\pi$ round-by-round, starting from $\Pi_{1}$ to $\Pi_{r}$. Simulation of each round consists of two subroutines: Interactive Slepian-Wolf compression and message reduction by public randomness.

The first subroutine uses an interactive version of the classical SlepianWolf compression [27] (see [50] for a single-shot version) for sending $X$ to an observer of $Y$. The standard (noninteractive) Slepian-Wolf coding entails hashing $X$ to $l$ values and sending the hash values to the observer of $Y$. The number of hash values $l$ is chosen to take into account the worst-case performance of the protocol. However, we are not interested in the worstcase performance of each round, but of the overall multiround protocol. As such, we seek to compress $X$ using the least possible instantaneous rate. To that end, we increase the number of hash values gradually, $\Delta$ at a time, until the receiver decodes $X$ and sends back an ACK. We apply this subroutine to each round $i$, say $i$ odd, with $\Pi_{i}$ in the role of $X$ and $\left(Y, \Pi_{1} \ldots, \Pi_{i-1}\right)$ in the role of $Y$. Similar interactive Slepian-Wolf compression schemes have been considered earlier in different contexts ( $c f$. [52, 53, 54, 55, 51]).

The second subroutine reduces the number of bits communicated in the first by realizing a portion of the required communication by the shared public randomness $U$. Specifically, instead of transmitting hash values of $\Pi_{i}$, we transmit hash values of a random variable $\hat{\Pi}_{i}$ generated in such a manner that some of its corresponding hash bits can be extracted from $U$ and the overall joint distributions do not change by much. Since $U$ is independent of $(X, Y)$, the number $k$ of hash bits that can be realized using public randomness is the maximum number of random hash bits of $\Pi_{i}$ that can be made almost independent of $(X, Y)$, a good bound for which is given by the leftover hash lemma. The overall simulation protocol for $\Pi_{i}$ now communicates $l-k$ instead of $l$ bits. A similar technique for message reduction appears in a different context in [43, 44, 45].

The overall performance of the protocol above is still suboptimal because the saving of $k$ bits is limited by the worst-case performance. To remedy this shortcoming, we once again take recourse to spectrum slicing to ensure that our saving $k$ is close to the best possible for each realization ( $\Pi, X, Y$ ).

Note that our protocol above is closely related to that proposed in [15]. However, the information theoretic form here makes it amenable to techniques such as spectrum slicing, which leads to tighter bounds than those established in [15].

### 2.2.3 Amortized Regime: Second-Order Asymptotics

It was shown in [15] that information complexity of a protocol equals the amortized communication rate for simulating the protocol, i.e.,

$$
\lim _{\varepsilon \rightarrow 0} \lim _{n \rightarrow \infty} \frac{1}{n} D_{\varepsilon}\left(\pi^{n} \mid \mathrm{P}_{X Y}^{n}\right)=\operatorname{IC}(\pi)
$$

where $\mathrm{P}_{X Y}^{n}$ denotes the $n$-fold product of the distribution $\mathrm{P}_{X Y}$, namely the distribution of random variables $\left(X_{i}, Y_{i}\right)_{i=1}^{n}$ drawn IID from $\mathrm{P}_{X Y}$, and $\pi^{n}$ corresponds to running the same protocol $\pi$ on every coordinate ( $X_{i}, Y_{i}$ ). Thus, $\operatorname{IC}(\pi)$ is the first-order term (coefficient of $n$ ) in the communication complexity of simulating the $n$-fold product of the protocol. However, the analysis in [15] sheds no light on finer asymptotics such as the second-order term or the dependence of $D_{\varepsilon}\left(\pi^{n} \mid \mathrm{P}_{X Y}^{n}\right)$ on ${ }^{9} \varepsilon$. On the one hand, it even remains unclear from [15] if a positive $\varepsilon$ reduces the amortized communication rate or not. On the other hand, the amortized communication rate yields only a loose bound for $D_{\varepsilon}\left(\pi^{n} \mid \mathrm{P}_{X Y}^{n}\right)$ for a finite, fixed $n$. A better estimate of $D_{\varepsilon}\left(\pi^{n} \mid \mathrm{P}_{X Y}^{n}\right)$ at a finite $n$ and for a fixed $\varepsilon$ can be obtained by identifying the second-order asymptotic term. Such second-order asymptotics were first considered in [56] and have received a lot of attention in information theory in recent years following [57, 58].

Our lower bound in Theorem 2.1 and upper bound in Theorem 2.2 show that the leading term in $D_{\varepsilon}\left(\pi^{n} \mid \mathrm{P}_{X Y}^{n}\right)$ is roughly the $\varepsilon$-tail $\lambda_{\varepsilon}$ of the random variable

$$
\operatorname{ic}\left(\Pi^{n} ; X^{n}, Y^{n}\right)=\sum_{i=1}^{n} \operatorname{ic}\left(\Pi_{i} ; X_{i}, Y_{i}\right)
$$

a sum of $n$ IID random variables. By the central limit theorem the first-order asymptotic term in $\lambda_{\varepsilon}$ equals

$$
n \mathbb{E}[\mathrm{ic}(\Pi ; X, Y)]=n \operatorname{IC}(\pi),
$$

recovering the result of [15]. Furthermore, the second-order asymptotic term depends on the variance $\mathrm{V}(\pi)$ of ic $(\Pi ; X, Y)$, i.e., on

$$
\mathrm{V}(\pi) \stackrel{\text { def }}{=} \operatorname{Var}[\mathrm{ic}(\Pi ; X, Y)]
$$

[^7]We have the following result.
Theorem 2.3. For every $0<\varepsilon<1$ and every protocol $\pi$ with $\mathrm{V}(\pi)>0$,

$$
D_{\varepsilon}\left(\pi^{n} \mid \mathrm{P}_{X Y}^{n}\right)=n \mathrm{IC}(\pi)+\sqrt{n \mathrm{~V}(\pi)} Q^{-1}(\varepsilon)+o(\sqrt{n}),
$$

where $Q(x)$ is equal to the probability that a standard normal random variable exceeds $x$.
(Proof in Section A.4.1)
As a corollary, we obtain the so-called strong converse.
Corollary 2.1. For every $0<\varepsilon<1$, the amortized communication rate

$$
\lim _{n \rightarrow \infty} \frac{1}{n} D_{\varepsilon}\left(\pi^{n} \mid \mathrm{P}_{X Y}^{n}\right)=\operatorname{IC}(\pi)
$$

Corollary 2.1 implies that the amortized communication complexity of simulating protocol $\pi$ cannot be smaller than its information complexity even if we allow a positive error. Thus, if the length of the simulation protocol $\pi_{\text {sim }}$ is "much smaller" than $n \operatorname{IC}(\pi)$, the corresponding simulation error $\varepsilon=\varepsilon_{n}$ must approach 1. But how fast does this $\varepsilon_{n}$ converge to 1 ? Our next result shows that this convergence is exponentially rapid in $n$.

Theorem 2.4. Given a protocol $\pi$ and an arbitrary $\delta>0$, for any simulation protocol $\pi_{\text {sim }}$ with

$$
\left\|\pi_{\mathrm{sim}}\right\| \leq n[\operatorname{IC}(\pi)-\delta]
$$

there exists a constant $E=E(\delta)>0$ such that for every n sufficiently large, it holds that

$$
d_{\mathrm{var}}\left(\mathrm{P}_{\Pi^{n} \Pi^{n} X^{n} Y^{n}}, \mathrm{P}_{\Pi_{\mathcal{X}}^{n} \Pi_{\mathcal{Y}}^{n} X^{n} Y^{n}}\right) \geq 1-2^{-E n}
$$

(Proof in Section A.4.2)
A similar converse was first shown for the channel coding problem in information theory by Arimoto [59] (see [60, 61] for further refinements of this result), and has been studied for other classical information theory problems as well. To the best of our knowledge, Theorem 2.4 is the first instance of an Arimoto converse for a problem involving interactive communication.

In the theoretical computer science literature, such converse results have been termed direct product theorems and have been considered in the context of the (distributional) communication complexity problem (for computing a given function) $[62,63,64]$. Our lower bound in Theorem 2.1, too, yields a direct product theorem for the communication complexity problem. We state this simple result without proof, since it closely mimics Theorem 2.4. Specifically, given a function $f$ on $\mathcal{X} \times \mathcal{Y}$, by slight abuse of notations and terminologies, let $D_{\varepsilon}(f)=D_{\varepsilon}\left(f \mid \mathrm{P}_{X Y}\right)$ be the communication complexity of computing $f$. As noted in Remark 3, Theorem 2.1 is valid for an arbitrary random variables $\Pi$, and not just an interactive protocol. Then, by following the proof of Theorem 2.4 with $F=f(X, Y)$ replacing $\Pi$ in the application of Theorem 2.1, we get the following direct product theorem.

Theorem 2.5. Given a function $f$ and an arbitrary $\delta>0$, for any function computation protocol $\pi$ computing estimates $F_{\mathcal{X}, n}$ and $F_{\mathcal{Y}, n}$ of $f^{n}$ at the Party 1 and Party 2, respectively, and with length

$$
\begin{equation*}
\|\pi\| \leq n[H(F \mid X)+H(F \mid Y)-\delta] \tag{2.6}
\end{equation*}
$$

there exists a constant $E=E(\delta)>0$ such that for every $n$ sufficiently large, it holds that $\operatorname{Pr}\left(F_{\mathcal{X}, n}=F_{\mathcal{Y}, n}=F^{n}\right) \leq 2^{-E n}$, where $F^{n}=\left(F_{1}, \ldots, F_{n}\right)$ and $F_{i}=f\left(X_{i}, Y_{i}\right), 1 \leq i \leq n$.

Recall that $[15,31]$ showed that the first order asymptotic term in the amortized communication complexity for function computation was shown to equal the information complexity $\operatorname{IC}(f)$ of the function, namely the infimum over $\operatorname{IC}(\pi)$ for all interactive protocols $\pi$ that recover $f$ with 0 error. Ideally, we would like to show an Arimoto converse for this problem, i.e., replace the threshold on the right-side of Equation (2.6) with $n[\operatorname{IC}(f)-\delta]$. The direct product result above is weaker than such an Arimoto converse, and proving the Arimoto converse for the function computation problem is an important future direction. Nevertheless, the simple result above is not comparable with the known direct product theorems in $[62,63]$ and can be stronger in some regimes. ${ }^{10}$

[^8]
### 2.3 Conclusion

In this chapter, we have studied the communication complexity of simulating an interactive protocol between two-parties. We have presented a general lower bound that is the $\varepsilon$-tail of the information complexity density of the protocol. This is done by relating the protocol simulation problem to secret key agreement. We have also presented a new round-by-round simulation protocol, that exploits ideas from Slepian-Wolf compression and message size reduction using public randomness. These lower and upper bounds have then been used to derive tight bounds (up to second-order term) on the distributional communication complexity of simulating an $n$-fold product protocol.

While our results have focused on the theoretical aspects of the canonical two-party problem, it would be interesting to borrow such ideas from the rich area of communication complexity and apply them in practical largescale distributed processing frameworks. This is a topic for future research. A complementary approach (to optimizing communication overhead) for reducing application-level latencies in data centers, is to also keep the networklevel latencies low through a faster network fabric and efficient scheduling. We discuss this in the next chapter.

## CHAPTER 3

## DATA CENTER NETWORKS

Modern data centers are massively scaling up to support demanding applications such as large-scale web services, big data analytics, and cloud computing. The computation in these applications is distributed across tens of thousands of interconnected servers. As the number and speed of servers increases, ${ }^{1}$ providing a fast, dynamic, and economic switching interconnect in data centers constitutes a topical networking challenge. Typically, data center networks use multi-rooted tree designs: the servers are arranged in racks and an Ethernet switch at the top of the rack (ToR) connects the rack of servers to one or more aggregation (or spine) layers. These designs use multiple paths between the ToRs to deliver uniform high bisection bandwidth, and consist of a large number of high-speed electronic packet switches that provide fine-grained switching capabilities but at poor speed/cost ratios.

Recent work has proposed the use of high-speed circuit switches based on optical $[65,66,67]$ or wireless $[68,69,70]$ links to interconnect the ToRs. These architectures enable a dynamic topology tuned to actual traffic patterns, and can provide a much higher aggregate capacity than a network of electronic switches at the same price point, consume significantly less power, and reduce cabling complexity. For instance, Farrington et al. [71] report $2.8 \times, 6 \times$ and $4.7 \times$ lower cost, power, and cabling complexity, respectively, using optical circuit switching relative to a baseline network of electronic switches.

The drawback of circuit switches, however, is that their switching configuration time is much slower than electronic switches. Depending on the specific technology, reconfiguring the circuit switch can take a few milliseconds (e.g., for 3D MEMS optical circuit switches [65, 66, 67]) to tens of microseconds (e.g., for 2D MEMS wavelength-selective switches [20]). Dur-

[^9]ing this reconfiguration period, the circuit switch cannot carry any traffic. By contrast, electronic switches can make per-packet switching decisions at sub-microsecond timescales. This makes the circuit switch suitable for routing stable traffic or bursts of packets (e.g., hundreds to thousands of packets at a time), but not for sporadic traffic or latency sensitive packets. A natural approach is then to have a hybrid circuit/packet switch architecture: the circuit switch can handle traffic flows that have heavy intensity but also require sparse connections, while a lower capacity packet switch handles the complementary (low intensity, but densely connected) traffic flows [66].

With this hybrid architecture, the relatively low intensity traffic is taken care of by the packet switch - switch scheduling here can be done dynamically based on the traffic arrival and is a well-studied topic [72, 73, 74]. On the other hand, scheduling the circuit switch, based on the heavy traffic demand matrix, is still a fundamental unresolved question. Consider an architecture where a centralized scheduler samples the traffic requirements at each of the ToR ports at regular intervals ( $W$, of the order of $100 \mu \mathrm{~s}-1 \mathrm{~ms}$ ), and looks to find the schedule of circuit switch configurations over the interval of $W$ that is "matched" to the traffic requirements. The challenge is to balance the overhead of reconfiguring the circuits with the capability to be flexible and meet the traffic demand requirements.

The centralized scheduler must essentially decide a sequence of matchings between sending and receiving ToRs which the circuit switch then implements. For an optical circuit switch, for instance, the switch realizes the schedule by appropriately configuring its MEMS mirrors. As another example, in a broadcast-select optical ring architecture [75], the ToRs implement the controller's schedule by tuning in to the appropriate wavelength to receive traffic from their matching sender as dictated by the schedule.

Hence, we need a scheduling algorithm that decides the state (i.e., matching) of the circuit switch at each time and also a routing protocol to decide on an appropriate (direct or indirect) route packets can take to reach their destination ToR port. This is a challenging problem and entails making several choices on: (a) number of matchings, (b) choice of matchings (switch configuration), (c) durations of the matchings and (d) the routing protocol, in each interval $W$. Mathematically, this leads to a well-defined optimization problem, albeit involving both combinatorial and real-valued variables. Even special cases of this problem [76] are NP hard to solve exactly.

Central to understanding this scheduling problem is finding a good sparse representation of the traffic matrix - a fundamental algorithmic question in Carathéodory's theorem that has remained largely unanswered so far [77]. Recent papers have proposed heuristic algorithms to address this scheduling problem. In Solstice [78], the authors present a greedy perfect-matching based heuristic for a hybrid electrical-optical switch. Experimental evaluations show Solstice performing well over a simple baseline (where the schedules are provided by a truncated Birkhoff-von Neumann decomposition of the traffic matrix), although no theoretical guarantees are presented. Indirect routing in a distributed setting, but without considerations of configuration switching costs, is studied in another recent work [75]. This chapter is based on results from our work [79].

## Our Contributions

We first focus on routing policies where packets are sent from the source port to the destination port only via a direct link connecting the two ports, leading to direct or single-hop routing.

Approximate Carathéodory's theorem. Our main result here is an approximately optimal, very simple and fast algorithm for computing the switch schedule in each interval. In turn this corresponds to a fast algorithm for computing a sparse approximate representation of a point on the Birkhoff polytope [80]. While Carathéodory's theorem guarantees the existence of such a representation, an efficient algorithm to compute it has remained elusive so far. Our algorithm, which we christen Eclipse, has a performance that is at least half that of optimal for every instance of the traffic demands, and experimentally shows a strict and consistent improvement over the state-of-the-art [78]. A key technical contribution here is the identification of a submodularity structure [81] in the problem, which allows us to make connections between submodular function maximization and the circuit switch scheduling problem with reconfiguration delay.

Indirect routing. Next, we consider routing polices where packets are allowed to reach their destination after (potentially) transiting through many intermediate ports, leading to indirect or multi-hop routing. This class of routing policies is motivated by our observation that if the number of match-
ings is limited, multi-hop routing can exponentially improve the reachability of nodes - a novel benefit of multi-hop routing distinct from the classical and well-known load balancing effects [82, 83, 84]. We again identify submodularity in the problem, but the constraints for this submodular maximization problem are no longer linear and efficient solutions are challenging to find. However, for the important special case where the sequence of switch configurations have already been calculated (and the indirect routing policy has to be decided) we propose a simple and fast greedy algorithm that is nearoptimal universally for all traffic requirements. Detailed simulation results demonstrate strong improvements over direct routing, which are especially pronounced when the switch reconfiguration delays are relatively large.

The chapter is organized as follows. In Section 3.1, the model, framework and the problem objective are formally stated along with a succinct summary of the state of the art. Section 3.2 focuses on direct routing and Section 3.3 on indirect routing. In Section 3.4, we present a detailed evaluation of the proposed algorithms on a variety of traffic inputs. Section 5.6 closes with a brief discussion. Technical aspects of the algorithm and its evaluation, including connections to submodularity and combinatorial optimization problems are deferred to Appendix B.

### 3.1 Switch System and Traffic Model

In this section, we present our model for a hybrid circuit-packet switched network fabric, and formally define our scheduling problem. Our model closely follows [78].

### 3.1.1 Hybrid Switch Model

We consider an $n$-port network where each port is simultaneously connected to a circuit switch and a packet switch as shown in Figure 3.1. A set of nodes are attached to the ports and communicate over the network. The nodes could either be individual servers or top-of-rack switches.

We model the circuit switch as an $n \times n$ crossbar comprising of $n$ input ports and $n$ output ports. At any point in time, each input port can send packets to at most one output port and each output port can receive packets


Figure 3.1: An illustration of our hybrid switch architecture.
from at most one input port over the circuit switch. The circuit switch can be reconfigured to change the input-output connections. We assume that the packets at the input ports are organized in virtual-output-queues [85] (VOQ) which hold packets destined to different output ports.

In practice, the circuit switch is typically an optical switch $[65,66,67] .^{2}$ These switches have a key limitation: changing the circuit configuration imposes a reconfiguration delay during which the switch cannot carry any traffic. The reconfiguration delay can range from few milliseconds to tens of microseconds depending on the technology [20, 86]. This makes the circuit switch suitable for routing stable traffic or bursts of packets (e.g., hundreds to thousands of packets at a time), but not for sporadic traffic or latency sensitive packets. Therefore, hybrid networks also use an (electrical) packet switch to carry traffic that cannot be handled by the circuit switch. The packet switch operates on a packet-by-packet basis, but has a much lower capacity than the circuit switch. For example, the circuit and packet switches might respectively run at 100 Gbps and 10 Gbps per port.

We divide time into slots, with each slot corresponding to a (full-sized) packet transmission time on the circuit switch. We consider a scheduling window of $W \in \mathbb{Z}$ time units. A central controller uses measurements of the aggregated traffic demand between different ports to determine a schedule for the circuit switch at the start of each scheduling window. The schedule comprises of a sequence of configurations and how long to use each configuration (Section 3.1.3). The controller communicates the schedule to the circuit switch, which then follows the schedule for the next scheduling win-

[^10]dow (W) without involving the controller. We assume that the delay for each reconfiguration is $\delta \in \mathbb{Z}$ time units.

### 3.1.2 Traffic Demand

Let $T \in \mathbb{Z}^{n \times n}$ denote the accumulated traffic at the start of a scheduling window. We assume $T$ is a feasible traffic demand, i.e., $T$ is such that $\sum_{j=1}^{n} T(i, j) \leq W$ and $\sum_{i=1}^{n} T(i, j) \leq W$ for all $i, j \in\{1,2, \ldots, n\}$. The $(i, j)$ th entry of $T$ denotes the amount of traffic that is in the VOQ at node $i$ destined for node $j$.

We assume that the controller knows $T .^{3}$ We also assume that non-zero entries in the traffic matrix $T$ are bounded as $2 \delta \leq T(i, j) \leq \epsilon W$ for all $i, j \in[n]: T(i, j)>0$ and some parameter $0<\epsilon<1$. This is a mild condition because traffic between pairs of ports that is small relative to $\delta$ is better served by the packet switch anyway.

Previous measurement studies have shown that the inter-rack traffic in production data centers is sparse $[78,1,87,88]$. Over short periods of time (e.g., 10 s of milliseconds), most nodes communicate with only a small number of other nodes (e.g., few to low tens). Further, in many cases, a large fraction of the traffic is sent by a small fraction of "elephant" flows [87]. While our algorithms and analysis are general, it is important to note that such sparse traffic patterns are necessary for hybrid networks to perform well (especially with larger reconfiguration delay).

### 3.1.3 The Scheduling Problem

Given the traffic demand, $T$, our goal is to compute a schedule that maximizes the total amount of traffic sent over the circuit switch during the scheduling window $W$. This is desirable to minimize the load on the slower packet switch. In general, the scheduling problem involves two aspects:

1. Determining a schedule of circuit switch configurations. The algorithm must determine a sequence of circuit switch configurations: $\left(\alpha_{1}, P_{1}\right)$,

[^11]$\left(\alpha_{2}, P_{2}\right), \ldots,\left(\alpha_{k}, P_{k}\right)$. Here, $\alpha_{i} \in \mathbb{Z}$ denotes the duration of the $i^{\text {th }}$ switch configuration, and $P_{i}$ is an $n \times n$ permutation matrix, where $P_{i}(s, t)=1$ if input port $s$ is connected to output port $t$ in the $i^{\text {th }}$ configuration. For a valid schedule, we must have $\alpha_{1}+\alpha_{2}+\ldots+\alpha_{k}+k \delta \leq W$ since the total duration of the configurations cannot exceed the scheduling window $W$.
2. Deciding how to route traffic. The simplest approach is to use only direct routes over the circuit switch. In other words, each node only sends traffic to destinations to which it has a direct circuit during the scheduling window. Alternatively, we can allow nodes to use indirect routes, where some traffic is forwarded via (potentially multiple) intermediate nodes before being delivered to the destination. Here, the intermediate nodes buffer traffic in their VOQs for transmission over a circuit in a subsequent configuration.

In Section 3.2, we begin by formally defining the problem in the simpler setting with direct routing and developing an algorithm for this case. Then, in Section 3.3, we consider the more general setting with indirect routing.

Remark 1. Prior work [78, 76] has considered the objective of covering the entire traffic demand in the least amount of time. For example, the ADJUST algorithm in [76] takes the traffic demand $T$ as input and computes a schedule $\left(\alpha_{1}, P_{1}\right), \ldots,\left(\alpha_{k}, P_{k}\right)$ such that $\sum_{i=1}^{k} \alpha_{i}+k \delta$ is minimized while $\sum_{i=1}^{k} \alpha_{i} P_{i} \geq T$. Our formulation (and solution) is more general, since an algorithm which maximizes throughput over a given time period can also be used to find the shortest duration to cover the traffic demand (e.g., via binary search).

Remark 2. From a systems viewpoint, the traffic demand estimation can be done either by directly polling ToR switches or end-host NICs [86, 67], or indirectly through applications and flow information [66, 89]. For systems at scale, this represents a non-trivial task (and often taking 100 s of $\mu \mathrm{s}$ [86]) necessitating the need for a window $W$ to compute the schedule (vis-à-vis dynamic policies; see Section 3.1.4).

### 3.1.4 Related Work

We briefly summarize related work on this topic. Scheduling in crossbar switches is a classical and well-studied topic in queuing theory. Tradition-
ally the crossbar has been used to model packet switches where the reconfiguration delay is very small. Hence the scheduling solutions proposed ranging from centralized Birkhoff-von-Neumann decomposition scheduler [72] on one end to the decentralized load-balanced scheduler [90] on the other - did not account for reconfiguration delay. With the proposals on hybrid circuit/packet switching systems [66, 67], simplified models that factor for the reconfiguration delay were considered. A variant of the well-known MaxWeight algorithm is presented in [91] and is shown to be throughput optimal. Fixed-Frame MaxWeight (FFMW) is a frame-based policy proposed in [92] and has good delay performance. However it requires the arrival statistics to be known in advance. A hysteresis-based algorithm that adapts many previously proposed algorithms for crossbar switch scheduling to the case with reconfiguration delay is presented in [93]. All of these works are "dynamic" policies where scheduling decisions are made time-slot by timeslot and the analyses are probabilistic. They also require perfect queue state information at every instant.

Another research direction is to consider "batch" policies [93] in which each computational call returns a schedule for an entire window of time. Research works in this category are often analyzed combinatorially or via real-world system evaluations. Early works often assumed the delay to be either zero [94] or infinity [95, 96]. The infinite delay setting corresponds to a problem where the number of matchings is minimized. However they still require $O(n)$ matchings. In a different context (satellite-switched time-division multiple access), works such as [97] also computed schedules that minimized the number of matchings. Moderate reconfiguration delays are considered in DOUBLE [95] and other algorithms such as [98, 76, 99] that explicitly take reconfiguration delay into account. The algorithm ADJUST [76] minimizes the covering time but still requires around $n$ configurations. All of these algorithms do not benefit from sparse demands and continue to require $O(n)$ configurations [78]. In a complementary approach, [100] considers conditions on the input traffic matrix under which efficient polynomial time algorithms to compute the optimal schedule exists. Yet other approaches have been to introduce speedup [74], or randomization in the algorithms [101], however they do not address the basic optimization problem underlying this scenario head-on.

### 3.2 Direct Routing

The centralized scheduler samples the ToR ports and arrives at the traffic demand (matrix) $T$ to be met in the upcoming slot. In this section, we develop an algorithm, named Eclipse, that takes the traffic demand, $T$, as input and computes a schedule of matchings (circuit configurations) and their durations to maximize throughput over the circuit switch; only direct routing of packets from source to destination ports are allowed here. Eclipse is fast, simple and nearly-optimal in every instance of the traffic matrix $T$. Toward a formal understanding of the notion of optimality, consider the following optimization problem:

$$
\begin{array}{r}
\quad \text { maximize }\left\|\min \left(\sum_{i=1}^{k} \alpha_{i} P_{i}, T\right)\right\|_{1}  \tag{3.1}\\
\text { s.t. } \quad \alpha_{1}+\alpha_{2}+\ldots+\alpha_{k}+k \delta \leq W \\
k \in \mathbb{N}, P_{i} \in \mathcal{P}, \alpha_{i} \geq 0 \forall i \in\{1,2, \ldots, k\},
\end{array}
$$

where $\mathbb{N}=\{1,2, \ldots\}$ and $\mathcal{P}$ is the set of permutation matrices.
This optimization problem is NP-hard [76], and a recent work [78] in the literature has focused on heuristic solutions. Our proposed algorithm has some similarities to the prior work in [78] in that the matchings and their durations are computed successively in a greedy fashion. However, the algorithm is overall quite different in terms of both ideas and details; we uncover and exploit the underlying submodularity [102] structure inherent in the problem to design and analyze the algorithm in a principled way.

We also note that this problem can be viewed as finding permutation matrices $P_{1}, \ldots, P_{k}$ and weightings $\alpha_{1}, \ldots, \alpha_{k}$ such that their weighted sum is a good approximation of the traffic matrix $T$. Carathéodory's theorem applied to the Birkhoff polytope guarantees the existence of such a dual representation; however to date we do not know of an efficient algorithm to compute this representation. A recent work [103] proposes an approximation algorithm, but it relies on an exhaustive search over the vertices of the polytope which can be very slow (moreover in the case of the Birkhoff polytope, the number of vertices is also exponential in the dimension). Our approach does not involve such a search, and is also very efficient in obtaining an approximate Carathéodory expansion.

```
Algorithm 1: A general greedy algorithm template
    Input : Traffic demand \(T\), reconfiguration delay \(\delta\) and scheduling
                    window size \(W\)
    Output: Sequence of matchings \(P_{1}, \ldots, P_{k}\) and their corresponding
            durations \(\alpha_{1}, \ldots, \alpha_{k}\) :
    sch \(\leftarrow\} ; \quad\) // schedule
    \(k \leftarrow 0\);
    \(T_{\text {rem }} \leftarrow T ; \quad\) // traffic remaining
    while \(\sum_{i=1}^{k}\left(\alpha_{i}+\delta\right) \leq W\) do
        \(k \leftarrow k+1 ;\)
        Decide on a duration \(\alpha\) for the matching;
        \(M \leftarrow \operatorname{argmax}_{M \in \mathcal{M}}\left\|\min \left(\alpha M, T_{\text {rem }}\right)\right\|_{1} ;\)
        \(\operatorname{sch} \leftarrow \operatorname{sch} \cup\{(\alpha, M)\}\);
        \(T_{\text {rem }} \leftarrow T_{\text {rem }}-\min \left(\alpha M, T_{\text {rem }}\right) ;\)
    end
    if \(\sum_{i=1}^{k}\left(\alpha_{i}+\delta\right)>W\) then
        \(\operatorname{sch} \leftarrow \operatorname{sch} \backslash\{(\alpha, M)\} ;\)
    end
    \(k \leftarrow k-1 ;\)
```


### 3.2.1 Intuition

Before a formal presentation and analysis of the algorithm, we begin with an intuitive and less-formal approach to how one might solve this optimization problem. Consider greedy algorithms with the template shown in Algorithm 1. The template starts with an empty schedule, and proceeds to add a new matching to the schedule in each iteration. This process continues until the total duration of the matchings exceeds the allotted time budget of $W$, at which point the algorithm terminates and outputs the schedule computed so far. In each iteration, the algorithm first picks the duration of the matching, $\alpha$. It then selects the maximum weight matching in the traffic graph whose edge weights are thresholded by $\alpha$ (i.e., edge weights $>\alpha$ are clipped to $\alpha$ ). The traffic graph is a bipartite graph between $n$ input and $n$ output vertices, with an edge of weight $T(i, j)$ between input node $i$ and output node $j$. It remains to specify how to choose $\alpha$ in each iteration.

Consider an exercise where we vary the matching duration $\alpha$ from 0 to $W$ and compute the maximum weight matching in the thresholded traffic graph for each $\alpha$. For a typical traffic matrix, this results in a curve similar to the solid-blue line in Figure 3.2. Notice that the value of the maximum weight


Figure 3.2: Throughput of max. weight matching as a function of threshold duration. The effective utilization curve of the matchings is also shown.
matching is precisely equal to the sum-throughput that can be achieved in that round of the switch schedule. It is straightforward to see that the maximum weight matching curve has the following properties: (a) it is nondecreasing and (b) piecewise linear. These are explained as follows: when $\alpha$ is very small a lot of the edges in the traffic graph have a weight that is saturated at $\alpha$. Hence it is likely to find a perfect matching with total weight of $n \alpha$. As such the slope of the curve when $\alpha$ is small is $n$. However, as $\alpha$ becomes large there are increasingly fewer edges whose weights are saturated at $\alpha$ and, correspondingly, the slope reduces. When $\alpha$ is so large that all of the edge weights are strictly smaller than $\alpha$, then the value of the maximum weight matching does not change even with any further increase in $\alpha$ and the curve ultimately flattens out.

Two operating points of interest, considering Figure 3.2, are (a) the largest $\alpha$ where the slope of the curve is maximum ( $=n$ in the typical case where every ingress/egress port has traffic) and (b) the smallest $\alpha$ where the value of the maximum weight matching is the largest. These points have been denoted by $\alpha_{1}$ and $\alpha_{2}$ in Figure 3.2 respectively. Setting $\alpha=\alpha_{1}$ is interesting because it results in a matching where the links are all fully utilized. For example, the Solstice algorithm presented in [78] implicitly adopts this operating point. On the other hand, $\alpha=\alpha_{2}$ gives a matching that achieves the largest possible sum-throughput in that round.

However we note that both choices of $\alpha$ are less than ideal for the following reasons. Recall that after every round of switching we incur a delay of $\delta$ time units. As such if the value of $\alpha_{1}$ is small (say comparable to $\delta$ ) in each round, then the number of matchings, and hence the time wasted due to the reconfiguration delay, becomes large. As a concrete example, consider the transpose of the traffic matrix $T_{1}=\left[A_{1}^{t} \mathbf{b}_{1}^{t}\right]$ where $A_{1}$ is a sparse $(n-1) \times n$ matrix and $\mathbf{b}=[2 \delta, 2 \delta, \ldots, 2 \delta, 0,0, \ldots, 0]$ comprises of some $k$ entries of value $2 \delta$ and $n-k$ entries of value 0 . In other words, we are considering an input where a node or a collection of nodes have a large number of small flows to a particular node or vice versa. For such an instance it is clear that if we insist on matchings with $100 \%$ utilized links, then the maximum duration of the matching is $2 \delta$ (i.e., $\alpha_{1}=2 \delta$ ). Thus, continuing the process described in Algorithm 1 results in a sequence of $k$ matchings each of which is only $2 \delta$ time units long. Hence in the worst case (if $k>1 /(3 \delta))$ about one-third of the entire scheduling window is wasted just due to reconfiguration delay limiting the maximum possible throughput to $2 n / 3$. On the other hand, if we had ignored the entries in $\mathbf{b}$, then we could have scheduled just $A_{1}$ achieving a total throughput of $n-2 k \delta \approx n$ for large $n$. We point out that the phenomenon described above happens in a large family of instances, of which $T_{1}$ is a specific example. We also emphasize that such instances are pretty likely to occur in practice; for example, [88, Figure 5-b] shows traffic measurements in a Facebook data center where the interactions between Cache and Web servers lead to traffic matrices having this property.

Similarly for the operating point with $\alpha=\alpha_{2}$, consider the traffic ma$\operatorname{trix} T_{2}=\left[\begin{array}{cc}A_{2} & \mathbf{0} \\ \mathbf{0} & B_{2}\end{array}\right]$ where $A_{2}$ is a sparse $(n-2) \times(n-2)$ matrix and $B_{2}=\left[\begin{array}{ll}0 & 1 \\ 1 & 0\end{array}\right]$. This is a diametrically opposite situation from $T_{1}$ where a small collection of nodes interact only among themselves with no interaction outside. Such a situation occurs, for example, in multi-tenant cloudcomputing data centers [104] where individual tenants run their jobs on small clusters of servers. In such a case, the value of the maximum weight matching can be maximum for a large $\alpha$. For $T_{2}$ the maximum value occurs at $\alpha=1-\delta$ (i.e., $\alpha_{2}=1-\delta$ ), resulting in a schedule with just one matching of duration $1-\delta$ and potentially missing a lot of traffic for $A_{2}$. For example, if $A_{2}$ is uniformly $k$-sparse, we miss out roughly $(k-1) n / k$ units of traffic.

On the other hand, by choosing the duration of the matching to be $1 / k-\delta$ in each step we can achieve a sum throughput of $n-O(\delta) \approx n$.

In scenarios exemplified by $T_{2}$, setting $\alpha=1$ is bad because the utilization of the resulting matching is poor, i.e., a vast majority of the matching links carry only a fraction of their capacity. This can be overcome by insisting that we choose only those matchings with utilization of at least $75 \%$ (say). However, in the case of $T_{1}$ we observe a poor performance in spite of all matchings having a utilization of $100 \%$. The issue in this case is that the duration of the matchings are small compared to the reconfiguration delay. Hence to avoid this scenario we can insist on $\alpha \geq 20 \delta$ (say) in Algorithm 1.

Our first main observation is that both of the above heuristics are captured if we consider the effective utilization of the matchings. We define effective utilization as the ratio $\operatorname{mwm}(\alpha) /(\alpha+\delta)$ where $\operatorname{mwm}(\alpha)$ denotes the value of the maximum weight matching at $\alpha$. This ratio indicates the overall efficiency of a matching by including the reconfiguration delay into the duration. In Figure 3.2 we plot the effective utilization of the matchings as the red-dotted curve. As can be seen there, the effective utilization at both $\alpha_{1}$ and $\alpha_{2}$ is suboptimal. We propose an algorithm that selects $\alpha$ to maximize effective utilization; a detailed description is deferred to Section 4.3.

The justification for selecting matchings according to the above is further reinforced by the submodularity structure of the problem (we discuss submodularity in Section 3.2.2). It turns out that for a certain class of submodular maximization problems with linear packing constraints, greedy algorithms take a form that precisely matches the intuitive thought process above [81]: the proposed intuitively correct algorithm is borne out naturally from submodular combinatorial optimization theory. We briefly recall relevant aspects of submodularity and associated optimization algorithms next.

### 3.2.2 Submodularity

A set function $f: 2^{[n]} \rightarrow \mathbb{R}$ is said to be submodular if it has the following property: for every $A, B \subseteq[n]$ we have $f(A \cup B)+f(A \cap B) \leq f(A)+f(B)$. Alternatively, submodular functions are also defined through the property of decreasing marginal values: for any $S, T$ such that $T \subseteq S \subseteq[n]$ and $j \notin S$,
we have

$$
f(S \cup\{j\})-f(S) \leq f(T \cup\{j\})-f(T)
$$

The difference $f(S \cup\{j\})-f(S)$ is called the incremental marginal value of element $j$ to set $S$ and is denoted by $f_{S}(j)$. For our purpose we will only focus on submodular functions that are monotone and normalized, i.e., for any $S \subseteq T \subseteq[n]$ we have $f(S) \leq f(T)$ and further $f(\})=0$.

Many applications in computer science involve maximizing submodular functions with linear packing constraints. This refers to problems of the form:

$$
\max f(S) \quad \text { s.t. } A \mathbf{x}_{S} \leq b \text { and } S \subseteq[n]
$$

where $A \in[0,1]^{m \times n}, b \in[1, \infty)^{m}$ and $\mathbf{x}_{S}$ denotes the characteristic vector of the set $S$. Each of the $A_{i j}$ 's is a cost incurred for including element $j$ in the solution. The $b_{i}$ 's represent a total budget constraint. A wellknown example of a problem in the above form is the Knapsack problem (the objective function in this case is in fact modular).

With the above background, we formulate the optimization problem under direct routing as one of submodular function maximization. Recall that for any given input traffic matrix $T$, the schedule that is computed is described by a sequence of matchings and corresponding durations. Consider the set $\mathcal{M}$ of all perfect matchings in the complete bipartite graph $K_{n \times n}$ with $n$ nodes in each partite. Then any round in the schedule is simply $(\alpha, P) \in \mathbb{Z} \times \mathcal{M}$. The key observation we make now is to view the schedules as a subset of $\mathbb{Z} \times \mathcal{M}$. Formally, define a switch schedule as any subset $\left\{\left(\alpha_{1}, M_{1}\right), \ldots,\left(\alpha_{k}, M_{k}\right)\right\}$ of $\mathbb{Z} \times \mathcal{M}$. The objective function in our case is the sum-throughput defined as

$$
\begin{equation*}
f\left(\left\{\left(\alpha_{1}, M_{1}\right), \ldots,\left(\alpha_{k}, M_{k}\right)\right\}\right)=\left\|\min \left(\sum_{i=1}^{k} \alpha_{i} M_{i}, T\right)\right\|_{1}, \tag{3.2}
\end{equation*}
$$

where the minimum is taken entrywise and $\|\cdot\|_{1}$ refers to the entrywise $L_{1}$-norm of the matrix. We observe that the function $f$ is submodular.

Theorem 3.1. The function $f: 2^{\mathbb{Z} \times \mathcal{M}} \rightarrow \mathbb{R}$ defined by Equation (3.2) is a monotone, normalized submodular function.

```
Algorithm 2: Eclipse: greedy direct routing algorithm
    Input : Traffic demand \(T\), reconfiguration delay \(\delta\) and scheduling
                    window size \(W\)
    Output: Sequence of matchings \(P_{1}, \ldots, P_{k}\) and their corresponding
            durations \(\alpha_{1}, \ldots, \alpha_{k}\) :
    sch \(\leftarrow\} ; \quad\) // schedule
    \(k \leftarrow 0\);
    \(T_{\text {rem }} \leftarrow T ; \quad\) // traffic remaining
    while \(\sum_{i=1}^{k}\left(\alpha_{i}+\delta\right) \leq W\) do
        \(k \leftarrow k+1 ;\)
        \((\alpha, M) \leftarrow \operatorname{argmax}_{M \in \mathcal{M}, \alpha \in \mathbb{Z}} \frac{\left\|\min \left(\alpha M, T_{\text {rem }}\right)\right\|_{1}}{(\alpha+\delta)} ;\)
        \(\operatorname{sch} \leftarrow \operatorname{sch} \cup\{(\alpha, M)\}\);
        \(T_{\text {rem }} \leftarrow T_{\text {rem }}-\min \left(\alpha M, T_{\text {rem }}\right) ;\)
    end
    if \(\sum_{i=1}^{k}\left(\alpha_{i}+\delta\right)>W\) then
        \(\operatorname{sch} \leftarrow \operatorname{sch} \backslash\{(\alpha, M)\} ;\)
    end
    \(k \leftarrow k-1 ;\)
```


## (Proof in Section B.1.1)

We have established that optical switch scheduling under the sum-throughput metric is a submodular maximization problem. With this, we are ready to present a greedy algorithm that achieves a sum-throughput of at least a constant factor of the optimal algorithm for every instance of the traffic matrix.

### 3.2.3 Algorithm

Algorithm 2 - Eclipse - captures our proposed solution under direct routing. Eclipse takes the traffic matrix $T$, the time window $W$ and reconfiguration delay $\delta$ as inputs, and computes a sequence of matchings and durations as the output. The algorithm proceeds in rounds (the "while loop"), where in each round a new matching is added to the existing sequence of matchings. The sequence terminates whenever the sum of the matching durations exceeds the allocated time window $W$ or whenever the traffic matrix $T$ is fully covered.

Consider any round $t$ in the algorithm; let $\left(\alpha_{1}, M_{1}\right), \ldots,\left(\alpha_{t-1}, M_{t-1}\right)$ denote the schedule computed so far in $t-1$ rounds (stored in variable sch) and let $T_{\text {rem }}(t)$ denote the amount of traffic yet to be routed. The matching that

```
Algorithm 3: Finding the greedy maximum
    Input : Traffic demand \(T\), reconfiguration delay \(\delta\)
    Output: \((\alpha, M) \in \mathbb{Z} \times \mathcal{M}\) such that
            \((\alpha, M)=\operatorname{argmax}_{M \in \mathcal{M}, \alpha \in \mathbb{Z}} \frac{\|\min (\alpha M, T)\|_{1}}{(\alpha+\delta)}\)
    \(H \leftarrow\) distinct entries of \(T\) sorted in ascending order;
    \(i_{\mathrm{lb}} \leftarrow 1\) and \(i_{\text {ub }} \leftarrow\) length \((H)\);
    while \(i_{\mathrm{lb}}<i_{\mathrm{ub}}\) do
        \(i \leftarrow\left(i_{\mathrm{lb}}+i_{\mathrm{ub}}\right) / 2 ;\)
        \(T_{1} \leftarrow \min \{T, H(i)\} ; \quad / /\) thresholding \(T\) to \(H(i)\)
        \(T_{2} \leftarrow \min \{T, H(i+1)\} ;\)
        \(v_{1} \leftarrow\left(\right.\) max. weight matching in \(\left.T_{1}\right) /(H(i)+\delta)\);
        \(v_{2} \leftarrow\) (max. weight matching in \(\left.T_{2}\right) /(H(i+1)+\delta)\);
        if \(v_{1}<v_{2}\) then
            \(i_{\mathrm{lb}} \leftarrow i\);
        else if \(v_{1}>v_{2}\) then
                \(i_{\mathrm{ub}} \leftarrow i\);
        else
        return \(\left(H(i)\right.\), max. weight matching in \(\left.T_{1}\right)\);
        end
    end
```

is selected in the $t$-th round is the one for which utilization - the percentage of the total matching capacity that is actually used - is maximum. Mathematically, we choose an $(\alpha, M)$ pair such that $\frac{\left\|\min \left(\alpha M, T_{\text {rem }}\right)\right\|_{1}}{\alpha+\delta}$ is maximized. In Section B.1.3 we have given a proof that the maximum (for $\alpha$ ) occurs on the support of $T_{\text {rem }}$. Hence this can be easily found by looking at the support of the (sparse) matrix $T_{\text {rem }}$. We also propose a simple binary-search procedure, discussed in Algorithm 3, that finds only a local maximum but performs extremely well in our evaluations (Section 3.4). This process of selecting a matching is repeated in each round until the sum-duration of the matchings exceeds the scheduling window $W$, when the last chosen matching is discarded and the remaining set of matchings are returned. Eclipse is simple and also fast, a fact the following calculation demonstrates.

Complexity. We begin with the complexity of Algorithm 3. Since $i_{\mathrm{ub}}$ is no more than the number of distinct entries of $T$, we have $i_{\text {ub }} \leq n^{2}$. In each iteration, the algorithm only considers entries of $H$ that have indices between $i_{\mathrm{lb}}$ and $i_{\mathrm{ub}}$. However, binary-search halves the effective size of $H$ (i.e., those numbers in $H$ with array indices $i_{\mathrm{lb}}, i_{\mathrm{lb}+1}, \ldots, i_{\mathrm{ub}}$ ), and the num-
ber of iterations of the while loop is bounded by $\log n^{2}=2 \log n$. Within the while loop, computing the maximum weight matching can be done in $O\left(d n^{3 / 2} \log (W \epsilon)\right)$ time (a basic fact of submodular optimization [105, 102]) where $d n$ is the number of edges in bipartite graph formed by $T$ (i.e., $d$ is the average sparsity). Further $(1-\epsilon)$ approximate maximum weight matching can be computed in linear time, e.g. $O\left(d n \epsilon^{-1} \log \epsilon^{-1}\right)$ [106], and efficient implementations in practice have been studied extensively in the literature [107, 108, 109]. Hence the overall time complexity is $O\left(d n^{3 / 2} \log n \log (W \epsilon)\right)$. Now, in Algorithm 2 the number of iterations in the while loop is bounded by $W / \delta$. As such the total complexity of the algorithm is $\tilde{O}\left(d n^{3 / 2} \frac{W}{\delta}\right)$. An exact search over the support of $T_{\text {rem }}$ in the maximization step results in an overall complexity of $\tilde{O}\left(d^{2} n^{5 / 2} \frac{W}{\delta}\right)$.

Approximation guarantee. Since the proposed direct routing algorithm is connected to submodular maximization with linear constraints, we can adapt standard combinatorial optimization techniques to show an approximation factor of $1-1 / e$. Let OPT denote the sum-throughput of the optimal algorithm for given inputs $T, \delta$ and $W$. Let ALG2 denote the sum-throughput achieved by Eclipse. We then have the following.

Theorem 3.2. If the entries of $T$ are bounded by $\epsilon W+\delta$ then Eclipse approximates the optimal algorithm to within a factor of $1-1 / e^{(1-\epsilon)}$, i.e., ALG2 $\geq\left(1-1 / e^{(1-\epsilon)}\right)$ OPT.
(Proof in Section B.1.2)
As a concluding remark, we note that the constant $\epsilon$ in the approximation factor comes from the requirement that $\alpha+\delta \leq \epsilon W$ hold. We observe that this mild technical condition, required to show that Eclipse is a constant factor approximation of the optimal algorithm, has an added implication. Informally, it ensures that no single matching occupies the bulk of the scheduling window.

### 3.3 Indirect Routing

In Section 3.2, we focused on direct routing where packets are forwarded to their destination ports only if a link directly connecting the source port to the destination port appeared in the schedule - this is essentially a "single-hop"
protocol. In this section, we explore allowing packets to be forwarded to (potentially) multiple intermediate ports before arriving at its final destination. In terms of implementing this more involved protocol, we note that there is no extra overhead needed: the destination of any received packet is read first upon reception and since the queues are maintained on a per-destination basis at each ToR port, any received packet can be diverted to the appropriate queue. The key point of allowing indirect routing is the vastly increased range of ports that can be reached from a small number of matchings.

Consider Figure 3.3 which illustrates a six-port network and a sequence of three consecutive matchings in the schedule. With direct routing, port 3 can only forward packets to ports 2,5 and 4 in rounds 1,2 and 3 respectively, i.e., the set of egress ports reachable by port 3 is $\{2,4,5\}$. In the indirect routing framework of this section, port 3 can also forward packets to port 1. This can be achieved by first forwarding the packets to port 2 in the first round where the packets are queued. Then in the second round we let port 2 forward those packets to the destination port 1 . Thus the reachability of the nodes is enhanced by allowing for indirect routing. Indirect routing can also be viewed as "multi-hop" routing.

Traditionally multi-hop routing has been used as a means of load balancing. This is known to be true in the context of networks such as the internet where the benefits of "Valiant load-balancing" are legion [82, 83, 84]. The benefits of load balancing are also well-known in the switching context - a classic example is the two-stage load-balancing algorithm in crossbar switches without reconfiguration delay [110]. The benefit of multi-hop routing in our context is markedly different: the reachability benefits of indirect routing are especially well-suited to the setting where input ports are directly connected to only a few output ports due to the small number of matchings in the scheduling window. In fact, an elementary calculation shows that over a period of $k$ matchings in the schedule, indirect routing can allow a node to forward packets to $O\left(2^{k}\right)$ other nodes, compared to only $O(k)$ nodes possible with direct routing. This is because of the recursion $f(k)=2 f(k-1)+1$ where $f(k)$ denotes the number of nodes reachable by any node in $k$ rounds. If a node (say, node 1) can reach $f(k-1)$ nodes in $k-1$ rounds, then in the $k$-th round (i) there is a new node directly connected to node 1 and (ii) each of the $f(k-1)$ nodes can be connected to a new node. Thus the number of nodes connected to node 1 in the $k$-th round becomes $f(k-1)+(f(k-1)+1)$.


Figure 3.3: Reachability of nodes under multi-hop routing.

Figure 3.3 also illustrates this phenomenon where reachability from node 3 is shown. As a corollary we observe that $O\left(\log _{2} n\right)$ rounds of matchings are sufficient to reach all other nodes in a $n$-port network.

As in the direct routing case, computing the optimal schedule remains a challenging problem. While it is clear that we can achieve a performance at least as good as with direct routing, the gain is different for each instance of the traffic matrix - precisely quantifying the gain in an instance-specific way appears to be challenging. Our main result is that the submodularity property of the objective function continues to hold, provided the variables are considered in an appropriate format. Further, if we restrict some of the variables (the matchings and their durations), then there is also a natural, simple and fast greedy algorithm to compute the switching schedules and routing policies that is approximately optimal for each instance of the traffic matrix. This algorithm serves as a heuristic solution to the more general problem of jointly finding number of switchings, their durations, the switching schedules and routing policies. We present these results, following the same format as in the direct routing section, leaving numerical evaluations to a later section. We follow the model as discussed in Section 3.1.

### 3.3.1 Submodularity of Objective Function

We first adopt an alternative way of describing the switch schedule by specifying the multi-hop path taken by each packet. Such a formulation serves us well in the causal structure of the routed traffic patterns that naturally occur here.

For simplicity let us fix the number of rounds $k$ in the schedule. Consider
a fully connected $k$-round time-layered directed graph $G$ consisting of $k+1$ partites, $V_{0}, V_{1}, \ldots, V_{k}$ (of $n$ nodes each), with nodes in each partite $i$ having directed edges to all the nodes in partite $i+1$. Let $\mathcal{P}$ denote the set of all paths in $G$ that begin at a node in $V_{0}$ and end at a node in $V_{k}$. Any such $p \in \mathcal{P}$ describes a multi-hop route for a packet in the system. If we are able to choose a path for every packet in the traffic matrix $T$, subject to capacity constraints, then we have a valid sequence of switch configurations and routing policy for the schedule. Now, for a set of paths $\left(\beta_{1}, p_{1}\right), \ldots,\left(\beta_{m}, p_{m}\right)$, where $\beta_{i}$ denotes the number of packets sharing the same path $p_{i}$, consider the sum-throughput given by a function $f: 2^{\mathbb{Z} \times \mathcal{P}} \rightarrow \mathbb{Z}$ defined as $f\left(\left\{\left(\beta_{1}, p_{1}\right), \ldots,\left(\beta_{m}, p_{m}\right)\right\}\right) \triangleq$

$$
\begin{equation*}
\sum_{i, j \in[n]} \min \left(\sum_{l=1}^{m} \beta_{l} \mathbf{1}_{\substack{p_{l}(0)=i, p_{l}(k+1)=j}}, T_{i j}\right), \tag{3.3}
\end{equation*}
$$

where $p(0)$ and $p(k+1)$ denote the starting and ending nodes of path $p$ and $\mathbf{1}_{\{\cdot\}}$ is the indicator function. Then the main observation is that $f$ is submodular.

Theorem 3.3. The function $f: 2^{\mathbb{Z} \times \mathcal{P}} \rightarrow \mathbb{Z}$ defined by Equation (3.3) is submodular.

The proof is analogous to Theorem 3.1 and is omitted. So far we have not imposed any restrictions on the set of paths that we choose for the schedule. This can be incorporated in the form of constraints to the problem, thus rephrasing the objective as a constrained submodular maximization problem.

Constraints. Since we can choose arbitrary weighted paths, we need constraints to ensure that
(i) the set of paths form a matching in each round and
(ii) the total duration of the matchings is at most $W-k \delta$.

This can be written mathematically as follows for any subset of weighted paths $\left\{\left(\beta_{1}, p_{1}\right), \ldots,\left(\beta_{m}, p_{m}\right)\right\} \in 2^{\mathbb{Z} \times \mathcal{P}}$ :

$$
\begin{align*}
& \sum_{\substack{e: v \in e, e \in E_{j}}} \mathbf{1}\left\{\sum_{i=1}^{m} \mathbf{1}_{\left\{e \in p_{i}\right\}} \beta_{i}>0\right\} \leq 1 \forall v \in V_{j-1}, j \in[k]  \tag{3.4}\\
& \sum_{\substack{e: v \in e, e \in E_{j}}} \mathbf{1}\left\{\sum_{i=1}^{m} \mathbf{1}_{\left\{e \in p_{i}\right\}} \beta_{i}>0\right\} \leq 1 \forall v \in V_{j}, j \in[k] \tag{3.5}
\end{align*}
$$

$$
\begin{equation*}
\sum_{j=1}^{k}\left(\left(\max _{e \in E_{j}} \sum_{i=1}^{m} \mathbf{1}_{\left\{e \in p_{i}\right\}} \beta_{i}\right)+\delta\right) \leq W \tag{3.6}
\end{equation*}
$$

where $E_{j}$ stands for the edges between $V_{j-1}$ and $V_{j}$ in $G$. Hence we can express the problem as maximization of Equation (3.3) as objective subject to Equations (3.4)-(3.6) as constraints.

However, the key challenge here is that the constraints in Equations (3.4)(3.6) are nonlinear - it is not clear whether an efficient (approximation) algorithm exists. The nonlinearities appear only in the sense of membership tests and a corresponding thresholding function - so it is possible that an efficient nearly optimal greedy algorithm exists, but we leave this study for future work. We do note, however, that for the special case in which the configurations are fixed and we only have to decide on the indirect routing policies, the constraints take on a linear form - in this setting, we are able to construct fast and efficient greedy algorithms. This case represents a composition of direct routing (where switch schedules are computed) and indirect routing (where the multi-hop routing policies are described), and is discussed next.

Multi-hop routing policies. Consider a fixed sequence $M_{1}, \ldots, M_{k}$ of switch configurations and an input traffic demand matrix $T$. Let $G$ denote the time-layered edge-capacitated graph obtained from the sequence of matchings, i.e., $G$ consists of $k+1$ partites $V_{0}, \ldots, V_{k}$ with $n$ nodes each, and $M_{i}$ is the matching between partites $V_{i-1}$ and $V_{i}$. In addition to the matching edges, there are also edges, with unlimited edge capacities, connecting the $j$ th nodes of $V_{i-1}$ and $V_{i}$ for all $j \in[n], i \in[k]$. In this setting, by constraining the total duration of the matchings, we can maximize our required objective by formulating the following linear program (LP) relaxation,

$$
\begin{align*}
& \operatorname{maximize} \sum_{p \in \mathcal{P}} x_{p}  \tag{3.7}\\
& \text { s.t. } \sum_{p \in \mathcal{P}_{i, j}} x_{p} \leq T_{i, j} \quad \forall i, j \in[n] \\
& \sum_{p: e \in p} x_{p} \leq \alpha_{l} \quad \forall e \in M_{l}, \forall l \in[k] \\
& \alpha_{1}+ \ldots+\alpha_{k} \leq W-k \delta \\
& x_{p} \geq 0 \forall p \in \mathcal{P}, \alpha_{i} \geq 0 \forall i \in[k],
\end{align*}
$$

where $\mathcal{P}_{i, j}$ denotes the set of paths starting from a node $i$ in $G$ and terminating at node $j, \mathcal{P}$ denotes the set of all paths $\cup_{i, j \in[n]} \mathcal{P}_{i, j}, x_{p}$ is the flow along path $p$ and $\alpha_{i}$ is the duration of the $i$-th matching. We note that though the present form of the LP can contain an exponential number of variables, equivalent edge-based formulations exist with only a polynomial number of variables and constraints. As such one could use a generic LP solver to obtain the desired schedule. A closely related problem is the classical multicommodity flow problem [111, 112, 113] that was predominantly solved using linear programming based approaches. However, despite many years of research in this direction the proposed algorithms were often too slow even for moderate-sized instances [114]. Since then there has been a renewed effort in providing efficient approximate solutions to the multicommodity flow problem $[115,116]$. The algorithm we present is also a step in this direction, favoring efficiency over exactness of the solution.

### 3.3.2 Algorithm: Eclipse++

Consider the graph $G$ discussed above under a fixed matching, duration sequence $\left(M_{1}, \alpha_{1}\right), \ldots,\left(M_{k}, \alpha_{k}\right)$. Let $R(e)$ denote the capacity of edge $e \in G$. In this setting, the capacity constraints on the end-to-end paths are the sole constraints to the submodular optimization problem - we consider subsets $\left\{\left(\beta_{1}, p_{1}\right), \ldots,\left(\beta_{m}, p_{m}\right)\right\}$ that obey

$$
\begin{equation*}
\sum_{i=1}^{m} \beta_{i} \mathbf{1}_{\left\{e \in p_{i}\right\}} \leq R(e) \quad \forall e \in G \tag{3.8}
\end{equation*}
$$

```
Algorithm 4: Eclipse++: greedy indirect routing algorithm
    Input : Traffic demand \(T\), switch configurations with residue
                    capacities \(R_{1}, \ldots, R_{k}\), update factor \(\lambda\)
    Output: Sequence of paths \(p_{1}, \ldots, p_{m}\) and corresponding weights
            \(\beta_{1}, \ldots, \beta_{m}\)
    sch \(\leftarrow\} ; \quad / /\) schedule
    \(T_{\text {rem }} \leftarrow T\); // traffic remaining
    \(w_{e} \leftarrow 1 / R(e)\) for all \(e \in E\);
    \(m \leftarrow 1 ;\)
    while \(\sum_{e \in E} R(e) w_{e} \leq \lambda\) and \(\left\|T_{\text {rem }}\right\|_{1}>0\) do
        \(\left(\beta_{m}, p_{m}\right) \leftarrow \operatorname{argmax}_{p \in \mathcal{P}, \beta \in \mathbb{Z}} \frac{\min \left(\beta, T_{\text {rem }}(p(0, p(k+1))\right.}{\sum_{e \in E} \beta \mathbf{1}_{\{e \in p\}} w_{e}} ;\)
        \(\operatorname{sch} \leftarrow \operatorname{sch} \cup\left\{\left(\beta_{m}, p_{m}\right)\right\} ;\)
        \(T_{\text {rem }}\left(p_{m}(0), p_{m}(k+1)\right) \leftarrow T_{\text {rem }}\left(p_{m}(0), p_{m}(k+1)\right)-\beta\);
        \(w_{e} \leftarrow w_{e} \lambda^{\beta_{m} \mathbf{1}_{\{e \in p\}} / R(e)} \quad \forall e \in G ;\)
        \(m \leftarrow m+1 ;\)
    end
    if \(\sum_{i=1}^{m-1} \beta_{i} \mathbf{1}_{\left\{e \in p_{i}\right\}} \leq R(e) \quad \forall e \in E\) then
        return sch
    else
        return \(\operatorname{sch} \backslash\left(\beta_{m-1}, p_{m-1}\right)\)
    end
```

Notice that the constraints above have a linear form, and there are a total of $k n$ such constraints (one for each edge). Hence, motivated by [81], which presents a fast and efficient multiplicative weights algorithm for submodular maximization under linear constraints, we propose Eclipse++ in Algorithm 4. The structure of Eclipse++ is similar in spirit to Eclipse (Algorithm 2) in the sense that (a) the algorithm proceeds in rounds, where one new path is added to the schedule in each round and (b) we select a path that offers the greatest utility per unit of cost incurred. However, unlike Algorithm 2 where there was only one linear constraint, we have multiple linear constraints now. This is addressed by assigning weights to the constraints and considering a linear combination of the costs as the true cost in each round. In the following, we describe the salient features of Eclipse++.

Recall the capacity constraints in Equation (3.8) for each edge $e \in G$; let $w_{e}$ denote the weight assigned to the constraint involving edge $e$. We set $w_{e}=1 / R(e)$ for all $e$ initially, i.e., edges with a large capacity are assigned a small weight and vice versa. We can now have another graph $G_{w}$ (with same topology as $G$ ) whose edges are weighted by $w_{e}$. Now, for any path
$p$ the "effective cost" of the path per packet is simply the total cost of $p$ in $G_{w}$. Thus for the path $(\beta, p)$ carrying $\beta$ packets, the effective cost is given by $\sum_{e \in E} \beta w_{e} \mathbf{1}_{e \in p}$. On the other hand, the benefit we get due to adding path $(\beta, p)$ is given by $\min (\beta, T(p(0), p(k+1)))$ where $p(0)$ and $p(k+1)$ stand for the starting and terminating nodes along path $p$. Thus, the ratio $\frac{\min (\beta, T(p(0), p(k+1)))}{\sum_{e \in E} \beta w_{e} 1_{e \in p}}$ denotes the benefit of path $p$ per unit cost incurred. In Algorithm 4 we select $p$ such that the utility per unit cost is maximized.

Now, once we have selected a weighted path $\left(\beta_{1}, p_{1}\right)$ in the first round, we update the weights $w_{e}$ on the edges. This is done as $w_{e} \leftarrow w_{e} \lambda^{\beta_{1} / R(e)}$ for each edge $e \in p$, where $\lambda$ is an input parameter. For the remaining edges the weights remain unchanged. Thus repeating the above iteratively until the while loop condition $\sum_{e \in E} R(e) w_{e} \leq \lambda$ becomes invalid, we get a schedule that is the output of the algorithm. It can also be shown that if the schedule returned sch violates any of the constraints (Equation (3.8)) then it must have happened at the very last iteration and hence we return a schedule with the last added path removed from it. It only remains to show how the maximizer of

$$
\begin{equation*}
\frac{\min \left(\beta, T_{\mathrm{rem}}(p(0, p(k+1))\right.}{\sum_{e \in E} \beta 1_{\{e \in p\}} w_{e}} \tag{3.9}
\end{equation*}
$$

is computed efficiently in each round (first line inside the while loop). Consider the set of shortest paths in $G_{w}$ (smallest $w_{e}$-weighted path) from vertices in $V_{0}$ to vertices in $V_{k}$. Let $p^{*}$ denote the shortest among them. Then by setting $\beta^{*} \leftarrow T_{\mathrm{rem}}\left(p^{*}(0), p^{*}(k+1)\right)$ we claim that Equation (3.9) is maximized. This is because,

$$
\frac{\min \left(\beta, T_{\mathrm{rem}}(p(0, p(k+1))\right.}{\sum_{e \in E} \beta \mathbf{1}_{\{e \in p\}} w_{e}} \leq \frac{\beta}{\sum_{e \in E} \beta \mathbf{1}_{\{e \in p\}} w_{e}} \leq \frac{1}{\min \sum_{e \in E} \mathbf{1}_{\{e \in p\}} w_{e}}
$$

If $T_{\text {rem }}\left(p^{*}(0), p^{*}(k+1)\right)=0$ we proceed to the second smallest shortest path and so on. This allows a very efficient implementation of the internal maximization step.

Approximation guarantee. We show, as in the direct-routing scenario, that Eclipse++ has a constant factor approximation guarantee. Specifically, for a fixed instance of the traffic matrix, let OPT and ALG4 denote the value of the objectives achieved by the optimal algorithm (under fixed matchings,
durations) and Eclipse ++ respectively. Let $\eta:=\max _{i, j \in[n], e \in E} T(i, j) / R(e)$. Then one can show that ALG4 $=\Omega\left(1 /(n k)^{\eta}\right)$ OPT for $\lambda=e^{1 / \eta} n k$; the proof is analogous to the direct-routing case and follows [81, Theorem 1.1]. Further, if $\eta=O\left(\epsilon^{2} / \log (n k)\right)$ for some fixed $\epsilon>0$ then we get a approximation ratio of $(1-\epsilon)(1-1 / e)$ by letting $\lambda=e^{\epsilon /(4 \eta)}$ (using [81, Theorem 1.2]). An interesting regime where this occurs is when the traffic matrices are dense with small skew. For example, we get a constant factor approximation if the sparsity of the traffic matrix grows at least logarithmically fast. This is in stark contrast to direct routing, where sparse matrices generally perform better.

Complexity. The proposed algorithm is simple and fast. In this subsection, we explicitly enumerate the time complexity of the full algorithm and show that the complexity is at most cubic in $n$ and nearly linear in $k$. Let $W$ denote a bound on the total incoming or outgoing traffic for a node. In each iteration of the while loop at least one packet is sent. Therefore there are at most $W$ iterations of the while loop. Now, in each iteration finding the shortest paths between nodes in $V_{0}$ to nodes in $V_{k}$ takes $k n^{2}(\log k+\log n)$ operations using Dijkstra's algorithm [117]. Sorting the computed distances takes $k n^{2}(\log k+\log n)^{2}$ time and at most $n^{2}$ more operations to find a pair $i, j$ such that $T_{\text {rem }}(i, j)>0$. Finally the weights update step takes $k n$ time. Therefore overall it takes $O\left(k n^{2}(\log k+\log n)^{2}\right)$ time per iteration. Hence the time complexity of the complete algorithm is $O\left(W k n^{3}(\log k+\log n)^{2}\right)$.

### 3.3.3 Discussion

In a schedule with a small number of matchings, the packets inherently have to take longer paths (more hops) and thus consume more capacity to reach their destinations. On the other hand, having a large number of matchings introduces capacity wastage in the form of reconfiguration delays. Thus, balancing this trade-off between the number of matchings and the average number of hops packets take to reach their destinations is a key challenge to finding a good matching sequence.

Throughout our discussion so far, we have adopted a model where the scheduling window is of a fixed duration $W$ and has at least one switching operation per window. From a throughput perspective, this inherently
causes a rate loss of at least $\delta / W$ fraction of capacity. Practical optical circuit switches, on the other hand, are often able to retain their previous switching state across adjacent time windows (i.e., without requiring a new configuration at the start of each window). This suggests a natural modification of our current algorithms where in the first round of each window we either (i) retain the last matching of the previous round or (ii) switch to a new matching. We believe such a modification will further improve performance; the precise changes and an evaluation under continuous traffic (such as Bernoulli arrivals) are left for future work.

### 3.4 Evaluation

In this section, we complement our analytical results with numerical simulations to explore the effectiveness of our algorithms and compare them to state-of-the-art techniques in the literature. We empirically evaluate both the direct routing algorithm (Eclipse: Algorithm 2) and the indirect routing algorithm (Eclipse++: Algorithm 4).

Metric. We consider the total fraction of traffic delivered via the circuit switch (sum-throughput) over the duration of a fixed scheduling window as the performance metric throughout this section. Evaluating our algorithms under continuous traffic arrival models remains an important future direction.

Schemes compared. Our experiments compare Eclipse against two existing algorithms for direct routing:
(1) Solstice [78]: This is the state-of-the-art hybrid circuit/packet scheduling algorithm for data centers. The key idea in Solstice is to choose matchings with $100 \%$ utilization. This is achieved by thresholding the demand matrix and selecting a perfect matching in each round. The algorithm presented in [78] tries to minimize the total duration of the window such that the entire traffic demand is covered. In this chapter, we have considered a more general setting where the scheduling window $W$ is constrained. To compare against Solstice in this setting, we truncate its output once the total schedule duration exceeds $W$.
(2) Truncated Birkhoff-von-Neumann (BvN) decomposition: The second algorithm we compare against is the truncated BvN decomposition algori-
thm [80]. BvN decomposition refers to expressing a doubly stochastic matrix as a convex combination of permutation matrices and this decomposition procedure has been extensively used in the context of packet switch scheduling $[78,118,80]$. However BvN decomposition is oblivious to reconfiguration delay and can produce a potentially large $\left(O\left(n^{2}\right)\right)$ number of matchings. Indeed, in our simulations BvN performs poorly.

Indirect routing is relatively new and to the best of our knowledge our work is the first to consider use of indirect routing for centralized scheduling. ${ }^{4}$ In our second set of simulations, we show that the benefits of indirect routing are in addition to the ones obtained from switch configurations scheduling. To this end, we compare Eclipse with Eclipse++ to quantify the additional throughput obtained by performing indirect routing (Algorithm 4) on a schedule that has been (pre)computed using Eclipse.

Traffic demands. We consider two classes of inputs: (a) Single-block inputs and (b) Multi-block inputs (explained in Section 3.4.1). Intuitively, singleblock inputs are matrices which consist of one $n \times n$ "block" that is sparse and skewed, and are similar to the traffic demands evaluated in the Solstice paper [78]. Multi-block inputs, on the other hand, denote traffic matrices that are composed of many sub-matrices each with disparate properties such as sparsity and skew.

Network size. The number of ports is fixed in the range of $50-200$. We find that the relative performances stayed numerically stable over this range as well as for increased number of ports.

### 3.4.1 Direct Routing

While maintaining the sum-throughput as the performance metric, we vary the various parameters of the system model to gauge the performance in different situations.

Single-block inputs. For a single-block input, our simulation setup consists of a network with 100 ports. The link rate of the circuit switch is normalized to 1 , and the scheduling window length is also $1(W=1)$. We consider

[^12]

Figure 3.4: Performance comparison of Eclipse under single-block inputs.
traffic inputs where the maximum traffic to or from any port is bounded by $W$. Further, we let the reconfiguration delay $\delta=W / 100$. The traffic matrix is generated similar to [78] as follows. We assume 4 large flows and 12 small flows to each input or output port. The large flows are assumed to carry $70 \%$ of the link bandwidth, while the small flows deliver the remaining $30 \%$ of the traffic. To do this, we let each flow be represented by a random weighted permutation matrix, i.e., we have

$$
\begin{equation*}
T=\sum_{i=1}^{n_{L}} \frac{c_{L}}{n_{L}} P_{i}+\sum_{i^{\prime}=1}^{n_{S}} \frac{c_{S}}{n_{S}} P_{i^{\prime}}+N \tag{3.10}
\end{equation*}
$$

where $n_{L}\left(n_{S}\right)$ denotes the number of large (small) flows and $c_{L}\left(c_{S}\right)$ denotes the total percentage of traffic carried by the large (small) flows. In this case, we have $n_{L}=4, n_{S}=12$ and $c_{L}=0.7, c_{S}=0.3$. Further, we have added a small amount of noise $N$ - additive Gaussian noise with standard deviation equal to $0.3 \%$ of the link capacity - to the non-zero entries to introduce some perturbation. Each experiment below has been repeated 25 times.

Reconfiguration delay: In Figure 3.4(a) we plot sum-throughput while varying the reconfiguration delay from $W / 3200$ to $4 W / 100$. Eclipse achieves a throughput of at least $90 \%$ for $\delta \leq W / 100$. We observe Eclipse to be consistently better than Solstice although the difference is not pronounced until $\delta>W / 100$. The BvN decomposition algorithm has a large throughput when the reconfiguration delay is small. As $\delta$ increases, its performance gradually worsens.

Skew: We control the skew by varying the ratio of the amount of traffic carried by small and large flows in the input traffic demand matrix $\left(c_{L} / c_{S}\right.$ in

Equation (3.10)). Figure 3.4(b) captures the scenario where the percentage traffic carried by the small flows is varied from 5 to 75 . We observe that Eclipse is very robust to skew variations and is able to consistently maintain a throughput of about $85 \%$. Solstice has a slightly better performance at low skew (when small-flows carry $\sim 75 \%$ of traffic); but overall, is dominated by Eclipse.

Sparsity: Finally, we tested the algorithms' dependence on sparsity and plotted the results in Figure 3.4(c). The total number of flows is varied from 4 to 32 , while fixing the ratio of the number of large to small flows at 1:3. As the input matrix becomes less sparse, the performance of algorithms degrade as expected. However, for Eclipse, the reduction in the throughput is never more than $10 \%$ over the range of sparsity parameters considered. Solstice, on the other hand, is affected much more severely by decreased sparsity.

Multi-block inputs. Next, we consider a more complex traffic model for a 200 node network with block diagonal inputs of the form

$$
T=\left[\begin{array}{ccc}
B_{1} & & \mathbf{0} \\
& \ddots & \\
\mathbf{0} & & B_{m}
\end{array}\right]
$$

where each of the component blocks $B_{1}, B_{2}, \ldots, B_{m}$ can have its own sparsity (number of flows) and skew (fraction of traffic carried by large versus small flows) parameters. The different blocks model the traffic demands of different tenants in a shared data center network such as a public cloud data center. To begin with, we consider inputs with two blocks $T=\left[\begin{array}{cc}B_{1} & \mathbf{0} \\ \mathbf{0} & B_{2}\end{array}\right]$ where $B_{1}$ is a $n_{1} \times n_{1}$ matrix with 4 large flows (carrying $70 \%$ of the traffic) and 12 small flows (carrying $30 \%$ of the traffic) and $B_{2}$ is a $\left(200-n_{1}\right) \times\left(200-n_{1}\right)$ matrix with uniform entries (up to sampling noise).

Size of block: Figure 3.5(a) plots the throughput as the block size of $B_{2}$ is increased from 0 to 70 . We observe a very pronounced difference in the performance of Eclipse and Solstice: Eclipse has roughly 1.5-2× the performance of Solstice. These findings are in tune with the intuition discussed in Section 3.2.1 - the deteriorated performance of Solstice is due its insistence on perfect matchings in each round.


Figure 3.5: Performance comparison of Eclipse under multi-block inputs.

Reconfiguration delay: Figure 3.5(b) plots throughput while varying the reconfiguration delay, for fixed size of $B_{2}$ to be $50 \times 50$. As expected, the throughput of Solstice and Eclipse both degrade as the reconfiguration delay $\delta$ increases. However, Eclipse throughput degrades at a much slower rate than Solstice. The gap between the two is particularly pronounced for $\delta / W \geq 0.02$, a numerical value that is well within range of practical system settings.

Varying numbers of flow: In the final experiment we consider block diagonal inputs with eight blocks of size $25 \times 25$ each. Each block carries $10+\lfloor\sigma *$ $(U-0.5))$ 」 equi-valued flows where $U \sim \operatorname{unif}(0,1)$ and $\sigma$ is a parameter that controls the variation in the number of flows. When increasing $\sigma$ from 0 to 20 we see from Figure 3.5(c) that Eclipse is more or less able to sustain its throughput at close to $80 \%$; whereas Solstice is significantly affected by the variation.

### 3.4.2 Indirect Routing

In this section, we consider a 50 node network with traffic matrices having varying number of large and small flows as before. We compare the performance of the direct routing algorithm and the indirect routing algorithm that is run on the schedule computed by Eclipse. To understand the benefits of indirect routing, we focus on the regime where the reconfiguration delay $\delta / W$ is relatively large and the scheduling window $W$ is relatively long compared to the traffic demand. This regime corresponds to realistic scenarios where the circuit switch is not fully utilized (real data center networks often have low to moderate utilization; e.g, $10-50 \%$ [1]), but the reconfiguration delay is


Figure 3.6: Performance of Eclipse++ and Eclipse. Here Eclipse++ uses the schedule computed by Eclipse.
large. In this setting of relatively large $\delta / W$, switch schedules are forced to have only a small number of matchings, and indirect routing is critical to support (non-sparse) demand matrices. The following experiments numerically demonstrate the added gains of indirect routing.

Sparsity: Figure 3.6(a) considers a demand with five large flows and number of small flows varying from 7 to 49. The large and the small flows each carry $50 \%$ of the traffic. We let $\delta=16 W / 100$ and consider a load of $20 \%$ (i.e., $W=5$, and traffic load at each port is 1 ). We observe that the performance of the Eclipse++ is roughly $10 \%$ better than Eclipse.

Load: As the network load increases (Figure 3.6(b)), we see that indirect routing becomes less effective. This is because at high load, the circuits do not have much spare capacity to support indirect traffic. However, at low to moderate levels of load, indirect routing provides a notable throughput gain over direct routing. For example, we see close to $20 \%$ improvement with Eclipse++ over Eclipse at $15 \%$ load.

Reconfiguration delay: Finally, we observe the effect of $\delta / W$ on throughput by varying $\delta$ from $3 W / 100$ to $21 W / 100$. At smaller values of reconfiguration delay $\delta$ both Eclipse and Eclipse++ are able to achieve near $100 \%$ throughput. With increasing $\delta$ both algorithms degrade with Eclipse++ providing an additional gain of roughly $20 \%$ over Eclipse.

Thus having the capability for indirection can offer a significantly improved performance over direct routing schemes.

### 3.5 Conclusion

We have studied scheduling in hybrid switch architectures with reconfiguration delays in the circuit switch, by taking a fundamental and first-principles approach. The connections to submodular optimization theory allows us to design simple and fast scheduling algorithms and show that they are near optimal - these results hold in the direct routing scenario and indirect routing provided switch configurations are calculated separately. However, we note that the proposed algorithms are not throughput optimal. This is because the matchings selected in Eclipse are chosen by first thresholding the traffic matrix and then choosing a maximum weight matching. Such a scheme is analogous to the maximum size matching studied in [119, 72] and can be shown to achieve strictly less than $100 \%$ throughput. Nevertheless from a practical point-of-view, where traffic loads are often only a fraction of the network capacity, the algorithm is still interesting and could offer potentially a near-optimal delay performance. A systematic study comparing the delay properties of Eclipse to the state-of-the-art is left for future work.

## CHAPTER 4

## PEER-TO-PEER NETWORKS

While data centers are useful in their massive computation and storage capabilities, they do not scale well as content providers - as the number of downloading clients increases the capacity of the data center also has to increase. Peer-to-peer ( p 2 p ) networks are emerging as an increasingly popular modality for content dissemination. Unlike the client-server model, in p 2 p a low-capacity server uploads content to a small number of clients which, together with the other clients, then exchange among themselves. Since each downloading client also simultaneously contributes upload bandwidth, the overall capacity of the network scales with the number of users. As such p2p networks are attractive for bandwidth-intensive applications such as file-sharing, multimedia streaming etc. However, efficient construction and maintenance of the p2p overlay continues to be a challenging problem. In this chapter we study this problem, and propose practical solutions in the context of a canonical p2p application: streaming.

In p2p streaming, a low-capacity server uploads content continuously in an online fashion to the network (for e.g., a live telecast of a public event). This is similar to the rumor spreading problem, in which a rumor from a source node is propagated to all the nodes of an unknown network. However unlike rumor spreading, where only a single rumor is communicated to neighbors over many rounds, in streaming new "rumors" arrive continuously, and need to be forwarded fast and effectively in order to prevent message loss. Limited upload capacity of peers disallows flooding-type message forwarding. Further, peers can arrive or depart the system at will (peer churn), requiring scheduling algorithms to be designed in order to effectively utilize the upload capacity available and to ensure playback continuity with small delay.

A popular method used by some early systems, was to divide the content into multiple substreams and distribute via multicast trees having disjoint interior nodes $[120,121,122,123,124]$. This way any peer could be an inte-
rior node in one multicast tree where it utilizes its upload bandwidth. While trees offer good playback rate and delay, managing trees in a distributed fashion can be very difficult under peer churn. It is known that the complexity of maintaining trees grows with the number of nodes [125, 126]. Hence, random sampling by the peers has commonly been used to help maintain the distribution trees [127]. Another line of work introduced randomness in the construction of the distribution graphs in order to handle the problems associated with peer churn [128]. Whenever a neighboring peer leaves, the peer chooses a new neighbor randomly as its new neighbor. While the distributed nature of the peer pairing makes unstructured networks robust to peer churn, connectivity is sacrificed because some of the peers may not be well-connected due to the inherent randomness. Also, high probability guarantees are often provided only when the number of nodes is large. Yet another drawback is the (potentially large) constants hidden in the order results of many algorithms.

Thus, while structured algorithms promise connectivity to all the peers and have deterministic $O(\log n)$ delay guarantees, a fundamental limitation is their vulnerability to peer churn. Randomized algorithms, on the other hand, provide only probabilistic guarantees for delay, convergence time and connectivity guarantees are weaker. Besides, few algorithms provide a formal guarantee on the transient rates received by the peers (an exception is [128]). A similar trend can be found in the literature on gossip, where a long line of algorithms tried to reduce the spreading time for randomized gossip [129, 130, 131]. It is a priori unclear if deterministic structures could even exist that satisfy the constraints of our model. However, recently a deterministic distributed algorithm for gossip was proposed in [132]. Apart from being faster and more robust than previous randomized algorithms, the deterministic nature has the advantage of running time guarantees holding with certainty instead of with high probability. Inspired by this, we propose and analyze a novel distribution structure for p 2 p streaming that can be maintained deterministically, distributedly by the peers and provides a strong transient rate guarantee under our departure model. The results of this chapter build on our work [133].

## Summary of Results

Our main result is the design of a distribution structure and algorithm that is (i) distributed, (ii) deterministic, (iii) has constant repair time to ensure connectivity and (iv) takes constant time for peer arrival and departure. As far as the author is aware, no other algorithm in the literature has all of the above properties. A key innovation is the introduction of redundancy in the network. Assuming peers are homogeneous with an upload capacity of $C$ each, the delay provided by our algorithm is given by the following.

Theorem 4.1. In the steady-state with $n$ peers in the system, the streaming delay is bounded by $\log _{2}(n+1)+\frac{2 R}{C-R}+\log _{2}\left(1-\frac{R}{C}\right)-2$ for a rate $R \in(0, C)$.
(Proof in Section 4.4)
Here each edge of the p2p network is assumed to contribute 1 unit of delay. The above delay of our algorithm has an additional term of $O(1 /(C-R))$ as compared to the $O(\log n)$ delay of tree-based structures, such as in $[121,127]$. However, the latter do not have constant repair time under churn. Peer departures can cause a sudden loss of transmission links and can lead to loss of connectivity in the multicast graphs. Under such events, the data rate received by some peers can drop considerably until the trees are repaired. Having redundancy in the network helps in this regard in ensuring continuity of streaming without outages under peer churn. The penalties paid due to the introduction of redundant links facilitate: (i) deterministic graph management and (ii) ensure continuity of playback under peer churn events. In our second result, we show that for the amount of redundancy used, the delay guarantee of the algorithm is order optimal.

Theorem 4.2. For structured streaming in which multiple spanning graphs each carry partial flows, the maximum delay across the substream graphs is at least $\log _{\Delta}(n)+\frac{R}{2(C-R)}+\log _{\Delta}\left(\frac{2(C-R)}{R}\right)-c^{\prime}$, where $c^{\prime}=(\Delta-2) \log _{\Delta}\left(\frac{\Delta!}{2}\right)+$ $\log _{e}(\Delta-1)+2$, for a rate $R$, degree bound $\Delta$ and $n \geq \frac{3 R}{C-R}$, if the partial flow graphs have enough capacity redundancy to handle arbitrary node departures.
(Proof in Appendix C)
Notice that any structured broadcast streaming can be viewed as an union of broadcasts of substreams over different spanning graphs, without loss of generality. Thus, we claim that the $R /(C-R)$ term in the delay is fundamental for all algorithms guaranteeing continuity of playback.

Table 4.1: Summary of comparison with previous work for flow based streaming.

| Flow dissemination <br> graph type | Graph maintenance <br> algorithm | References |
| :---: | :---: | :---: |
| Structured | Centralized | $[122,120,124]$ |
|  | Involves randomness | $[121,127,134,135,136,123]$ |
|  | Deterministic | This chapter |
| Unstructured | Random | $[137,138,139,128,140,141]$ |

Transient state characterization. Much of the algorithm is designed to stabilize the transient state. We also guarantee a transient rate equal to the original rate under departure events. The transient rate can be traded for delay as discussed in Section 4.5. Hence apart from providing deterministic guarantees for delay and churn management, the algorithm offers key insights into the continuity aspect of the playback rate.

## Ramifications of Main Result

P2P networks have several interesting applications such as distributed file sharing and computation, social networks, multimedia etc. While our algorithm is designed specifically for the p2p one-to-many streaming problem, several subproblems that we solve are of key interest in-of-themselves.

1. Fast repair. Qualities such as fast addition or removal of nodes, fault tolerance, and efficient routing are desirable in many p2p systems. Designing a distributed, structured overlay that admits a fast repair time has been a challenging problem in the p2p domain. Within the class of structured overlays, even the fastest algorithm takes $O(\log n)$ time for node addition and removal as far as the author is aware (see for example, $[142,143]$ or the literature on Skip graphs [144]). On the other hand, ensuring connectivity after node arrival or departure (even without prior notice) can be performed as quickly as in two time rounds in our overlay. Further the deterministic structure enables design of very efficient routing algorithms for general traffic patterns. For example, all-cast streaming - where every node has a message to be broadcast to all other nodes - can be easily incorporated by our topology.
2. Balancing. The balancing algorithm is another problem of independent interest. For overlays comprised of multiple spanning trees, balancing the trees to ensure a low depth (under node addition/removal) has been hard to achieve. However, with a slight modification in the topology, we show that balancing can be performed fast and efficiently. For example, restoring the structure after a single node insertion/deletion in the steady-state takes only a constant number of rounds. A key idea here is to use the other balanced trees to correct a faulted tree. Thus, being fast, robust and simple, our algorithm can easily be adapted to fit in other p2p applications.

## Related Work

A classical approach in structured streaming involves multicast trees, often with constant-degree nodes [145, 146]. Algorithms in [122, 120, 124] used centralized control. Pastry [136], a routing substrate, was used by the SplitStream algorithm in [121] for tree construction and maintenance. Other distributed lookup protocols have also been proposed in [134, 135]. In [127], an asynchronous distributed algorithm was presented to construct and manage multiple distribution trees by means of random sampling done by the peers. Deterministic overlay topologies have been discussed in [142, 143], however they do not have constant time for peer arrival or departure.

In the other research direction of unstructured p2p networks, where each node communicates with a random subset of other peers, much of the previous theoretical studies of the delay performance have focused primarily on fully connected networks with homogeneous capacities; examples include [137, 138, 139] which make interesting connections between p2p streaming networks, gossip and epidemic models to analyze the maximum streaming delay. In [128], multiple random Hamiltonian cycles were constructed and superposed. The distribution is then done over the union of the cycles. A key idea was to exploit the fact that the superposition of random directed Hamiltonian cycles is an expander with high probability. Additionally, Hamiltonian cycles are easy to maintain in response to peer churn, a fact that was first noted in the case of undirected graphs in [147]. Some other formats of unstructured p2p include mesh based streaming in [140, 141], in which packets were distributed over a randomly constructed mesh. A comparison between
the previous work discussed above and this work has been presented in Table 4.1. We note that the idea of using redundancy to counter transient effects has been observed in other contexts as well $[148,149]$.

System issues. Two key real-world issues with p2p networks are peer churn dynamics and heterogeneity of upload capacities of the participating peers. We model adversarial and arbitrary peer arrival and departures and show tight characterizations of the transient behavior under the operation of our algorithm. In particular, we provide deterministic guarantees toward a very short transient timeline even due to adversarial peer arrivals and departures. As such, more intuition on the transient behavior via simulations on synthetic networks is limited. On the other hand, heterogeneity can be far more nuanced in the real world than that represented by the models in this chapter. For example, user connections in the underlying internet could range from very fast fiber networks to slow WiFi networks. The impact of such wide disparities in upload capacities and latencies on transient behavior is hard to predict analytically or even via synthetic computer simulation - a real world deployment of the algorithms in this chapter would be better suited to capture the performance of the system-at-large (stability, delay and rate guarantees) in the limits of diverse heterogeneity.

### 4.1 Network Model and Steady-State Topology

### 4.1.1 Network Model

The p2p overlay network at each time $t$ is modeled as an undirected node capacitated graph $G(t)=(V(t), E(t))$, in which all the peers have an uniform and download upload capacity of $C$. Peers also have a constant bound $\Delta$ on the number of links allowed. In practice these could be TCP links made over the internet. The time $t$ is slotted. In addition to the upload capability, we let the nodes be able to communicate $O(\log |V(t)|)$ bits of information in any round $t$ as control messages through the edges. We have implicitly assumed that the cost of communicating polylogarithmic bits of control messages by the nodes every round (via existing links) is cheap, compared to breaking and making new connection links. Peers have a constant amount of memory
for storing $M$ addresses (node IDs).
Churn. We model peer arrival and departure as follows. Whenever a peer departs, the node and all the edges connected to it are lost immediately; only the neighbors of the departing peers in $G(V(t), E(t))$ are aware of this event. Peer arrivals, in which a new peer becomes part of the overlay, happens at most one at a time. We also assume communication happens as a flow (or equivalently as time-shared discrete messages) and do not consider network coding in this chapter. Note: For any positive integer $n,[n]$ denotes the set $\{1,2, \ldots, n\}$. The terms peer, node and client have been used interchangeably. In the above model of peer dynamics, with the network changing to adapt to peer arrival or departure, we call a network that is in the process of reconfiguring itself as being in a transient state. A network that is not in the reconfiguration process will be called as being in a steady-state.

### 4.1.2 Steady-State Topology

The main idea of the algorithm is to divide the stream of rate $R$ into several substreams of smaller rates, that are each independently distributed. Such a division of a stream into many substreams has a twofold advantage: (i) it allows for an easier capacity management of peers and (ii) it reduces the overall diameter (delay) of the p 2 p overlay. To distribute individual substreams we propose a novel tree-like topology which has the logarithmic delay of trees, while at the same is also easy to repair and maintain under churn. This is achieved by scaling down the stream rate to less than the optimal (i.e., $R<C)$, and using the leftover capacity $(C-R)$ to aid in the repairing process.

Without loss of generality, assume $C=1$ throughout. We consider streaming at a rate $R=m /(m+1)$ where $m \in \mathbb{Z}$ is a design parameter - a large $m$ guarantees a high rate, but the delay also becomes large and vice versa. For a fixed $m$, the rate of $m /(m+1)$ is divided into $m$ substreams with each substream carrying a rate of $1 /(m+1)$. The remaining upload capacity of $1 /(m+1)$ available at the peers is used to manage the network. We have assumed each user is interested in receiving all the $m$ substreams; in practice coding techniques such as Multiple Description Coding (MDC) [121] could be used to further increase robustness.


Figure 4.1: An example showing the directed graphs (a) $T_{1}$, (b) $U_{1}$, and (c) $G_{1}$ for a network with $n=11$ peers and $m=3$. S is the stream source, while the nodes with indices 1 to 11 denote the peers.

Next, we describe how the substreams are forwarded among peers. Let $G_{i}(t)=\left(V(t), E_{i}(t)\right)$, for $i=1, \ldots, m$, denote the directed graph for broadcasting the $i$ th substream, where $V(t)$ is the set of all peers in the system and $E_{i}(t)$ is the set of links used for the $i$ th substream at time $t$. An edge from $u$ to $v$ in $G_{i}(t)$ means packets from the $i$-th substream are forwarded unidirectionally from peer $u$ to peer $v$ at a rate of $1 /(m+1)$; control messages (of only a few bytes in practice) can be exchanged in both directions from $u$ to $v$ or from $v$ to $u$ along the link. The overall p2p network is a union of the substream networks $G=G_{1} \cup G_{2} \cup \ldots \cup G_{m} .{ }^{1}$ As peers arrive or depart from the system, the topologies of the $G_{i}$ 's adjust dynamically according to our algorithm. These adjustments are strived toward attaining a globally ideal topology for the substream networks. We refer to this ideal topology as the steady-state since in the absence of churn the network converges to this state.

Steady-state. In the steady-state all of the topologies $G_{1}, G_{2}, \ldots, G_{m}$ are symmetric, and as such we begin by describing the topology of $G_{1}$. Let $n$ be the number of peers in the system. Then, $G_{1}$ is the union of two graphs $T_{1}$ and $U_{1}$ as described below. $T_{1}$ is a directed binary tree with its root connected to the server and spans all $n$ nodes. The nodes in $T_{1}$ have an (out-)degree of either one or two, with the degree two nodes occurring above the degree one nodes (i.e., no degree two node occurs in the subtree rooted at a degree one node in $T_{1}$ ). Further, the tree is balanced such that for every

[^13]

Figure 4.2: An example showing a steady-state topology of (a) $G_{1}$, (b) $G_{2}$ and (c) $G_{3}$ for $n=11$ and $R=3 / 4(m=3)$. As before S is the stream source and the nodes 1 to 11 are the peers.
degree two node, the height of the left subtree and the right subtree differs by at most a constant. For a degree two node, the left outgoing edge is called the primary edge and the right outgoing edge is called the secondary edge. The corresponding children are called primary and secondary children respectively. Finally, the chain of degree one nodes leading to a leaf, for every leaf, consists of at least $m-1$ nodes and at most $2 m-2$ nodes (including the leaf node). In Figure 4.1(a), we have illustrated $T_{1}$ for $n=11$ and $m=3$.

Now given $T_{1}, U_{1}$ consists of edges that connect each leaf node of $T_{1}$ to a secondary child in $T_{1}$. Specifically, each leaf node of $T_{1}$ has an edge to the secondary child of the last degree 2 node in the path from the root to the leaf, such that, the secondary child itself does not lie in the path. For the $T_{1}$ shown in Figure 4.1(a), the graph $U_{1}$ has been illustrated in Figure 4.1(b). These edges are essentially redundant since the packets forwarded by the leaf nodes would have been already received at the other end. The overall topology $G_{1}$ is the union of $T_{1}$ and $U_{1}$ and has been shown in Figure 4.1(c). Thus in the steady-state each peer in $G_{1}$ has an upload rate of either $2 /(m+1)$ or $1 /(m+1)$.

The other substream graphs $G_{2}, \ldots, G_{m}$ also have a topology similar to $G_{1}$, except with the location of the peers interchanged. A peer with a degree two in $T_{1}$ has a degree one in the remaining $T_{2}, \ldots, T_{m}$. Conversely, a degree one peer in $T_{1}$ could be a degree two peer in some other $T_{i}$ for $i \neq 1$. Overall, the substream topologies are such that each peer is a degree two node in at most one of the $T_{i}$ 's. This property has been illustrated in Figure 4.2 where we have shown $G_{1}, G_{2}$ and $G_{3}$ for the same example as in Figure 4.1.

Example of peer churn. Clearly the steady-state p2p topology discussed above cannot hold if there are node failure or departures. How to cope under such events forms the content of the rest of this chapter. As a warm-up example to illustrate a basic repair operation, let us consider an event where node 4 departs from the network shown in Figure 4.2. In this case, the substream network $G_{1}$ remains unaffected since peer 4 was only a leaf node. In $G_{2}$, peer 2 which was receiving the stream from peer 4 no longer receives the stream. However, since peer 3 now has an available upload capacity, peer 2 can connect to it. Lastly, in $G_{3}$ both peers 2 and 1 stop receiving the stream. In this case, peer 2 connects to peer 6 , while peer 1 receives the stream from the redundant edge from peer 3. Thus, under the proposed topology, we are able to immediately restore connectivity to the peers in this example. However, the resultant topology is no longer ideal and can cause a large delay in the stream. Thus, and more generally, the following key questions arise: (i) How to keep the peers connected to server? (ii) How to keep the substream topologies balanced under churn, that our algorithm must take care of?

To answer these questions, we next present a few key properties that will serve as a guideline for the peers to restore the network to an ideal steadystate topology. Implementing these guidelines distributedly will form the basis of our algorithm in Section 4.3.

### 4.2 Properties to Be Maintained

Before we state the properties, let us first discuss the notion of a cluster. A cluster refers to a group of peers (ideally between $m$ to $2 m-1$ peers) that are connected to each other and execute protocol as one coherent entity. Peers within a cluster, by way of constantly exchanging control messages, are aware of the state of other peers, i.e., whether they are active, have departed from the network etc., within the cluster. ${ }^{2}$ Also, a peer can belong to only one cluster at any time; if a peer becomes isolated, we consider the peer by itself to form a trivial cluster. Initially, when a peer arrives into the p2p network it can contact any arbitrary peer and become a member of

[^14]its cluster. However at a later point in time, due to the various operations required by the algorithm (as we will see in Section 4.3), the membership of the peer can shift from one cluster to another.

This way of grouping peers into clusters has a twofold benefit: (i) The cumulative upload capacity available can be evenly distributed across the $m$ substream graphs and (ii) Network changes due to node failures, departures within a cluster or arrivals can be accommodated fast and efficiently. To harness these benefits, we specify "rules" in the form of Properties 4.1-4.4 below that the clusters must obey. Each cluster distributedly tries to satisfy the rules using the algorithms discussed in Section 4.3. Also, the rules are enlisted in a decreasing priority order, i.e., a cluster always tries to satisfy the first rule in the list that is violated before moving to the next. The algorithm also ensures that the procedure to satisfy a rule does not invalidate a previously satisfied rule for a cluster.

Let us now state the required rules. We begin with Properties 4.1 and 4.2 that place bounds on the size and capacity within the cluster. Throughout we have assumed a rate of $R=\frac{m}{m+1}$ for $m \in \mathbb{Z}$, and have used $G_{i}, i=1, \ldots, m$, to denote the substream graphs at any time.

Property 4.1. For a peer $v \in V$,
(i) Peer $v$ can belong to only one cluster at any time.
(ii) The size of $v$ 's cluster must be between $m$ and $2 m-1$.

Property 4.2. For a cluster $\mathcal{C}$,
(i) Any peer $v \in \mathcal{C}$ can have out-degree two in at most one substream graph.
(ii) No two peers $u, v \in \mathcal{C}$ can have out-degree two in the same substream graph $G_{i}$ for any $i \in[m]$.
(iii) For each $G_{i}, i \in[m]$, the peers in $\mathcal{C}$ that do not have an out-degree two form a chain.

Note that here out-degree of a peer in a substream graph $G_{i}$ denotes the number of nodes to which the peer forwards the $i$-th substream packets. For example, in Figure 4.2(a) the out-degree of peer 1 is two, while peers 3 and 4 have an out-degree of one. For ease of notation, we will refer to the outdegree of a peer in a substream graph, as simply its degree wherever there is no confusion. Next, to symmetrize the substream graphs and obtain the benefit of fast repair, we have the following Properties 4.3 and 4.4.

Property 4.3. For a cluster $\mathcal{C}$,
(i) There exists a degree two peer $v \in \mathcal{C}$ in every substream graph $G_{i}$.
(ii) In each of the $G_{i}, i \in[m]$, the cluster of the parent of the degree two node from $\mathcal{C}$ and their orientation (i.e., whether they are a primary or secondary child) with respect to their parents must be the same.

Property 4.4. For a cluster $\mathcal{C}$,
(i) For each $G_{i}, i \in[m], v_{i}$ 's secondary child must receive a redundant edge from another peer $u \in \mathcal{C}$, where $v_{i}$ is the degree two peer from $\mathcal{C}$ in $G_{i}$.

As an illustration, consider the network as shown in Figure 4.2 with $R=$ $3 / 4$ and three substream graphs. Let $\mathcal{C}_{1}=\{2,3,4\}, \mathcal{C}_{2}=\{1,5,6\}, \mathcal{C}_{3}=$ $\{7,8,9\}$ and $\mathcal{C}_{4}=\{10,11,12\}$ be the peer clusters. Then, clearly Property 4.1 is satisfied since all the clusters are of size between 3 and 5 . Property 4.2 is also satisfied because each peer within a cluster is a degree two node in exactly one substream. For example, in $\mathcal{C}_{1}$, peer 2 has degree two in $G_{1}$, peer 3 has degree two in $G_{2}$ and peer 4 has degree two in $G_{3}$. Moreover these are the only places where they have a degree of two. To demonstrate Property 4.3 let us consider $\mathcal{C}_{1}$. Clearly each of $G_{1}, G_{2}$ and $G_{3}$ has a degree two peer from $\mathcal{C}_{1}$. Also, in $G_{1}$ peer 2 has peer 1 belonging to $\mathcal{C}_{2}$ as its parent. Peer 2 is also a primary child of its parent. Similarly, in $G_{2}$ and $G_{3}$ peers 3 and 4 respectively are primary children and have parents from the cluster $\mathcal{C}_{2}$. Thus every out-degree two peer in $\mathcal{C}_{2}$ have parents from the same cluster, and also have the same orientation with respect to their parents. This rule is true for the other clusters as well; hence Property 4.3 is also satisfied. As seen in this example, enforcing Properties 4.1-4.3 essentially symmetrizes the substream graphs. This allows a balancing operation performed on one substream graph to be easily mirrored in the remaining substream graphs, as we will see in Section 4.4. Property 4.4 is also satisfied, since each degree two node, such as peer 1 in $G_{1}$ has a secondary child (peer 7 in $G_{1}$ ) that receives a redundant edge (from peer 6).

With Properties 4.1-4.4 in place, our next task is to develop methods to implement these rules distributed in the network. As previously mentioned, we assign a priority to these properties in the order of their appearance, i.e., Property 4.1 has a higher priority than Property 4.2 and so on. At any point in time, a cluster might violate one or more of the properties. Then our algorithm presents a procedure to satisfy the first property to be violated


Figure 4.3: The topology resulting immediately after the departure of peer 1 from the network shown in Figure 4.2.


Figure 4.4: The topology resulting after repairing the clusters in Figure 4.3.
(i.e., Property $i$ such that Properties $1, \ldots, i-1$ are satisfied), while ensuring that the previous properties $(1, \ldots, i-1)$ continue to hold for the cluster. The algorithm also ensures that a cluster implementing the procedure for Property $i$ does not affect the satisfiability of Property $i$ for the other clusters. Thus, continuing this way the network can reach a stable state (if there is no churn) where all the clusters satisfy Properties 4.1-4.4. Before we present the algorithm to enforce Properties 4.1-4.4, let us illustrate their functioning by means of an example.

### 4.2.1 Example

As before, we will consider the network shown in Figure 4.2 and analyze the dynamics under the departure or arrival of a single node.

Peer departure. First let us consider the departure of peer 1 from the


Figure 4.5: The final stable topology after repairing the clusters in Figure 4.4.
network. Just after the peer has departed, the neighbors of node detect the departure and repair the network as shown in Figure 4.3. The clusters in this network are $\{2,3,4\},\{5,6\},\{7,8,9\}$ and $\{10,11,12\}$. Now, for all the clusters except $\{5,6\}$ properties $4.1-4.4$ are satisfied, as one can verify. However for the cluster $\{5,6\}$ even the basic size rule (Property 4.1) is violated. As such, this cluster can merge with a neighboring cluster in order to satisfy the size requirements. Let us suppose this merger happens with the cluster $\{2,3,4\}$ to form the cluster $\{2,3,4,5,6\}$.

Now, we re-evaluate Properties 4.1-4.4 for the updated set of clusters. As before, the clusters $\{7,8,9\}$ and $\{10,11,12\}$ satisfy all the properties. But the newly formed cluster $\{2,3,4,5,6\}$ satisfies Property 4.1 and does not satisfy Property 4.2. This is because both peer 3 and peer 5 have an out-degree two in $G_{2}$ (and also peers 4 and 6 in $G_{3}$; see Figures 4.3(b) and 4.3(c)). However this can be easily corrected since the peers within a cluster are all connected to each other. In this case, peers 3 and 4 can withdraw their secondary links to peers 6 and 5 in $G_{2}$ and $G_{3}$ respectively. This results in a topology as shown in Figure 4.4.

Further, Property 4.2 also requires the degree one peers in any substream graph to form a chain, which is not satisfied in $G_{1}$ by the $\{2,3,4,5,6\}$ cluster. To rectify this, the peers that are outside of the chain can depart from their current locations and insert into the chain. For example, in our case, peer 6 can first depart and insert itself after peer 4 . Then, peer 5 can depart from its position and insert after peer 4. Each time a peer performs a planned departure care is taken to ensure the network remains connected (by linking the parent and children of the departed peer). Hence at the end of the


Figure 4.6: The topology after peer 13 becomes part of the cluster $\{2,3,4,5,6\}$ in Figure 4.5 .
departures by peers 5 and 6 , the secondary edge of peer 2 connects to peer 7 . The resulting topology is shown in Figure 4.5. Now, since all the properties hold for all clusters in this topology, a stable state has been achieved.

Peer arrival. Next, let us consider the arrival of a new peer, say peer 13, in the network shown in Figure 4.5. When a new peer arrives, as a first step, it contacts any other peer currently in the system. In practice, a DNS server can seed new peers with the address information of a small number of peers currently in the system. An arriving peer can then contact an arbitrary peer from this list to get started. In our example, let us suppose the arriving peer 13 contacts peer 2 in Figure 4.5. Then peer 2 immediately includes peer 13 into its cluster ( $\{2,3,4,5,6\}$ ), and also provides the address of an out-degree one peer in the cluster for each substream (e.g. addresses of peers 5,2 and 3 ), so that peer 13 can insert itself into the network and start receiving the stream. This results in the topology shown in Figure 4.6.

However, now the newly formed cluster $\{2,3,4,5,6,13\}$ does not satisfy the size bounds of Property 4.1. This can be resolved by dividing the cluster into two smaller clusters $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$, where $\mathcal{C}_{1}=\{2,5,6\}$ and $\mathcal{C}_{2}=\{3,4,13\}$. Clearly Properties 4.1-4.3 hold for clusters $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$. However, to satisfy Property 4 for $\mathcal{C}_{1}$ we move its degree one chain in $G_{1}, G_{2}$ and $G_{3}$ so that the redundant edge emanates from within the cluster. Further, all of the peers in $\mathcal{C}_{2}$ have a degree one in the substreams. Hence in each substream, one peer from $\mathcal{C}_{2}$ forms a secondary edge to the $\mathcal{C}_{1}$ peer occurring below. This results in a topology as shown in Figure 4.7 and stability is reached.


Figure 4.7: The final topology after splitting clusters in Figure 4.6.

### 4.3 Repairing Algorithm

Next, we present the algorithm to enforce Properties 4.1-4.4. The procedures involve making/breaking a small number of connections and exchanging signaling messages with other peers. The departing process is particularly ubiquitous and hence is discussed first in the following.

Changing locations. Consider a degree one peer $v$ in substream graph $G_{i}$. Let $r$ and $p$ be the parent and child of $v$ respectively in $G_{i}$. Let $s, t$ be any other set of connected peers with an edge from $a$ to $b$. Then, we say " $v$ departs from its current location and inserts into the edge $(a, b)$ " to mean the following. First, $v$ asks for permission from $a, b, r, p$ to perform the operation. If the peers permit, then the edges $(r, v),(v, p)$ are broken and instead the edge $(r, p)$ is made. Similarly, the edge $(a, b)$ is broken and the edges $(a, v),(v, b)$ are made. This has been illustrated in Figure 4.8. The case where $v$ is a degree two peer is similar and has also been illustrated in the Figure 4.8 .

Avoiding cycles. Next, to prevent the formation of cycles in the substream networks, we implement an address forwarding function (Algorithm 5) that constantly forwards peer addresses down the substream graphs. If a leaf peer $v$ in $G_{i}$ receives an address $A$ that is inconsistent with the address of the peer to which $v$ has a redundant edge, then it immediately breaks its existing redundant edge and instead forms an edge to $A$.

The addresses in Algorithm 5 are forwarded as follows. In a substream $G_{i}$, each degree two peer $v$ forwards the address of its secondary child to its primary child. Also, any address received by $v$ (from its parent) is forwarded

(a)

(b)

(c)

Figure 4.8: Illustration showing departure of peer $v$ from one location to another location in the graph.
to its secondary child. If a peer is degree one, then it simply forwards the address received from its parent, to its child. The leaf nodes do not forward the addresses to their children. Notice that each edge has to forward the address of at most one peer. Though this approach could possibly cause the leaf nodes to make and break redundant edges often, it does not affect the stream reception at the nodes.

### 4.3.1 Cluster Size

Let us begin with Property 4.1 which stipulates that peers can belong to only one cluster whose size must be between $m$ and $2 m-1$.

To enforce this, first we consider the case where a peer $v$ belongs to multiple clusters $\mathcal{C}_{1}, \ldots, \mathcal{C}_{k}$. In this case, the peer can arbitrarily choose a cluster $\mathcal{C}_{i}, i \in[k]$, and declare its membership to $\mathcal{C}_{i}$. This is performed by broadcasting an acknowledgment periodically to all the peers in $\mathcal{C}_{i}$. Peer $v$ also terminates any existing cluster-edges to the peers in the other clusters. In the absence of an acknowledgment to peers in the other clusters, $v$ is deemed to have left those clusters.

Next to enforce Property 4.1-(ii), for a cluster $\mathcal{C}$ let us consider the case where $|\mathcal{C}|<m$. In this case, the cluster $\mathcal{C}$ can contact another cluster $\mathcal{C}^{\prime}$ (e.g. a neighbor), and become part of it. The contacting (and also the peer/cluster to be contacted) can be initiated by an arbitrarily chosen representative peer in $\mathcal{C}$. Also the cluster $\mathcal{C}^{\prime}$ that is receiving the merge request from $\mathcal{C}$ always accepts the request. Once the request is accepted, a new cluster $\mathcal{C}^{\prime \prime}=\mathcal{C} \cup \mathcal{C}^{\prime}$ is formed and the peers in $\mathcal{C}^{\prime \prime}$ form a clique by sharing their address information and broadcasting acknowledgments to each other.

Lastly let us consider the case where a cluster $\mathcal{C}$ is such that $|\mathcal{C}| \geq 2 m$. In this case, the peers in $\mathcal{C}$ arbitrarily partition themselves into clusters of size

```
Algorithm 5: Address forwarding in \(G_{i}\)
    Input : \(A\) - address from \(v\) 's parent in \(G_{i}\)
    Output: \(A_{p}\) - address forwarded to \(v\) 's primary child and \(A_{s}\) - address
                    forwarded to \(v\) 's secondary child (if degree two node) in \(G_{i}\)
    if \(v\) is degree two then
        \(A_{s} \leftarrow A ;\)
        \(A_{p} \leftarrow\) address of \(v\) 's secondary child;
        forward \(A_{p}\) and \(A_{s}\) to \(v\) 's primary and secondary children
        respectively;
    else if \(v\) has a non-redundant edge to a child then
        \(A_{p} \leftarrow A ;\)
        forward \(A_{p}\) to \(v\) 's child;
    else
        no action;
    end
```

between $m$ and $2 m-1$. For example, if $|\mathcal{C}|=2 m$, then it can be divided into $\mathcal{C}_{1}$ and $\mathcal{C}_{2}$ consisting of $m$ peers each. The peers in the newly formed clusters (such as $\mathcal{C}_{1}$ or $\mathcal{C}_{2}$ ) in turn form a clique among themselves (if such a clique did not already exist) and exchange acks. As before, the links that are made or broken in this step are low-bandwidth control links, and do not affect the streaming network in any way.

### 4.3.2 Cluster Capacity

In Property 4.2, the first condition requires that for a cluster $\mathcal{C}$, any peer $v \in \mathcal{C}$ has a degree of two in at most one substream graph $G_{i}$. This can be easily enforced, since if the peer $v$ has a degree two in multiple substreams then it can break its secondary edge in all but one arbitrarily chosen substream.

Property 4.2-(ii) is also easily enforced since the peers in a cluster are all connected to each other. If there are multiple degree two peers in a substream graph, then every peer except one can break its secondary edge (i.e., become degree one) in that substream. For example, the peer closest to the server in that substream can retain its secondary edge while the other peer(s) break the edge. The closeness of a peer to the server is measured through a depth message that is passed down the substream network as shown in Algorithm 6.

The final condition that needs to be satisfied in Property 4.2 is that the degree one nodes of a cluster $\mathcal{C}$, in each $G_{i}, i \in[m]$, must form a chain,

```
Algorithm 6: Depth of peer in substream \(G_{i}\)
    Input : \(d_{r}\)-depth of \(v\) 's parent in \(G_{i}\)
    Output: \(d_{v}\) - depth of \(v\) in \(G_{i}\)
    \(d_{v} \leftarrow d_{r}+1 ;\)
    if \(v\) has children in \(G_{i}\) then
        forward \(d_{v}\) to \(v\) 's children in \(G_{i}\);
    else
        no action;
    end
```

i.e., each degree one peer must have an outgoing edge to another degree one peer in $\mathcal{C}$ (except the last peer) in that substream. To enforce this rule, if the degree one peers do not form a chain, they can depart from their locations and insert themselves one after another to form a chain. For example, suppose $S=\left\{v_{1}, \ldots, v_{k}\right\} \subseteq \mathcal{C}$ is the set of degree one peers in $G_{i}$. Then, if the peers in $S$ do not form a chain, it can be made through the following two steps: (i) choose an arbitrary node $v^{*} \in S$, and (ii) for each $v \in S, v \neq v^{*}$ depart from current location and insert between $v^{*}$ and $v^{*}$ s child in $G_{i}$. When a peer departs from its location, it does so according to the departure procedure mentioned at the beginning of this section, to ensure that the network does not get disconnected.

### 4.3.3 Inter Cluster Consistency

Next, let us present a procedure for Property 4.3. Consider any cluster $\mathcal{C}$ that satisfies Properties 4.1 and 4.2. If there are no degree two peers in $\mathcal{C}$ then the first node in the chain of nodes can form a secondary edge to the last node, in each substream graph. Otherwise there exist peers $S=\left\{v_{1}, \ldots, v_{k}\right\} \subseteq \mathcal{C}$ that have a degree two in some substream graph. If $S=\mathcal{C}$ and the clusters, orientation of the parents match, but there exists a parent of a degree two peer in $\mathcal{C}$ that does not satisfy Properties 4.3 -(ii) and (iii) then the peers in $\mathcal{C}$ wait for the latter to be satisfied. Hence, the interesting case is when $S \neq \mathcal{C}$ or there is a mismatch in the parent clusters or orientation.

A simple solution in this case is to first fix a node $v^{*} \in S$ (say the peer that is closest to the server in the substream graph where it is degree two). Let $v^{*}$ be degree two in substream graph $G_{i^{*}}$, and let $\mathcal{C}_{r}^{*}, \Omega^{*} \in\{$ primary, secondary $\}$ be the cluster of $v^{*}$ 's parent, $v^{*}$ 's orientation respectively in $G_{i^{*}}$. Then, for
any other peer $v \in S, v \neq v^{*}$ with a degree two in $G_{i}$, such that $v$ 's parent cluster or orientation in $G_{i}$ differs from $\mathcal{C}_{r}^{*}, \Omega^{*}$, it can depart from its current location in $G_{i}$ and instead insert itself after the primary or secondary edge (according to $\Omega^{*}$ ) of $\mathcal{C}_{r}^{*}$ 's degree two peer in $G_{i}$.

However, in the above method collisions can happen if multiple peer clusters try to have the same parent cluster. To avoid this, we let the parent cluster allow peers from only one cluster to become their primary (secondary) children in $G_{i}, i \in[m]$. Such a cluster can be chosen arbitrarily, for example through a majority vote among its existing children clusters. The chosen primary (secondary) cluster must also acknowledge the choice to the parent cluster.

The insertions happen in the following way. Let us assume that $\mathcal{C}_{r}^{*}$ and $\mathcal{C}$ mutually agree to be the parent and primary child cluster respectively in all of the $G_{i}, i \in[m]$. As before, let $S \subseteq \mathcal{C}$ be the set of peers that have a degree two in some substream graph. Then, we have the following two cases:
Case 1. For $v \in S$, let $v$ have a degree two in $G_{i}$ with secondary child $s$. Let $r \in \mathcal{C}_{r}^{*}$ be the prospective parent in $G_{i}$ that $v$ wants to become a primary child of. Also let $p^{\prime}$ be the existing primary child of $r$. Then, $v$ (i) breaks existing edges with its current parent and primary child and (ii) inserts into edge $\left(r, p^{\prime}\right)$ such that $v$ is a primary child of $r$, and $p^{\prime}$ is a primary child of $v$. Notice that $v$ 's secondary link to $s$ is retained in this process.
Case 2. For $v \notin S$, arbitrarily choose an $i$ such that there is no degree two peer from $\mathcal{C}$ in $G_{i}$. Let $r \in \mathcal{C}_{r}^{*}$ be the prospective parent in $G_{i}$ that $v$ wants to become a primary child of. Then $v$ simply departs from its existing location, and insert into the primary edge of $r$. Then, $v$ can make an secondary edge to its prospective secondary child in $G_{i}$.

The situation is analogous if $v$ wants to become a secondary child of a parent $r$. We again consider two cases:
Case 1. For $v \in S$, let $v$ have a degree two in $G_{i}$ with secondary child $s$. Let $r \in \mathcal{C}_{r}^{*}$ be the prospective parent in $G_{i}$ that $v$ wants to become a secondary child of. Also let $s^{\prime}$ be the existing secondary child of $r$ (if $r$ has no secondary child, it would pull $v$ as discussed above). Then, $v$ (i) breaks existing edges with its current parent and primary child and (ii) inserts into edge ( $r, s^{\prime}$ ) such that $v$ is a secondary child of $r$, and $s^{\prime}$ is a primary child of $v$.
Case 2. For $v \notin S$, arbitrarily choose an $i$ such that there is no degree two peer from $\mathcal{C}$ in $G_{i}$. Let $r \in \mathcal{C}_{r}^{*}$ be the prospective parent in $G_{i}$ that $v$
wants to become a secondary child of. Also let $s^{\prime}$ be the existing secondary child of $r$ (if $r$ has no secondary child, it would pull $v$ as discussed above). Then, $v$ (i) breaks existing edges with its current parent and primary child and (ii) inserts into edge $\left(r, s^{\prime}\right)$ such that $v$ is a secondary child of $r$, and $s^{\prime}$ is a primary child of $v$. Following this, $v$ makes a secondary edge to its prospective secondary child in $G_{i}$.

Finally, let us describe how peers react when a secondary edge is formed to them (in case 2 above). Let $s^{\prime \prime}$ be the prospective secondary child of a peer $v$ in substream $G_{i}$ and let $r^{\prime \prime}$ be the parent of $s^{\prime \prime}$. Then if $v$ forms a secondary edge to $s^{\prime \prime}, s^{\prime \prime}$ breaks the edge to $r^{\prime \prime}$ and forms an edge with $v$.

### 4.3.4 Redundancy

The condition in Property 4.4 requires the terminal leaf nodes of any substream graph to have a redundant edge to the secondary child of the degree two peer in the same cluster. To achieve this, consider a cluster $\mathcal{C}=$ $\left\{v_{1}, v_{2}, \ldots, v_{k}\right\}$ satisfying Properties 4.1-4.3. This implies, for every degree two node $v_{i} \in \mathcal{C}$ in a substream graph, the remaining nodes form a degree one chain in that substream (according to Property 4.2). Without loss of generality, suppose $v_{1}$ is the degree two peer in $G_{1}$; also let $v_{2}, \ldots, v_{k}$ (in that order) form the degree one chain of nodes in $G_{1}$. Let $s$ be the secondary child of $v_{1}$ in $G_{1}$. If $v_{k}$ has a redundant edge to $s$ in $G_{1}$, then the property holds. Otherwise, let $p$ be the peer supplying the redundant edge to $s$ in $G_{1}$ (if such a peer $p$ exists). Then, the chain of nodes $v_{2}, \ldots, v_{k}$ can depart from their location and insert themselves between $p$ and $s$. The addresses of peers $p$ and $s$ are known to $v_{1}$ who can then share it with its cluster. This way, $v_{k}$ becomes the supplier of the redundant edge to $s$ thus satisfying Property 4.4.

The only issue arises if $s$ does not receive a redundant edge from any other peer. In this case, $v_{k}$ can wait for the address to reach a leaf node via the address procedure. Once a redundant edge is made, then the operations described above are performed again.

### 4.3.5 Peer Churn

In Section 4.2.1 we have seen an example of how a single peer departure or arrival is handled by our algorithm. In this section, let us discuss the general case where multiple peers arrive or depart simultaneously. If a peer or a group of peers is completely disconnected from the server due to such an event, their ability to rejoin the network is limited by how many peer addresses they know, that are currently connected to the server in the network. Otherwise, the disconnected peer has to contact a central server (such as a DNS) to get seeded with addresses of a few peers, and essentially appearing to join the network afresh.

Peer arrival. A new peer $u$ wanting to join the p2p network first obtains a list of addresses of peers currently in the system. It then contacts an arbitrary peer, say $v$, from the list and announces its arrival to $v$. The host peer $v$ then immediately includes $u$ within its cluster by assisting $u$ to form a link to every peer in the cluster. This concludes the arrival procedure.

Note that once $u$ becomes part of $v$ 's cluster, the enlarged cluster might no longer satisfy Properties 4.1-4.4. But this is readily handled by our algorithm running at the peers. If multiple peers arrive into the system independently, then each peer can join the system as described above.

Peer departure. Consider the departure of a peer $v$, with out-degree one, from substream graph $G_{1}$. Let $r$ and $p$ be the parent and child of $v$ in $G_{1}$ respectively. Then, if $v$ departs, $p$ contacts $r$ and can resume receiving the stream from it.

Now, suppose peer $v$ has an out-degree of two in $G_{1}$. Let $r, p$ and $s$ be the parent, primary and secondary children of $v$ in $G_{1}$ respectively. Ideally, by Property 4.4, $s$ receives a redundant edge from another peer belonging to the same cluster as $v$. Then if $v$ departs, peer $p$ can contact $r$, while $s$ continues to receive the stream from its redundant edge.

### 4.4 Balancing Algorithm

Previously in Sections 4.2 and 4.3 we have seen how Properties 4.1-4.4 ensure that (i) there are as many degree two nodes in a substream graph as there are clusters and (ii) no degree two node is a child of a degree one node

```
Algorithm 7: Estimating the size of the subtree rooted at \(v\) in \(G_{i}\)
    Input : \(\operatorname{size}(p)\) - size of subtree at \(v\) 's primary child;
            \(\operatorname{size}(s)\) - size of subtree at \(v\) 's secondary child (if \(v\) is
    out-degree two)
    Output: \(\operatorname{size}(v)-\) size of subtree at \(v\)
    if \(v\) is degree two in \(G_{i}\) then
        \(\operatorname{size}(v) \leftarrow \operatorname{size}(p)+\operatorname{size}(s)+1 ;\)
    else if \(v\) has a child then
        \(\operatorname{size}(v) \leftarrow \operatorname{size}(p)+1 ;\)
    else
        \(\operatorname{size}(p) \leftarrow 1 ;\)
    end
    forward \(\operatorname{size}(v)\) to \(v\) 's parent in \(G_{i}\);
```

in a susbtream. The algorithms described there also introduce a level of symmetry to the $G_{i}$ 's. However, the depth of the substream graphs could still be linear in the total number of peers (in the worst case). Therefore, we also describe an algorithm that can balance the graphs. The balancing operation performed at a degree two peer in a substream graph, ensures that the number of peers in its left and right subtrees are roughly the same. Thus executing this operation recursively, starting with the server node and moving downward along the tree, we can ensure that the entire tree is balanced. The procedure also ensures that Properties 4.1-4.4 of the clusters are not violated.

### 4.4.1 Size Estimation

Key to our balancing algorithm is estimating the sizes of subtrees (i.e., number of peers) rooted at degree-two peers. This is easily accomplished by forwarding the tree sizes to the parent peers, who then sum the sizes received from their two children peers. This procedure has been illustrated in Algorithm 7. Using Algorithm 7 degree two peers in a substream graph can estimate the number of peers in their left and right subtrees. However depending on the churn dynamics this estimate could be accurate or inaccurate, since it takes a time proportional to the height of the subtree for the size messages to reach a peer. Nevertheless, for moderate churn (e.g. happening over a timescale of 10 s of seconds or minutes), we can assume the size estimates that peers receive are close to their true values. This also
suggests that the balancing procedure can be performed at a slower timescale matching the churn dynamics. ${ }^{3}$

### 4.4.2 Searching and Rotation

If a degree two peer in a substream graph observes a pronounced disparity in the sizes of its left and right subtrees, the balancing procedure rearranges the subtrees so that their sizes are balanced. Let us illustrate this idea through an example. Consider a substream graph as shown in Figure 4.9(a). In this network, the left subtree of the peer $x$ consists of peers $y, z$ and trees $T_{1}, T_{2}$ and $T_{3}$, while its right subtree consists of the tree $T_{4}$. Suppose the trees are such that the size of the left subtree $\left|T_{1}\right|+\left|T_{2}\right|+\left|T_{3}\right|+2$ is much bigger than the size of the right subtree $\left|T_{4}\right|$. Then, to balance the tree, we perform the following two steps: (i) search and (ii) rotation.

Search. In this first step, the peer $x$ computes a number $\Delta$ of peers to be transferred from the left to the right subtree. In our example, transferring $\Delta=\left(\left|T_{1}\right|+\left|T_{2}\right|+\left|T_{3}\right|+2-\left|T_{4}\right|\right) / 2$ peers can achieve a balance. Following this, $x$ traverses down the secondary edges of its left subtree (such as $y \rightarrow$ $z \rightarrow \operatorname{root}\left(T_{3}\right)$ etc.) until it finds a degree two peer whose right subtree has a size roughly equal to $\Delta$. Let us suppose the right subtree $T_{3}$ of peer $z$ has the required number $\Delta$ of peers. Once this peer has been identified, in the next step we execute a series of link make-break operations that essentially transforms the topology in Figure 4.9(a) to the more balanced Figure 4.9(b).

Rotate. This step involves breaking five links, and then relinking the peers differently. In our example, the links to be broken are $(s \rightarrow x),(x \rightarrow$ $y),(y \rightarrow z),\left(z \rightarrow T_{3}\right)$ and $\left(z \rightarrow T_{4}\right)$. After the links are broken, new links $(s \rightarrow z),(z \rightarrow y),(z \rightarrow x),\left(y \rightarrow T_{2}\right)$ and $\left(x \rightarrow T_{3}\right)$ are formed to yield Figure 4.9(b). This full process can be performed distributedly and involves communication between at most seven peers.

In general, depending on which of the subtrees (right or left) is heavier and whether the root peer $(x)$ is a primary or secondary child or its parent $(s)$, four different cases arise. The algorithm in each of these cases is very similar to the example above, and have been illustrated in Figures 4.10-4.12.

[^15]

Figure 4.9: Balancing the right subtree of peer $s$ through a right rotation.


Figure 4.10: Balancing the left subtree of peer $s$ through a right rotation.

(a)

(b)

Figure 4.11: Balancing the right subtree of peer $s$ through a left rotation.


Figure 4.12: Balancing the left subtree of peer $s$ through a left rotation.

We also note that if the peer clusters obey Properties 4.1-4.4 before balancing, then they obey them after balancing too. Hence, we can also execute the balancing procedure concurrently in all $m$ substream graphs, without violating the consistency requirements. Alternatively, a single substream graph, say $G_{1}$, can execute balancing, while the remaining substream graphs can copy the topology using the algorithm for Property 4.3. At the end of the balancing operation, the sizes of the left and right subtrees of any degree two peer do not differ by much as shown in the following.

Proposition 4.1. For any degree two peer $v$ in $G_{1}$, at the end of the balancing procedure, the sizes of $v$ 's left and right subtrees differ by at most $2 m$.

Proof. Let $T_{1}$ and $T_{2}$ respectively be the left and right subtree of $v$ in $G_{1}$. Without loss of generality, let us suppose $T_{1}$ has more peers than $T_{2}$. If $\left|T_{1}\right|-\left|T_{2}\right| \leq 2 m$ then we are done. Otherwise, consider traversing along the secondary edges of $T_{1}$. If there exists a degree two peer whose right subtree has a size of precisely $\mid\left(T_{1}\left|-\left|T_{2}\right|\right) / 2\right.$ again we are done. Otherwise, let us suppose $\delta$ is the size of the smallest subtree less than $\left(T_{1}\left|-\left|T_{2}\right|\right) / 2\right.$ along the secondary child path of $T_{1}$. Clearly $m-1 \leq \delta \leq 2 m-2$. This is because the very last degree two peer must necessarily have just a chain of degree one peers on its right subtree. Rotating around this degree two peer, we can reduce the difference $\left|T_{1}\right|-\left|T_{2}\right|$ by at least $m$ and at most $2 m-1$. Hence, whenever $\left|T_{1}\right|-\left|T_{2}\right|>2 m-1$, there always exists a rotation that reduces this difference. Repeating this process, we get that $\left|\left|T_{1}\right|-\left|T_{2}\right|\right| \leq 2 m$ upon balancing.

### 4.4.3 Steady-State Delay

Next, let us show that the delay of the network in the steady-state can be bounded as in Theorem 4.1.

Proof of Theorem 4.1. In the steady-state, all of the clusters in the network satisfy Properties 4.1-4.4. As such, the substream graphs are all symmetric and topologically similar. Further, by Proposition 4.1 the left and right subtrees of each degree two peer $v$ in $G_{1}$ have sizes that differ by at most $2 m$. If there are $n$ peers in the systems, let $f(n)$ denote the worst-case
steady-state delay possible. Then, we have the recursion

$$
\begin{equation*}
f(n) \leq 1+f((n-1) / 2+m) \quad \forall n \in \mathbb{N} \tag{4.1}
\end{equation*}
$$

by the above properties. This is because, for the node directly connected to the server in $G_{1}$, its left and right subtrees have a total of $n-1$ peers and differ in size by at most $2 m$. Continuing this recursion for $d=\log _{2}(n-1)$ times, we get

$$
\begin{align*}
f(n) & \leq d+f\left(\frac{n-1}{2^{d}}+\frac{m-1}{2}+\ldots+\frac{m-1}{2^{d-1}}+m\right) \\
& =\log _{2}(n-1)+f\left(1+\frac{m-1}{2}+\ldots+\frac{m-1}{2^{d-1}}+m\right) \\
& \leq \log _{2}(n-1)+1+\frac{m-1}{2}+\ldots+\frac{m-1}{2^{d-1}}+m  \tag{4.2}\\
& \leq \log _{2}(n-1)+2 m=\log _{2}(n-1)+\frac{2 R}{1-R},
\end{align*}
$$

where we have used $f(n) \leq n \forall n \in \mathbb{N}$ in inequality 4.2. For a general upload capacity of $C$ instead of 1 , by proportionately scaling the substream rates, we have the required delay bound.

In Section 4.5.1, we show that the above delay of the algorithm is order optimal. We now briefly discuss the scenario of a lowered redundancy in the network.

### 4.5 Reducing Redundancy

Tree-based algorithms, such as $[121,127]$, have a delay guarantee of $\left\lceil\log _{2} n\right\rceil$ for a streaming rate of $R=1$, while the algorithm we have presented has a weaker delay guarantee of order $O(\log n+1 /(1-R))$ (Theorem 4.1) for a rate $R \leq 1$ in steady-state. This can be explained by introducing a parameter called tolerance, $\tau$. In the streaming algorithm discussed, we have incorporated redundant capacity into the individual substream graphs using the edges in $U_{i}, i \in[m]$. Now, consider a scenario in which the the redundancy is reduced by a factor of $1-\tau$ for some $0 \leq \tau \leq 1$, i.e., let the edges in $U_{i}, \forall i \in[m]$, have a rate of $(1-\tau) / m$ instead of $1 / m$. The following theorem demonstrates the gain in the delay obtained for a lowered redundancy.

Theorem 4.3. For a tolerance parameter $\tau$, the steady-state delay guaranteed by the algorithm is bounded by

$$
\begin{equation*}
D(R, \tau, n) \leq \log _{2}(n+1)-\log _{2}\left(\frac{R(1-\tau)}{1-R}+1\right)+\frac{2 R(1-\tau)}{1-R}-2 \tag{4.3}
\end{equation*}
$$

for $0 \leq \tau \leq 1, n$ peers in the system and a rate of $R$.
Proof. In the steady-state of the original algorithm, the peers had a degree of one in all the substream graphs or they had a degree two in one of the graphs and degree one in all the rest. By a slight modification, we can make the algorithm more symmetric where every peer with degree two in some $T_{i}$ is necessarily a leaf node in some other tree in the steady-state. This leads to a more even distribution of capacity, i.e., any peer has degree one in $m-2$ trees and degree zero, two in one tree each or it has degree one in all the trees in the steady-state. This corresponds to a total upload capacity of $m r$ and $(m-2) r+r+r(1-\tau)$ respectively, where $r$ denotes the rate carried by each tree $T_{i}$. Therefore, we must have $m r+(1-\tau) r \leq 1 \Rightarrow r \leq \frac{1}{m+1-\tau}$. As such, in this scenario we can support a total rate of $R=m /(m+1-\tau)$ across the $m$ substream trees, which is higher than the rate $m /(m+1)$ of our algorithm. Since the topology is the same in both cases, by substituting for $m$ in the proof of Theorem 4.1 for delay, we get the desired bound in Equation (4.3).

Theorem 4.3 shows that for a rate of $R$, the steady-state delay obtained by lowering the amount of redundancy in the system is lower. The extreme case in which there is no redundancy at all in the system, i.e. $\tau=1$, corresponds to tree-based algorithms with a deterministic delay of $\left\lceil\log _{2} n\right\rceil$. Thus, we have obtained a relationship which shows the tradeoff between rate, delay and redundancy for the framework of our algorithm. For $\tau=0$, one implication of the way the substream graphs are structured is that connectivity of the nodes within the substream graphs directly translates to availability of download bandwidth from which peers can receive packets at a full rate of $R=m /(m+$ 1). However, if we reduce the redundancy in the graphs, i.e., for $\tau>0$, then we can only guarantee a rate of $(1-\tau) m /(m+1)=(1-\tau) R$ for the peers. This highlights the drawback with using a non-zero tolerance $\tau$; a large tolerance parameter can cause the transient drops in the rate received to be large. Hence, the lower rate and larger delay of our algorithm, compared to the
tree-based algorithms mentioned in the beginning of this section, has the advantage of guaranteed continuous playback at full rate even during peer churn.

### 4.5.1 Converse

The streaming algorithm we have presented involved binary trees in the substream graphs. In general, the streaming algorithm can work over any connected graph of $n$ vertices (mesh), where each vertex has an out-degree of at most $\Delta$. However, in Theorem 4.2 we show that the steady-state delay of our algorithm in Theorem 4.1 is order optimal within the general class of algorithms that use multiple arbitrarily structured graphs with redundancies for streaming.

In the steady-state, if communication happens via flow (copy + forward), and is deterministic, then one can always consider the flow to be an union of many constant rate sub-flows. Therefore, without loss of generality let us consider $T$ trees with the $i$ th tree carrying a rate of $r_{i}$. The full topology of the multicast streams can include more edges than just the trees above. The trees simply correspond to the routes by which the packets arrive earliest from the source to the peers. Now, suppose any one node departs the system; then at least one or more of the trees are broken. As such, reception of flow at full rate is hindered for some of the nodes and needs to be restored. Restoring is possible only by contacting another node in the tree corresponding to the substream, that is still connected to the server. Here, we are looking at a class of algorithms in which such a restoration is done by means of redundant links. Within this class of algorithms (that are solutions to the problem) we have the converse result stated in the theorem. The proof of Theorem 4.2 has been presented in Appendix C.

### 4.6 Conclusion

The p2p algorithm we have presented has the advantage of being simple, exact and lightweight. With the peer churn handling capability of unstructured algorithms combined with the deterministic delay guarantees of structured algorithms, the algorithm offers the best of both worlds. We have also iden-
tified a tolerance parameter, that is related to the transient rate guarantee, and have discussed its relationship to rate and delay. For the class of algorithms we discussed, we have shown that an additional delay of $R /(C-R)$ is the price paid for ensuring continuity. In general, other forms of adding redundancy exist - particularly coding techniques such as MDC or network coding. It would be interesting to study how these other methods interact with delay, rate and continuity. While the model captures many important issues, such as peer churn, testing in a real world environment is needed to identify the practical issues.

In Chapter 5 we discuss another important and emerging p2p application: cryptocurrencies. These networks also share many of constraints discussed in this chapter - such as requiring a low transmission delay, churn management, decentralized algorithms etc. - however, we will see that the core application requirements are very different. In particular we focus on achieving optimal anonymity in these networks. We find that the key, yet again, is in adopting the right network topology. These observations seem to suggest that the topological fluidity of p 2 p networks, usually seen as an inconvenience that needs to be handled, can in fact be harnessed for achieving multiple parallel network-level objectives in many instances.

## CHAPTER 5

## ANONYMITY IN NETWORKS

Cryptocurrencies are digital currencies that provide cryptographic verification of transactions. Bitcoin is the best-known example of a cryptocurrency [5]. In recent years, cryptocurrencies have transitioned from an academic research topic to a multi-billion dollar industry [150].

Cryptocurrencies exhibit two key properties: egalitarianism and transparency. In this context, egalitarianism means that no single party wields disproportionate power over the network's operation. This diffusion of power is achieved by asking other network nodes (e.g., other Bitcoin users) to validate transactions, instead of the traditional method of using a centralized authority for this purpose. Moreover, all transactions and communications are managed over a fully distributed, peer-to-peer (p2p) network.

Cryptocurrencies are transparent in the sense that all transactions are verified and recorded with cryptographic integrity guarantees; this prevents fraudulent activity like double-spending of money. Transparency is achieved through a combination of clever cryptographic protocols and the publication of transactions in a ledger known as a blockchain. This blockchain serves as a public record of every financial transaction in the network.

A property that Bitcoin does not provide is anonymity. Each user is identified in the network by a public, cryptographic key. If one were to link such a key to its owner's human identity, the owner's financial history could be partially learned from the public blockchain. Indeed, even if a user uses fresh public keys for each transaction, the blockchain can help adversaries cluster keys from the same user [151]. In practice, it is possible to link public keys to identities through a number of channels, including the networking protocols on which Bitcoin is built [19]. This is a massive privacy violation, and can be dangerous for deanonymized users.

Bitcoin is often associated with anonymity or privacy in the public eye, despite explicit statements to the contrary in the original Bitcoin paper [5].

People may therefore use Bitcoin without considering the potential privacy repercussions [152]. Moreover, this problem is not unique to Bitcoin; many spinoff cryptocurrencies (known as altcoins) use similar technologies, and therefore suffer from the same lack of anonymity in their p2p networks.

The objective of this chapter is to redesign the Bitcoin networking stack from first principles to prevent network-facilitated user deanonymization. We consider an adversary that aims to jointly deanonymize all users, rather than targeting one specific user. Critically, the network's reliability and performance must not be reduced. Although the networking stack is only one avenue for deanonymization attacks, it is an avenue that is powerful, poorly understood, and often ignored. To better convey the problem, we begin with a brief primer on Bitcoin and its networking stack. This chapter is based on our recent work [153].

## Bitcoin Primer

Bitcoin represents each user and each unit of Bitcoin currency by a publicprivate key pair. A user "possesses" a coin by knowing its private key. Any time a user Alice wishes to transfer her coin $m$ to Bob, she generates a signed transaction message, which states that Alice (denoted by her public key) transmitted $m$ (denoted by its public key) to Bob (denoted by his public key). This transaction message is broadcast to all active Bitcoin nodes, at which point miners, or nodes who choose to help validate transactions, race to append the transaction to a global ledger known as the blockchain. Specifically, each miner aggregates a group of transaction messages into a block, or list, and then completes a computational proof-of-work for the block; the first miner to complete a proof-of-work appends their block to the blockchain and reaps a reward of newly minted bitcoins and transaction fees.

Bitcoin message propagation. We focus on one key step in the pipeline: broadcasting transactions to other nodes. The broadcasting process is critical because it affects which nodes can reap a transaction's mining reward (by virtue of the delivery delays to different nodes), and it also affects the global consistency of the network (e.g., if only a subset of the users receive a given transaction).

To understand the mechanics of broadcasting, note that cryptocurrencies
can be abstracted into two layers: the application layer and the network layer. The application layer handles tasks like transaction management, blockchain processing, and mining. Nodes are identified by their public keys in the application layer. The network layer handles communication between nodes, which occurs over a p2p network of inter-node TCP connections. In the network layer, nodes are identified by their IP addresses. As we shall see momentarily, a node's IP address and public key should remain unlinkable for privacy reasons.

Bitcoin's peer-to-peer broadcast of transactions and blocks is based on flooding information along links in the p2p network. When a node learns of a new transaction or block, it passes the message to its neighbors who have not yet seen the message with an independent, exponential delay. The process continues recursively until all reachable peers receive the message. This broadcast protocol is commonly known as a diffusion process; it forms the basis of Bitcoin's global, eventually consistent log and is therefore of utmost importance to its correct and fair operation.

Desirable network properties. Bitcoin's network layer should exhibit two principal properties: low latency and anonymity.
Low latency means that the maximum time for a message to reach all network nodes should be bounded and small. Latency matters because if the network fails to deliver messages within a predictable time bound, the network risks reaching an inconsistent state.
Anonymity means that the adversary should be unable to link transaction messages (and hence, the associated public keys) to the IP address that originated a transaction. Recall that every transaction made by a public key is listed in plaintext in the blockchain. Therefore, if a public key can be linked to an IP address, the adversary can link all of that user's transactions. In some cases, the IP address could even be used to learn a node operator's human identity. Thus, deanonymization attacks can result in a user's entire banking history being revealed. Cryptocurrency users are typically recommended to choose fresh public keys and "mix" their coins with others to obscure their transaction history [154, 155] (in practice, few users do so $[156,151]$ ). However, these techniques are useless if the IP address of the source of the transaction can be recovered.

How the current network fails. In recent years, security researchers have
demonstrated multiple deanonymization attacks on the Bitcoin p2p network. These attacks typically use a "supernode" that connects to active Bitcoin nodes and listens to the transaction traffic relayed by honest nodes [157, 19, 158]. Because nodes diffuse transactions symmetrically over the network, researchers were able to link Bitcoin users' public keys to their IP addresses with an accuracy of up to $30 \%$ [19]. Moreover, the source estimators used in these papers are simple, and exploit only minimal knowledge of the p2p graph structure and the structured randomness of diffusion. We hypothesize that even higher accuracies may be possible with more sophisticated estimation tools.

These attacks demonstrate that Bitcoin's networking stack is inadequate for protecting users' anonymity. Moreover, the Bitcoin networking codebase is copied almost directly in other cryptocurrencies, so the problem pervades the ecosystem.

## Problem Statement and Contributions

We aim to address the Bitcoin p2p network's poor anonymity properties through a ground-up redesign of the networking stack. We seek a network management policy that exhibits two properties: (a) strong anonymity against an adversarial group of colluding nodes (which are a fraction $p$ of the total network size), and (b) low broadcasting latency. The anonymity guarantees we seek to provide are network-wide that uniformly protect all the users against a full-network deanonymization. We define these notions formally in Section 5.1. Critically, these networking protocols should be lightweight and provide statistical anonymity guarantees against computationally-unbounded adversaries. Lightweight statistical solutions are complementary to cryptographic solutions, which aim to provide worst-case anonymity guarantees, usually in the face of computationally bounded adversaries. Lightweight anonymization methods lower the barrier to adoption since a more efficient, faster protocol leads to a better user experience and also places less burden on developers to significantly modify existing code; their study is also of basic scientific and engineering interest.

Part of the novelty of our work is that the Bitcoin p2p networking stack has not been modeled in any detailed way (much less analyzed theoretically), to


Figure 5.1: Bounds on the precision and recall of any networking protocol, plotted for $p=0.2$. DANDELION has strong anonymity properties, achieving a precision-recall region close to the fundamental lower bounds.
the best of our knowledge. In addition to modeling this complex, real-world networking system, our contributions are threefold:
(1) Fundamental anonymity bounds. The act of user deanonymization can be thought of as classifying transactions to source nodes. Precision and recall are natural performance metrics. Recall is simply the probability of detection, a common anonymity metric that captures completeness of the estimator, whereas precision captures the exactness. We define these terms precisely in Section 5.1.4.

Given a networking protocol, the adversary has a region of feasible (recall, precision) operating points, which are achieved by varying the source classification algorithm. We give fundamental bounds on the best precision and recall achieved by the adversary for any networking protocol, as illustrated in Figure 5.1; here $p$ refers to the ratio of colluding nodes to the total number of nodes in the network. We show that a (recall, precision) point is feasible only if it lies between the red and blue lines in Figure 5.1. Moreover, every networking protocol yields an achievable (recall, precision) region to the adversary that intersects with the shaded region (a) in Figure 5.1 in at least one point.
(2) Optimal algorithm. We propose a simple networking protocol called DANDELION, whose achievable precision-recall region is nearly optimal, in the sense that it is contained in the achievable region of (nearly) every other possible networking protocol.

Dandelion consists of two phases. In the first phase, each transaction is propagated on a random line; that is, each relay passes the message to exactly one (random) node for a random number of hops. In the second phase, the message is broadcast as fast as possible using diffusion. Dandelion has two key constraints: (a) in the first phase, all transactions from all sources should propagate over the same line, and (b) the adversary should not be able to learn the structure of the line beyond the adversarial nodes' immediate neighbors.

The point labeled "Dandelion" in Figure 5.1 is the Pareto frontier of Dandelion's achievable precision-recall region (shaded in blue). The point labeled "Diffusion" was obtained by simulating a diffusion process on a snapshot of the Bitcoin server network from 2015 [21], and using a suboptimal source classifier. Because of this, the achievable region for diffusion must contain the red-shaded region, including the plotted point, but may be larger. Not only is the region for diffusion larger than the one for Dandelion, but Dandelion's region is nearly as small as possible. We revisit Figure 5.1 in greater detail in Sections 5.2 and 5.4.
(3) Practical considerations. We outline the practical challenges associated with implementing Dandelion. In particular, constructing the graph for Dandelion in a distributed fashion, and enforcing the assumption that the adversary cannot learn the graph, are non-trivial. We therefore propose simple heuristics for addressing these challenges.

We begin by discussing Bitcoin's p2p networking stack and our problem of interest, which we model in Section 5.1. We then present fundamental bounds on our anonymity metric in Section 5.2; these bounds are used for comparison with various networking policies later in the chapter. In Section 5.3, we present some first-order solutions, and explain why they do not work. We present our main result, Dandelion, in Section 5.4. Section 5.5 discusses the systems challenges of implementing Dandelion, and proposes some simple, heuristic solutions. We discuss the relation between Dandelionand prior related work in Section 5.1.6, and conclude with some open problems in Section 5.6.

### 5.1 Bitcoin Network and Adversarial Model

We model three critical aspects of Bitcoin's p2p network: the network topology, the message propagation protocol, and the deanonymizing adversary's capabilities. These models are based on existing protocols and observed behavior.

### 5.1.1 P2P Network Model

The Bitcoin p2p network contains two classes of nodes: servers and clients. Clients are nodes that do not accept incoming TCP connections (e.g., nodes behind NAT), whereas servers do accept incoming connections. We focus in this work on servers because (a) they are more permanent in the network, and (b) it is straightforward to generalize server-oriented anonymity solutions to also protect clients.

We model the p2p network as a graph $G(V, E)$, where $V$ is the set of all server nodes and $E$ is the set of edges, or connections, between them. For a node $v, \Gamma(v)$ denotes the set of $v$ 's neighbors in $G$. Similarly for a set of nodes $U, \Gamma(U)$ denotes the set of all neighborhood sets of the nodes in $U$. To model the graph's topology, we first discuss Bitcoin's network management protocols.

Each node in the Bitcoin p2p network has an address manager - a list of other nodes' contact information represented as a (IP address, port) pair, along with a time estimate of when that node was last active. When a server first joins the network, its address manager is empty, but the node can learn a random set of active addresses by contacting a hard-coded DNS server. During normal network operation, nodes periodically relay entries from their address managers, which helps spread information regarding active peers. We model address managers by assuming that each node possesses the contact information for every other Bitcoin server. In practice, address managers actually contain a random sample of population IP addresses.

Each server is allowed to establish up to eight outgoing connections to nodes in the server's address manager. An outgoing connection from Alice to Bob is one that is initiated by Alice. However, these TCP connections are bidirectional once established. We therefore model the subgraph of server-to-server connections as a random 16-regular graph. In practice, the degree
distribution is not quite uniform - we revisit this issue in Section 5.5.

### 5.1.2 Transaction Model

As explained in Section 5.1.3, the network is partitioned into honest nodes and colluding, adversarial nodes, who attempt to deanonymize users. In this work, we assume that all honest nodes generate one transaction in the time period of interest. In practice, servers generate transactions at different rates; however, transactions by a single node can be linked in practice [151]. Therefore, we treat multiple transactions from the same node as a single transaction to be deanonymized. We also assume the exact time when each server starts broadcasting its transaction is unknown to the adversary. A typical transaction can take up to 60 seconds to propagate through the Bitcoin network [159], so estimating its time of origin at a useful granularity of a second or sub-second can be difficult. $\mathcal{X}$ is the set of all transaction messages from honest servers. $X_{v}$ is the transaction message originating from honest server $v$ and $\mathbf{X}$ is a vector containing the ground truth mapping between source nodes $v$ and transactions $X_{v}$. We model the mapping between servers and transaction messages as being drawn uniformly from the set of all such mappings.

Spreading model. Once a Bitcoin transaction is complete, the source broadcasts the transaction message over the network. The protocol for broadcasting transactions should ensure low transaction latency, in order to provide network consistency and fairness.

Bitcoin currently uses a diffusion propagation mechanism to broadcast transactions, in which each transaction source or relay passes the transaction to the node's neighbors with independent, exponential delays. Once a node has received a particular transaction, the node does not accept future relays of the transaction. This diffusion spreading serves as a baseline for our algorithmic improvements. It has good latency properties due to its exponential spreading [160].

More generally, in this work, we consider spreading policies that are symmetric in the neighbor node IDs; that is, a forwarding node does not use the IP address values (or other metadata) of its neighbors to influence its forwarding decisions. This holds for diffusion spreading, but we constrain
our proposed solutions to also satisfy the same property.

### 5.1.3 Adversarial Model

We consider an adversary whose goal is to deanonymize all users by linking their transactions (and hence, their public keys) to their IP addresses. Note that an IP address does not necessarily determine a user's human identity. However, it can significantly narrow the set of candidates, particularly if combined with side information. Also, note that we are not interested in adversaries that aim to deanonymize specific nodes; this is a separate (important) problem that has been studied extensively in the literature [161, 162, 163]. Such solutions tend to require users to change their behavior, e.g., by adopting a new application. Our goal in this work is to provide network-wide (weaker) protection to all nodes, but without requiring human users to change their behavior.

In particular, we are interested in defending against botnets - large sets of malware-infected hosts that are controlled remotely, often without the host owners' knowledge [164]. Botnets are a commonly studied adversarial model for various Bitcoin attacks [165], largely because they are easy to access, cheap, and pervasive in the Bitcoin network [166]. While botnets can have many uses, we wish to defend against a botnet that aims to deanonymize users.

We model the botnet adversary as a set of adversarial, colluding "spy" nodes that participate in the Bitcoin network as if they were honest nodes (i.e., honest-but-curious). ${ }^{1}$ We denote honest nodes by $V_{H}$ and adversarial nodes by $V_{A}$. For a parameter $p$, we assume a fixed number of adversarial nodes $\left(\left|V_{A}\right|=n p\right)$ and honest nodes $\left(\tilde{n}=\left|V_{H}\right|=(1-p) n\right)$. The adversarial nodes are dispersed uniformly at random in the network; this reflects the botnet's ability to obtain IP addresses uniformly across the IP address space. However, for a given topology, the actual locations of the honest/adversarial nodes are random. We further assume that all nodes know the complete list of active IP addresses, and honest nodes cannot distinguish between an adversarial and honest IP address.

Whenever a transaction is broadcast over the network, the adversarial

[^16]

Figure 5.2: Red nodes are adversarial spies; blue nodes are honest. Message $X_{v}$ reaches the spy $w$ at time $t=2$.
nodes $\log$ the timestamps and the honest neighbors from which they receive the transaction. We assume a continuous-time system, in which simultaneous transmissions do not occur. For each honest server node $v$, we let $S_{v}$ denote the set of (transaction, receiving spy node, timestamp) tuples $\left(x, u, T_{u}(x)\right)$ such that transaction $x$ was forwarded by honest node $v$ to adversary $u \in$ $V_{A}$ at time $T_{u}(x)$ (Figure 5.2); $\mathbf{S}$ is the vector of all $S_{v}$ 's. We shall see in Section 5.4 that the honest server who first delivers a given transaction to the adversary plays a special role.

In addition to the transaction timestamps, the adversaries can also learn the network structure $G$, partially or completely, over time. The extent of such knowledge depends on the dynamism of the network, and will be made clear in the context of the specific networks being considered. For example, if the network is static over an extended period of time then adversaries can learn the entire graph $G$. On the other hand, in a fast changing network, the adversaries have knowledge of only their local neighborhood $\Gamma\left(V_{A}\right)$. For ease of exposition, let us, for now, use $\boldsymbol{\Gamma}$ to denote the adversary's knowledge of the graph.

Once the timestamps have been collected, the adversarial nodes collude to infer the transaction source. The adversary uses its observations $\mathbf{O}=$ $(\mathbf{S}, \boldsymbol{\Gamma})$ to output a mapping between transactions and honest servers; we let $\mathrm{M}\left(X_{v}\right) \in V_{H}$ denote the server associated with transaction $X_{v}$ in the adversary's mapping. This mapping is chosen to maximize the adversary's deanonymization payoff, defined in Section 5.1.4.

### 5.1.4 Anonymity Metric

A common metric for measuring a broadcasting scheme's anonymity is probability of detection. For a fixed transaction and estimator, probability of
detection is defined as

$$
\begin{equation*}
\mathbb{P}_{\mathrm{m}, G}(\text { detection })=\frac{\sum_{v \in V_{H}} \mathbb{P}\left(\mathrm{M}\left(X_{v}\right)=v\right)}{\tilde{n}} \tag{5.1}
\end{equation*}
$$

or the probability that the estimator outputs the correct source of a single transaction, computed over all transaction sources $v \in V_{H}$, mappings between sources and transactions $\mathbf{X}$, realizations of the message propagation trajectory, and graph realizations $G$ (if the graph is random). While probability of detection considers a single source, our problem considers the joint deanonymization of transactions from distinct sources. In this case, probability of detection inherently captures the recall, or completeness, of an estimator. We propose to augment this metric by also studying precision, which captures the exactness of an estimator.

Precision and recall are performance metrics commonly used in information retrieval for binary classification. Suppose we have $n$ data items, each associated with a class: 0 or 1 . We are given a classifier that labels each data item as either a 0 or a 1 , without access to the ground truth. We designate one of these classes (e.g. class 1) "positive". For a given classifier output on a single item, a true positive means the item was correctly assigned to class 1 , and a true negative means the item was correctly assigned to class 0 . A false positive means a 0 item was incorrectly classified as a 1 , and a false negative means a 1 item was incorrectly classified as a 0 . If we run this classifier on all $n$ data items, precision and recall are defined as follows:

$$
\begin{aligned}
\text { Precision } & =\frac{\mid \text { True Positives } \mid}{\mid \text { True Positives }|+| \text { False Positives } \mid} \\
\text { Recall } & =\frac{\mid \text { True Positives } \mid}{\mid \text { True Positives }|+| \text { False Negatives } \mid}
\end{aligned}
$$

where $|\cdot|$ denotes the cardinality of a set, and "True Positives" denotes the set of all data items whose classification output was a true positive (and so forth).

Precision can be interpreted as the probability that a randomly selected item with label 1 is correct, whereas recall can be interpreted as the probability that a randomly selected data item from class 1 is correctly classified. Adapting this terminology to our problem, we have a multiclass classification problem; each server is a class, and each transaction is to be classified. For
a given server $v$ and mapping M , the precision $D_{\mathrm{M}}(v)$ comparing class $v$ to all other classes is computed as ${ }^{2}$

$$
\begin{equation*}
D_{\mathrm{M}}(v)=\frac{\mathbb{1}\left\{\mathrm{M}\left(X_{v}\right)=v\right\}}{\sum_{w \in V_{H}} \mathbb{1}\left\{\mathrm{M}\left(X_{w}\right)=v\right\}}, \tag{5.2}
\end{equation*}
$$

and the recall is computed as

$$
\begin{equation*}
R_{\mathrm{M}}(v)=\mathbb{1}\left\{\mathrm{M}\left(X_{v}\right)=v\right\} \tag{5.3}
\end{equation*}
$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function. In multiclass classification settings, precision and recall are often aggregated through macro-averaging, which consists of averaging precision/recall across classes. This approach is typically used when the number of items in each class is equal [167], as in our problem. We therefore average the precision and recall over all servers and take expectation, giving an expected macro-averaged precision of $\mathbb{E}\left[D_{\mathrm{M}}\right]=\frac{1}{\tilde{n}} \sum_{v \in V_{H}} \mathbb{E}\left[D_{\mathrm{M}}(v)\right]$ and recall of $\mathbb{E}\left[R_{\mathrm{M}}\right]=\frac{1}{\tilde{n}} \sum_{v \in V_{H}} \mathbb{E}\left[R_{\mathrm{M}}(v)\right]$.

We now explain why probability of detection does not capture the distinction between precision and recall. Consider two estimators: in the first, the adversary's strategy is to assign all $\tilde{n}$ transactions to one randomly selected server $v$. In the second, the adversary creates a random matching between the $\tilde{n}$ transactions and honest servers. Both estimators have a probability of detection (i.e., expected per-node recall) of $1 / \tilde{n}$. However, the first estimator has an expected per-node precision of $1 / \tilde{n}^{2}$, while the second has an expected per-node precision of $1 / \tilde{n}$. Operationally, this can be interpreted as a difference in plausible deniability: the implicated node $v$ in the first case can deny being the source of any given transaction, because it could not have generated all $\tilde{n}$ transactions. If a node is correctly implicated in the second estimator, it has no plausible deniability. Probability of detection alone does not capture this difference, and is therefore insufficient as a standalone metric.

In this work, we quantify anonymity through a combination of expected macro-averaged precision (or "precision" for short) and expected macroaveraged recall (or "recall", or probability of detection). Higher precision and recall favor the adversary. For a mapping strategy M let $D_{\mathrm{M}}$ and $R_{\mathrm{M}}$ denote

[^17]the average precision and recall, respectively, obtained in a realization. Our metrics of interest, then, are the overall expected precision $\mathbf{D}_{\mathrm{M}}=\mathbb{E}\left[D_{\mathrm{M}}\right]$ and recall $\mathbf{R}_{\mathrm{M}}=\mathbb{E}\left[R_{\mathrm{M}}\right]$. This expectation is taken over four random variables: the graph realization $G$ (which can be random in general), the mapping between servers and messages $\mathbf{X}$, the observed timestamp and topological information $\mathbf{O}$, and the adversary's mapping strategy M. Similarly let $D_{\mathrm{M}}(v)$ and $\mathbf{D}_{\mathrm{M}}(v)$ denote the instantaneous and expected precisions at a server $v \in V_{H}$, and let $R_{\mathrm{M}}(v)$ and $\mathbf{R}_{\mathrm{M}}(v)$ denote the instantaneous and expected recalls. Let $\mathbf{D}_{\text {opt }}$ and $\mathbf{R}_{\text {OPT }}$ denote the precision and recall, respectively, of the precision-maximizing and recall-maximizing mapping strategies, respectively. The optimal precision is not necessarily achieved by the same mapping strategy as the optimal recall. The adversary is computationally unbounded.

### 5.1.5 Problem Statement

As network designers, we control two aspects of the network: the graph creation/maintenance strategy and the spreading protocol. Our goal is to choose a graph-selection strategy and a spreading protocol that simultaneously give low average latency, precision, and recall guarantees. We restrict ourselves to the following model of graph generation: For a fixed topology $\tau$, we assume that the nodes are equally likely to assume each possible label ordering in $\tau$. Moving forward, $G(V, E)$ will describe the resulting, labeled graph.

Let $\mathcal{T}$ denote the set of all graph topologies over $n$ nodes, and $\Sigma$ the set of graph-independent spreading strategies. The adversary controls only the estimation algorithm for mapping transactions to nodes. Given a topology $\tau \in \mathcal{T}$ and a spreading strategy $\sigma \in \Sigma$, let $\mathcal{M}_{\tau, \sigma}$ denote the set of mapping strategies that map $\tilde{n}$ transactions to $\tilde{n}$ servers, with all knowledge derived from the topology and the spreading strategy. If $\tau$ and $\sigma$ are clear from context, we simply use $\mathcal{M}$ to denote the space of mapping strategies. We define the detection region for $\tau$ and $\sigma$ as the set of achievable precision and recall operating points:

$$
\Omega(\tau, \sigma)=\left\{(D, R) \mid \exists \mathrm{M} \in \mathcal{M}_{\tau, \sigma}, D=\mathbf{D}_{\mathrm{M}}, R=\mathbf{R}_{\mathrm{M}}\right\} .
$$

Note that the detection region always contains the origin. The adversary's goal is to find estimators that achieve the boundary points of the region,
whereas our goal is to make the detection region as small as possible.
Problem. Characterize fundamental, protocol-independent bounds on the detection region. Further, identify a $\left(\tau^{*}, \sigma^{*}\right)$ pair whose detection region is a subset of the detection region of every graph-generation and spreading strategy:

$$
\begin{equation*}
\Omega\left(\tau^{*}, \sigma^{*}\right)=\bigcap_{\sigma \in \Sigma, \tau \in \mathcal{T}} \Omega(\tau, \sigma) . \tag{5.4}
\end{equation*}
$$

It is unclear a priori if such a strategy pair exists. We show a simple networking policy that closely approximates condition Equation (5.4).

### 5.1.6 Related Work

Related work includes anonymity attacks on Bitcoin, source detection analysis in diffusion processes, anonymous broadcasting, and privacy-conscious cryptocurrencies.

Anonymity attacks on Bitcoin. Most attacks on Bitcoin's anonymity harness the public blockchain [168, 169, 170]. Transaction patterns can be used to link user transactions over time, and in some cases identify the human owner of a public key. More recently, authors have demonstrated deanonymization attacks on Bitcoin's networking stack. These attacks typically use the first-spy estimator, and achieve surprisingly high accuracies [157, 19, 158]. The Bitcoin Core developers responded to these attacks with ad hoc changes to its networking stack for improved anonymity [171]. More recently, researchers have considered ISP-level adversaries [165], which are beyond the scope of this work.

Analysis of diffusion. A number of researchers have studied source detection on diffusion processes on graphs. These results show that for various classes of graphs and adversarial models, reliable deanonymization is possible $[172,173,174,175,176,177]$. However, there has been a relative lack of theoretical results in the analysis of diffusion under a spy-based adversary like ours. Many of the results in this space propose effective heuristics that achieve high recall in practice $[178,179,180]$. These papers suggest that by using centrality information, adversaries may be able to launch stronger attacks than prior practical network attacks [19, 158].

Anonymous broadcasting. The best-known work on anonymous broadcasting is dining cryptographer networks (DC nets), which enable a user to broadcast a message anonymously with information-theoretic guarantees [181]. DC nets are communication-intensive, which has prevented them from scaling beyond a few thousand nodes [182, 183, 184].

Another relevant topic is adaptive diffusion (AD) [185], recently proposed as an anonymous broadcasting protocol over fixed graphs. AD shares some properties with Dandelion, such as symmetry-breaking. However, AD can "get stuck" on real graphs, meaning that some messages do not reach the entire network [185]. This property is unacceptable in cryptocurrencies: all nodes should receive all messages for fairness and consistency purposes.

Finally, the core idea of Dandelion spreading - passing content through proxies - has been used in numerous anonymity systems, mainly for point-to-point communication $[186,187]$. However, existing systems have not connected Dandelion spreading to fundamental precision-recall guarantees, and they typically assume a complete graph topology [186]. In contrast, we identify topologies over which DANDELION spreading actually provides strong guarantees (i.e., not complete graphs). More fundamentally, our problem is focused on broadcasting over a network, which has different requirements and models than point-to-point messaging.

Privacy-conscious cryptocurrencies. Researchers have proposed several privacy-conscious alternatives to Bitcoin, including ZCash [162, 163, 188], Mimblewimble [189], CoinJoin [190], and TumbleBit [161]. These solutions mainly on cryptographic protocols that must be implemented either as a secondary service or as a separate cryptocurrency. Our work differs in that Dandelion provides weaker statistical guarantees, but it is inherent to the Bitcoin implementation without requiring users to change their behavior.

### 5.2 Anonymity Metric Properties

Precision and recall are not generally used as anonymity metrics, since most anonymity systems provide per-user anonymity guarantees [186, 191, 187, 185]. We instead want guarantees against a stronger adversary that jointly deanonymizes multiple users. The goal of this section is to give intuition about precision and recall as metrics, and to provide fundamental bounds on
both.
Our problem differs from traditional classification in that there is only one data item (transaction) per class (server). This restricts the set of achievable macro-averaged precision-recall points in a somewhat unconventional way. We first explain how precision and recall are typically used, and then prove fundamental bounds that illustrate the ways in which our problem differs from traditional classification problems.

Precision-recall curves. Most binary classifiers have an internal parameter (e.g., a threshold) that can be varied to give the classifier different precision and recall characteristics. Sweeping this parameter yields a tradeoff between precision and recall. While this tradeoff has been studied theoretically [192], it is most often illustrated empirically for a given classifier, through curves like Figure 5.3 (right). Notably, a classifier can achieve high recall $(\approx 1)$ at the expense of precision or vice versa. Hence the precision-recall points $(0,1)$ and $(1,0)$ are typically achievable in practice.

Unlike traditional precision-recall curves, we are not interested in the curve for a single estimator; we want to identify the achievable detection region across all estimators. Moreover, since ours is a multi-class classification problem, we consider macro-averaged precision and recall. With macro-averaging, increasing the recall (resp. precision) for one class will often reduce the recall (resp. precision) for another. Therefore, it is unclear what the precisionrecall tradeoff will look like, or even if the boundary points $(0,1)$ and $(1,0)$ are achievable. The following theorem restricts the set of feasible, macroaveraged precision-recall points for any estimator the adversary employs.

Theorem 5.1. Any mapping policy $\mathrm{M} \in \mathcal{M}_{\tau, \sigma}$ on a network with topology $\tau \in \mathcal{T}$ and spreading strategy $\sigma \in \Sigma$ has a precision and recall that are bounded as

$$
\begin{equation*}
\mathbf{D}_{\mathrm{M}} \stackrel{(a)}{\leq} \mathbf{R}_{\mathrm{M}} \stackrel{(b)}{\leq} \sqrt{\mathbf{D}_{\mathrm{M}}} \tag{5.5}
\end{equation*}
$$

(Proof in Section D.1.1)
This theorem follows from the definition of macro-averaged precision and recall; it implies that not only are corner points $(0,1)$ and $(1,0)$ unachievable, but every estimator's detection region must lie between the blue and red lines in Figure 5.3 (left). Given this constraint, a natural question is whether there


Figure 5.3: Bounds on the precision-recall detection region for any networking policy (left). Each bound is labeled with the corresponding equation number from Section 5.2. Example of a typical precision-recall curve (right).
exist precision and recall points that can always be achieved, regardless of the networking protocol. We demonstrate the existence of such points by analyzing a simple estimator.

Lower bounds. Computing lower bounds on precision and recall is challenging because the adversary's knowledge can vary depending on the networking policy. However, the so-called first-spy estimator (which is used in practical attacks like [19]) relies only on the adversary's knowledge of its local network neighborhood. The adversaries we consider will always have access to this information. The first-spy estimator outputs the first honest node to send a given message to any of the adversarial nodes. We start by showing that the first-spy estimator always achieves a precision and recall of at least $p^{2}$ and $p$, respectively, where $p$ is the fraction of spies. This in turn implies that the maximum precision and recall over all estimators are individually lower-bounded by $p^{2}$ and $p$, respectively.

Theorem 5.2. The optimal precision and recall on a network with a fraction $p$ of adversaries and any spreading policy are lower bounded as

$$
\begin{align*}
& \mathbf{D}_{0 \mathrm{PT}} \geq p^{2}  \tag{5.6}\\
& \mathbf{R}_{0 \mathrm{PT}} \geq p . \tag{5.7}
\end{align*}
$$

## (Proof in Section D.1.2)

This theorem implies that for any networking policy, the detection region must include at least one point in the shaded region of Figure 5.3 (left). Note that if an estimator can achieve a given (recall, precision) point, then it can
also achieve points with elementwise lower precision and recall by choosing to discard observed information. The purple curves labeled (i) and (ii) outline the boundaries of two examples of feasible detection regions, staggered for visibility.

Optimizing estimators. Given these constraints on the detection region, we want to understand what estimators achieve the maximum precision and recall, respectively. For a given network specification, precision and recall might be maximized by different estimators; if this is the case, then the detection region will have a non-trivial Pareto frontier, like curve (i) in Figure 5.3 (left). On the other hand, if the same estimator maximizes precision and recall, the detection region's Pareto frontier will be a single point, like curve (ii).

We start by proving that in order to maximize precision, the adversary should use a maximum-weight matching estimator, where the weights depend on the information observed by the adversary, such as graph structure and timestamps.

Theorem 5.3 (Precision-Optimal Estimator). The precision-optimizing estimator for an adversary with observations $\mathbf{O}=(\mathbf{S}, \boldsymbol{\Gamma})$, is achieved by a matching over the bipartite graph $\left(V_{H}, \mathcal{X}\right)$. Moreover, such a matching is a maximum-weight matching for edge weights $\mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right)$ on each edge $(v, x) \in V_{H} \times \mathcal{X}$ of the graph.
(Proof in Section D.1.3)
Theorem 5.3 gives a corollary used in Section 5.3 for bounding the performance of various networking protocols.

Corollary 5.1. The optimal expected payoff at a server $v$, under observations $\mathbf{O}=(\mathbf{S}, \boldsymbol{\Gamma})$ for the adversaries, is upper bounded as

$$
\begin{equation*}
\mathbb{E}\left[D_{\text {OPT }}(v) \mid \mathbf{O}\right] \leq \max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right) \tag{5.8}
\end{equation*}
$$

## (Proof in Section D.1.4)

Computing the probabilities in Corollary 5.1 may be challenging, depending on how much information the adversary has. Nonetheless, if the adversary can approximate these probabilities with some accuracy (e.g., if it knows the underlying graph $G$ ), there exist polynomial-time algorithms for computing max-weight matchings [193, 194].

The precision-optimal maximum-weight matching does not necessarily maximize recall. Notice that for any matching, its precision and recall are equal, due to the definitions of precision and recall. The following theorem characterizes a recall-optimal estimator, which assigns each message $x$ to any server $v$ for which $\mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right)$ is maximized.

Theorem 5.4 (Recall-Optimal Estimator). The recall-optimizing estimator for an adversary with observations $\mathbf{O}=(\mathbf{S}, \boldsymbol{\Gamma})$, is a mapping that assigns each transaction $x \in \mathcal{X}$ to any server $v^{*} \in \underset{v \in V_{H}}{\operatorname{argmax}} \mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right)$.

## (Proof in Section D.1.5)

The first-spy estimator is an instance of a recall-optimal estimator for spreading models in which the exit node to the first-spy is the most likely source. Moreover, Theorem 5.4 implies that a precision-optimal, maximumweight matching is only recall-optimal if it also maps each message to its most likely source, elementwise. For example, if $k$ servers are equally likely sources for $k$ messages, then the precision-optimal matching estimator is also recall-optimal.

Summary. This section provides fundamental limits on both precision and recall, as well as detailing estimators that optimize precision (Theorem 5.3) and recall (Theorem 5.4), respectively. These fundamental limits and estimators will be useful benchmarks as we analyze the precision-recall regions for networking policies in Sections 5.3 and 5.4.

### 5.3 Baseline Algorithms

With the fundamental bounds from Section 5.2, we now tackle our main problem: designing a networking policy with a minimal detection region. A key message of our work is that statistical anonymity requires mixing of messages: users should spread their own messages and those of their peers in a way that is difficult to distinguish. Degree of mixing depends on three key properties of a networking policy: (1) the spreading protocol, (2) the topology of the network, and (3) the dynamicity of the network (i.e., how often the p2p graph changes). For example, the current Bitcoin network uses diffusion spreading over a static, roughly 16-regular topology. This policy has
poor mixing - i.e., a large detection region - because different nodes have unique spreading patterns and can therefore be deanonymized.

In this section, we first identify a taxonomy of networking policies, based on the properties above. We then systematically evaluate the anonymity of various first-order, natural networking policies from this taxonomy. We show that most of these baseline policies have poor anonymity guarantees, and we extract rules of thumb for improving a policy's anonymity. These rules of thumb will build the groundwork for our main result, DANDELION, presented in Section 5.4.

### 5.3.1 Taxonomy of Networking Policies

Our taxonomy has three axes: spreading protocol, topology, and dynamicity.
Spreading protocol. The space of spreading protocols is vast. In this chapter, we consider a few natural, first-order spreading policies, and also propose a new protocol called dandelion spreading. ${ }^{3}$ Perhaps the most natural spreading strategy is flooding, where messages are propagated with a fixed delay to all neighbors. A slightly refined version is diffusion, which adds independent randomness to the transmission delays of flooding. Diffusion is explained in Section 5.1.2. Flooding and diffusion reflect the current status quo in the Bitcoin network.

Given that our goal is to provide anonymity, another natural strategy is to forward a message to a randomly chosen node, which then runs diffusion or flooding. We call this spreading protocol diffusion-by-proxy.

Finally, we propose in this chapter a new protocol called dandelion spreading. Dandelion spreading forwards each message on a randomly selected line before diffusing it to the rest of the network. Since dandelion spreading is a comparatively new protocol (not a first-order baseline), we defer a detailed discussion to Section 5.4.

Topology. We are interested in topologies that are simultaneously simple to construct, analyzable, and good for anonymity. We therefore limit ourselves to a set of canonical graph models: lines, trees, $d$-regular graphs, and complete graphs. These categories are not mutually exclusive; lines are a

[^18]special case of both trees and regular graphs (we consider lines and cycles interchangeably), and complete graphs are a special case of regular graphs.

Dynamicity. Many network-based deanonymization attacks use partial or full knowledge of the connectivity graph between nodes [172]. We assume that the network can change the graph at varying rates to control the adversary's ability to learn it. We consider two extremes on this spectrum: static graphs and dynamic graphs. In static graphs, the network never changes the graph, so the adversary learns it fully over time. We define dynamic graphs as graphs that are changed at a rate such that the adversary only knows its local neighborhood at any given point in time.

In the remainder of this section, we first explore the regions of our taxonomy by studying three baseline networking policies: flooding, diffusion, and diffusion-by-proxy. Although none of these baselines has satisfactory anonymity guarantees, the associated analysis provides valuable intuition that helps us design better policies in Section 5.4.

### 5.3.2 Flooding

To model flooding, we assume that messages propagate along each graph edge with a deterministic delay, and nodes forward incoming messages to their neighbors with a constant delay. On undirected topologies, flooding has poor source-hiding due to symmetry and the deterministic spreading scheme. However, it is unclear if directed topologies fare better. We begin by showing that flooding has poor performance on directed, static, $d$-regular graphs.

Proposition 5.1. The expected precision of flooding on a static d-regular graph is at least $\mathbf{D}_{\text {OPT }} \geq\left(1-(1-p)^{d}\right) \geq p$.
(Proof in Section D.2.1)
Flooding performs poorly on static regular graphs because each honest node has a unique spreading "timestamp signature", and the adversary can predict these signatures. That is, if node $v$ is the source, then the adversarial nodes receive all messages from $v$ in a deterministic timing pattern. Moreover, the adversary can predict this pattern from the structure of the graph, due to the fixed nature of flooding.

This reasoning suggests that if the adversary does not know the graph, it cannot predict nodes' spreading patterns, and therefore cannot deanonymize nodes. However, the following proposition shows that even when the graph is dynamic, the adversary can achieve a high precision.

Proposition 5.2. The expected precision of flooding on a dynamic (defined in Section 5.3.1) d-regular graph is bounded as $\mathbf{D}_{\mathrm{OPT}} \geq c p$ for some constant $c>0$ independent of $p$.
(Proof sketch in Section D.2.2)
This result highlights that even if the adversary cannot predict the exact timestamp pattern for a given node, it can infer certain statistical properties of the pattern that are sufficient for deanonymization. In short, as long as the topology allows messages to flood in more than one direction, the adversary can use the statistics of observed timestamp signatures to infer the source of a message.

Lesson. Do not flood content in multiple directions on the graph at the same rate.

### 5.3.3 Diffusion

Diffusion is a natural successor to flooding; instead of using deterministic delays, it uses random ones. By introducing uncertainty into the adversary's timing estimates, diffusion reduces the adversary's overall precision and recall. However, much research in recent years has shown that the source of a diffusion process can nonetheless be identified reliably [172, 173, 174, 175, $176,177,178,179]$. Although there are no theoretical results on the precision or recall under our particular adversarial model, several heuristic estimation algorithms are able to identify the source of a diffusion process on many classes of graphs [178, 179]. Moreover, theoretical results exist on other adversarial models $[172,173,175]$. All of these results rely on the intuition that diffusion spreads content symmetrically. Because of this, the source node appears at the center of the adversary's observed spreading pattern, and can be identified. Diffusion is therefore not a satisfactory solution to this problem.

Lesson. Random forwarding delays are not powerful enough to provide anonymity against spreading protocols that spread content symmetrically.

### 5.3.4 Diffusion-by-Proxy

The takeaway message from diffusion and flooding is that symmetry of spreading leads to deanonymization. To counter this, we must break the symmetry of diffusion. A natural strategy for breaking symmetry about the source is to ask someone else to spread the message. That is, for every transaction, the source node chooses a peer uniformly at random from the pool of all nodes. It transmits the message to that node, who then broadcasts the message. More generally, the network could forward each message a few hops (each hop choosing a new node at random) before diffusing it. We call this approach diffusion-by-proxy, and it is conceptually equivalent to propagating over a line that changes for every transmission. Diffusion-by-proxy might seem like it should have low precision because the graph is so dynamic, but that intuition turns out to be false.

Proposition 5.3. The expected first-spy precision of diffusion-by-proxy is bounded as $\mathbf{D}_{\mathrm{FS}} \geq \frac{p}{1-p}\left(1-e^{p-1}\right)$.
(Proof in Section D.2.3)
Intuitively, this statement holds because each node delivers its own message to the adversary with probability $p$, and few other nodes report to the adversary over the same edge. So even though diffusion-by-proxy breaks the symmetry of diffusion, it also provides many paths for messages to reach the adversary. Since there are many total paths to the adversary, each path sees (relatively) less traffic, which in turn reduces the amount of mixing that happens. A simple countermeasure is to reduce the number of paths over which messages can flow.

Lesson. There is anonymity in numbers; dense graphs achieve poor mixing because they do not constrain messages to flow over the same paths.

### 5.4 Dandelion

The baseline spreading protocols from Section 5.3 provide us with a key guideline for building more anonymous networking policies: spread asymmetrically over a sparse graph. In this vein, we propose a new protocol: dandelion spreading. While the basic intuition of dandelion spreading is

```
Algorithm 8: Dandelion Spreading. \(\mathcal{N}_{\text {out }}(G, v)\) denotes the out-
neighbors of node \(v\) on directed graph \(G\).
    Input: Message \(X_{v}\), source \(v\), anonymity graph \(G\), spreading graph
            \(H\), parameter \(q \in(0,1)\)
    anonPhase \(\leftarrow\) True
    head \(\leftarrow v\)
    recipients \(\leftarrow\{v\}\)
    while anonPhase do
        /* forward message to random node */
        target \(\sim \operatorname{Unif}\left(\mathcal{N}_{\text {out }}(G\right.\), head \(\left.)\right)\)
        recipients \(\leftarrow\) recipients \(\cup\left\{X_{v}\right\}\) from head to target
        head \(\leftarrow\) target
        \(u \sim \operatorname{Unif}([0,1])\)
        if \(u \leq q\) then
            anonPhase \(\leftarrow\) False
        end
    end
    /* Run diffusion over \(H\) from 'head' */
    \(\operatorname{Diffusion}\left(X_{v}\right.\), head, \(\left.H\right)\)
```

used in several point-to-point anonymous communication systems [186, 187], it has not been formally studied in the context of anonymous broadcast messaging.

Dandelion spreading consists of an anonymity phase and a spreading phase (Algorithm 8). In the anonymity phase, the protocol spreads the message over a randomly-selected line for a random number of hops; in the spreading phase, the message is broadcast using diffusion until the whole network receives the message. In general, the two phases can occur over different graphs. In this work, we will design a (possibly time-varying) graph $G$ over which the anonymity phase occurs, and we will assume the spreading phase occurs over the current Bitcoin p2p network $H$. The name "dandelion spreading" reflects the spreading pattern's resemblance to a dandelion seed head (Figure 5.4).

The two-phase nature of dandelion spreading allows us to separately design networking policies that optimize anonymity and latency. This separated architecture is not necessarily optimal in terms of a latency-anonymity tradeoff; exploring that tradeoff is an interesting direction for future work. However, diffusion is known to have good spreading properties [160], but poor anonymity properties [172]. Therefore, we combine it with an anonymity


Figure 5.4: Dandelion spreading forwards a message in a line over the graph, then broadcasts it using diffusion. Here both phases occur over the same graph, i.e., $H=G$.
phase of constant duration (in an order sense), such that the average latency is increased by a small, bounded factor. We subsequently assume that the spreading phase can be fully deanonymized; i.e., the node that launches the diffusion process can be identified. As such, we only need to analyze the precision and recall of the anonymity phase. This assumption does not weaken our anonymity guarantees since it gives the adversary more power.

A key observation for this analysis is that the anonymity phase of dandelion spreading largely removes the need for exact timestamps. For honest server $v$, let $S_{v}^{\prime} \subseteq S_{v}$ denote a trimmed down version of $S_{v}$, in which we retain only those transaction $\log$ tuples $\left(x, u, T_{u}(x)\right)$ that correspond to the first time transaction $x$ was received by an adversary from any honest node. That is, we only keep a tuple if $u$ was the first spy to see message $x$, and $x$ was delivered to $u$ by honest exit node $v$. As before, let $\mathbf{S}^{\prime}$ denote the vector of all $S_{v}^{\prime}$ 's. Then, with dandelion spreading, it holds that $\mathbf{X}-\left(\mathbf{S}^{\prime}, \boldsymbol{\Gamma}\right)-\mathbf{S}$ forms a Markov chain. Therefore it is sufficient to use only the first observation information $\mathbf{S}^{\prime}$ instead of $\mathbf{S}$ for computing transaction likelihoods. In fact, the sufficient statistic $\mathbf{S}^{\prime}$ can be further simplified by ignoring the timestamp coordinate $T_{u}(x)$ in the tuples. This is possible due to our assumption that the transactions' originating times are unknown a priori to the adversary, which removes the observed timestamps from any temporal reference frame. Hence, in the remainder of this chapter, with a slight abuse of notation, we use $S_{v}$, for honest server $v$, to denote the set of message tuples $(x, u)$ such that (i) $u$ was the first adversarial node to receive $x$ and (ii) $u$ received $x$ from $v$. S denotes the vector of $S_{v}$ 's.

Note that a similar argument does not hold for spreading mechanisms
like flooding or diffusion, in which multiple independent timestamps across different nodes (i.e., not just the first observation of a message) are used to compute likelihoods. The diversity of such observations allows the estimator to compare timestamps across nodes, thus making them useful for detection.

We begin by showing that the maximum recall for dandelion spreading over any connected topology is $p$, the lower bound from Theorem 5.2.

Theorem 5.5. The expected maximum recall for dandelion spreading on any connected graph of $n$ nodes with a fraction $p$ of adversaries is $\mathbf{R}_{0 \mathrm{PT}}=p+O\left(\frac{1}{n}\right)$.
(Proof in Section D.3.1)
The reason for this result is that dandelion spreading propagates content unidirectionally over a line. This lack of symmetry makes the first-spy estimator - which has a recall of $p$ - optimal. Theorem 5.5 result implies that as we explore various topologies of dandelion spreading, we only need to analyze and minimize their precision. We do so for three topologies of the graph $G$ : static trees, dynamic trees, and dynamic lines. Each topology provides intuition about how to achieve anonymity. We find that dynamic lines achieve nearly-optimal average precision and recall.

### 5.4.1 Static Trees

Recall that our goal is to mix messages from different users; in this sense, trees are a natural topology to study. That is, consider a rooted, directed $d$-regular tree, with each edge directed toward the parent node. Dandelion spreading respects the directedness of the graph, so during the anonymity phase, each node passes all messages to its parent node (i.e. toward the root). Nodes near the root are therefore able to mix their own messages with exponentially many other messages from users beneath them in the tree. However nodes near the leaves of the tree have few nodes beneath them, and therefore experience minimal mixing. This fundamental asymmetry results in a high average precision.

Proposition 5.4. The expected precision under a matching estimator MAT on any tree is given by $\mathbf{D}_{\mathrm{MAT}} \geq p$.
(Proof in Section D.3.2)

Intuitively, when the graph is known, the adversary can partition nodes into wards, or sets of honest nodes that share the same first spy. Each ward contributes equally to the adversary's precision, so we would like to minimize the number of wards. On trees, the expected number of wards is $p \tilde{n}$, most of which consist of a single leaf with an adversarial parent node. This gives an overall precision of $p$.

Although a precision of $p$ is an improvement over Bitcoin's current networking policy, we would like to achieve a precision close to the lower bound of $p^{2}$ (Theorem 5.2). We therefore consider topologies with fewer wards on average.

Lesson. Use topologies in which it is difficult for the adversary to partition nodes into wards.

### 5.4.2 Dynamic Trees

The adversary was able to partition the nodes of a static tree into wards largely because the graph was known. A natural question is whether dynamic trees have the same problem, since most of the graph is hidden, except the adversary's local neighborhood.

A perfect $d$-ary tree is a rooted tree in which each node has either $d$ children or no children, and all leaves are at the same depth. Again, we assume each edge in such a tree is directed toward the parent node. We find that dandelion spreading on perfect $d$-ary trees has an expected precision similar to that of static trees.

Proposition 5.5. The expected precision of the first-spy estimator on a perfect d-ary tree, $d \geq 2$, can be bounded as $\mathbf{D}_{\mathrm{FS}} \geq p / 2$.
(Proof in Section D.3.3)
Since the graph is now dynamic, the adversary cannot explicitly determine every ward like it could in the static case. However, the first-spy estimator naturally identifies wards that consist of a single honest leaf. Statistically, there are many such wards on trees that are not lines, so we obtain similar guarantees to the static case. This implies that the problem with trees is mainly the fact that they have many leaves.

Lesson. A dynamic graph does not mitigate the negative impact of leaf nodes.

### 5.4.3 Dynamic Lines: Dandelion

Next, we study dynamic line graphs. Lines are 2-regular trees, but unlike higher degree trees, they do not suffer from the asymmetry problems associated with leaves. However, line graphs seem to lack the strong mixing properties of higher-degree graphs. Nonetheless, we show near-optimal precision for this class of graphs. This happens because despite the moderate mixing on lines, the number of honest nodes visible to the adversary is also small. As such, the adversary cannot accurately partition nodes into wards, which reduces the overall precision. Note that this would not hold in the static case, since the adversary could identify the wards exactly.


Figure 5.5: The Dandelion networking policy: (1) dandelion spreading, (2) a line topology, (3) a dynamic graph.

We use the name Dandelion to refer to a full networking policy (Figure 5.5): dandelion spreading over dynamic lines (i.e., 2-regular graphs with out-degree 1). We begin by showing that Dandelion has near-optimal precision.

Theorem 5.6. The expected precision of Dandelion (i.e., dandelion spreading on a dynamic line graph) with $n$ nodes and a fraction $p<1 / 3$ of adversaries, is upper bounded by

$$
\begin{equation*}
\mathbf{D}_{\mathrm{opT}} \leq \frac{2 p^{2}}{1-p} \log \left(\frac{2}{p}\right)+O\left(\frac{1}{n}\right) \tag{5.9}
\end{equation*}
$$

(Proof in Section D.3.4)
This result states that for small $p$, the expected maximum precision is within a logarithmic factor of our lower bound of $p^{2}$. The stated bound has


Figure 5.6: Detection regions for studied networking policies, $p=0.2$.
Dandelion has a detection region close to the fundamental lower bounds.
loose constants for improved readability; a tighter expression is included in the proof. The proof depends heavily on the fact that the adversary cannot reliably assign nodes toward outside of its local neighborhood on the graph. As such, it is forced to use estimators that would give suboptimal precision in the static case, like variants of the first-spy estimator.

Figure 5.6 illustrates Dandelion's detection region compared to those of other benchmark policies. The points for diffusion and flooding are generated through simulation over a snapshot of the Bitcoin server graph from 2015 [21]. Since dandelion spreading has optimally-low recall (Theorem 5.5), the Pareto frontier for Dandelion is exactly the plotted point (i.e., points below and to the left are achievable). The other policies are analyzed using possiblysuboptimal estimators, so their detection regions must at least contain the plotted points. Dandelion therefore satisfies the theoretical demands of our problem, and performs favorably compared to baseline alternatives.

Remarks. (1) While Dandelion is near-optimal, whether the logarithmic gap between our algorithm (Theorem 5.6) and the lower bound (Theorem 5.2) can be improved remains an important open question. (2) We have assumed that the exact start-times of transactions are unknown to the adversary. Analyzing Dandelion's performance under a partial knowledge of starttimes is also an important future direction.

```
Algorithm 9: \(k\) Approximate Line Approximates a directed line
graph in a fully-distributed fashion. Each node picks an edge from \(k\)
options
    Input: Set \(V=\left\{v_{1}, v_{2}, \ldots, v_{n}\right\}\) of nodes; parameter \(k\)
    Output: A connected, directed graph \(G(V, E)\) with average degree 2
    for \(v \leftarrow V\) do
        /* pick \(k\) random targets */
        \(u_{i} \sim \operatorname{Unif}(V \backslash\{v\})\), for \(i \in\{1, \ldots, k\}\)
        /* pick the smallest in-degree */
        \(u \leftarrow \operatorname{argmin} \operatorname{deg}_{i n}\left(u_{i}\right)\)
        \(E=E \cup(v \rightarrow u) / *\) make connection \({ }^{u_{i}}(v /\)
    end
    return \(G(V, E)\)
```


### 5.5 Systems Issues

Theoretically, Dandelion is simple and exhibits desirable anonymity properties. However, the implementation raises a number of practical considerations, like how to construct the underlying line graph and how to provide sufficient graph dynamicity. We discuss each of these challenges, and introduce practical, heuristic solutions for addressing them.

### 5.5.1 Constructing a Line Graph

In Dandelion, all nodes propagate their messages over the same line. To implement this, the network must build either a Hamiltonian circuit or a set of long, disjoint lines in a fully distributed fashion. Constructing a Hamiltonian circuit is challenging in our case because it is not a one-time event; in order to provide dynamicity, the network must frequently construct a new random line. To ensure scalability, the algorithm for constructing such a line should be fully-distributed, lightweight, and asynchronous.

Traditional algorithms for computing Hamiltonian circuits are often computationally intensive and/or require centralized control [195, 196], but recent papers have studied lightweight, distributed alternatives [128, 197]. For instance, [197] first generates line fragments, then splices them together into a circuit. However, it relies on the nodes of each line fragment knowing the identities of the fragment's head and tail nodes. This could partially


Figure 5.7: Degree distribution of $k$-approximate lines (Algorithm 9) for various $k$. The fraction of leaves decreases as the number of edge choices $k$ increases.
reveal the graph structure to the adversary, which would likely change our anonymity guarantees.

On the other hand, [128] builds up the circuit sequentially; a pair of nodes start as the circuit "seeds". Each node $v$ who joins the circuit contacts a random node $u$ from the partially-built circuit; $u$ replies with the IP address of its outgoing neighbor $w$. Then $v$ splices itself into the $(u, w)$ edge, so the new ordering is $u \rightarrow v \rightarrow w$. This distributed protocol is a viable solution for constructing an exact line.

Another alternative is to use Bitcoin's current networking strategy to approximate a line. Currently, each Bitcoin server generates eight connections at random. We can approximate a line by asking each server to create one outgoing connection at random. This protocol can be refined by having each server, prior to making a connection, contact $k$ nodes and connect to the node with the smallest in-degree. This protocol (specified in Algorithm 9) is fully distributed, but it is unclear how well it approximates a line.

Figure 5.7 illustrates the degree distribution of Algorithm 9's approximation of a line graph with 1,000 nodes, averaged over 1,000 trials, for different values of $k$. First, note that the average degree is two by construction. As $k$ increases, the fraction of leaves decays, with the greatest reduction coming as we transition from $k=1$ to $k=2$. This empirical observation is supported by the following proposition:

Proposition 5.6. Suppose Algorithm 9 is used to construct a $k$-approximate line over n nodes. Let the empirical degree distribution of the resulting graph's
nodes have support $\left(d_{1}, \ldots, d_{m}\right)$, where $d_{1}<\ldots<d_{m}$. Then with probability $1-o(1)$, the maximum degree $d_{m}$ satisfies the following condition:

$$
d_{m}= \begin{cases}\frac{\log n}{\log \log n}(1+o(1))+\Theta(1) & \text { if } k=1 \\ \frac{\log \log n}{\log k}(1+o(1))+\Theta(1) & \text { if } k>1 .\end{cases}
$$

(Proof in Section D.4.1)
Here we are using maximum degree as a proxy for regularity (or number of leaves), but recall that the expected degree is fixed by construction. Therefore, if we can drive the maximum degree down to 2 , the minimum degree must also be 2. Proposition 5.6 suggests that we can reap most of the precision gains of a more regular graph by connecting to one of $k=2$ nodes with minimum in-degree, whereas larger $k$ only improves the regularity by a factor logarithmic in $k$.

Section 5.4.2 showed that leaves increase the precision of a scheme because the leaf nodes' messages cannot be mixed with other messages. This suggests that Dandelion can achieve lower precision over $k$-approximate lines (Algorithm 9) by increasing $k$ and decreasing the number of leaves. Figure 5.8 compares the the first-spy estimator precision for exact lines (generated by [128]) and $k$-approximate lines (Algorithm 9). The figure shows that over $k$-approximate lines, average precision decreases as $k$ increases (i.e., as the distribution becomes more regular), but the returns are diminishing in $k$. The most significant decrease in precision occurs as we transition from $k=1$ to $k=2$; higher values of $k$ give marginal improvements. Moreover, the precision of $k$-approximate lines is significantly larger than that of exact lines, which could be obtained through the line-creation protocol in [128].

Algorithm 9 and [128] are both viable options for constructing a line. Although [128] has lower overall precision, it uses more fine-grained information - connection IPs rather than simple degree information. As such, [128] may be less robust to misbehaving nodes. Understanding this tradeoff, and developing alternatives that are resistant to adversarial misbehavior, are of practical interest.


Figure 5.8: The $k$ edge choices during graph creation (Algorithm 9) do not significantly reduce the precision of the first-spy estimator beyond $k=1$.

### 5.5.2 Preventing Graph Leakage

Another challenge is that Dandelion assumes the graph $G$ is unknown to the adversary. However, lines can be learned over time. First, note that for any given adversarial node $s_{1}$ on a 2-regular digraph, $s_{1}$ can learn the honest nodes immediately before and after it on the graph by sending probe messages. Now consider the following scenario: a message from an honest user propagates on the line, and passes $s_{1}$. At an honest node $v$ between $s_{1}$ and the next adversarial node $s_{2}$ (see Figure 5.9), the message transitions into the spreading phase at and starts diffusing over the main p2p graph $H$. We assume that the adversary can reliably infer the diffusion source $v$. Since $s_{2}$ did not receive the message before the spreading phase began, and $v$ was the source of the spreading phase, the adversary learns that $v$ lies between $s_{1}$ and $s_{2}$. Thus, the adversary learns $G$ at a rate proportional to the creation of new transactions, which raises the adversary's expected per-node precision to $p$.


Figure 5.9: The adversary can easily learn line graphs.

This problem must be managed by changing the graph quickly enough that the adversary cannot learn it - on the timescale of transaction execution. As an estimate, the Bitcoin network currently sees about three transactions per second [198]. For $\$ 200$, one can rent a botnet of 1,000 US-based zombies,
or corrupted hosts for an hour [199]. Since the current Bitcoin network consists of about 5,500 servers [200], this corresponds to $p \approx 0.15$; each ward would have about seven nodes on average, of which five are unknown to the adversary in the fully dynamic setting. We conservatively assume that each transaction launches its spreading phase from a different honest node. If we want to ensure that the adversary never learns more than $40 \%$ of interior nodes, we should change the graph every 5500 transactions $\times \frac{5}{7} \times$ $0.4 \times \frac{1 \mathrm{sec}}{3 \text { transactions }} \approx 9$ minutes. This is easy to enforce in a distributed fashion; every nine minutes, each node will tear down its connections and form new ones. Synchrony between nodes is not needed for this restructuring due to the fully distributed line approximation protocol.

More powerful attackers can create botnets of tens of thousands of nodes, which would overwhelm Dandelion. In such scenarios, statistical solutions are no longer appropriate.

### 5.6 Conclusion

In this chapter, we have proposed a redesign for the Bitcoin p2p stack to provide anonymity against botnet-like adversaries who wish to link users to transactions. We have presented the Dandelion networking policy, which achieves nearly optimal anonymity guarantees with a simple, distributed implementation. Dandelion achieves these guarantees by mixing messages from different users on a graph that is unknown to the adversary. This mixing makes it theoretically difficult for the adversary to jointly deanonymize users. We have also presented a framework for analyzing other networking policies in terms of precision and recall.

## CHAPTER 6

## CONCLUSION

In this dissertation we have studied some of the key algorithmic challenges, and methods to solve them, in canonical large-scale networking systems underpinning today's internet. Motivated by communication intensive distributed computing frameworks in data centers, we have considered the twoparty protocol simulation problem in Chapter 2, and presented both an improved lower bound for communication complexity and a round-by-round compression protocol. In Chapter 3 we have proposed a fast algorithm for scheduling delay constrained circuit switches in data centers. The algorithm is directly motivated by the underlying submodularity in the problem. We have also discussed indirect routing as a promising (and sometimes essential) alternative while routing packets. Chapters 4 and 5 deals with p2p networks requiring decentralized algorithms. In Chapter 4 we have presented a deterministic algorithm and topology for live streaming in a p2p network. The key observation here is that QoS deterioration arising from unpredictable peer arrivals and departure can be buffered by introducing redundancy into the network. We have also presented fundamental lower bounds on the best delay possible for a fixed amount of redundancy. Finally in Chapter 5 we have discussed the problem of network deanonymization in the Bitcoin network, and presented fundamental bounds under a common adversarial model. By essentially forwarding transaction messages over a line graph, we have showed that near-optimal anonymity can be achieved.

### 6.1 Open Problems and Future Directions

We discuss chapterwise open problems and directions for future research below.

Chapter 2. A key step in our lower-bound approach is identifying the
amount of common randomness generated through protocol simulation. However our estimate for the amount of common randomness does not rely on the structure of the function to be computed. This is in contrast to most of the existing lower bounds on communication complexity for function computation, such as the partition bound or the discrepancy bound, where the structure of the computed function plays an important role. In particular, a comparison of our approach with other existing approaches for specific functions is not available. An important future research agenda is to incorporate the structure of functions in our bound; the case of functions with a small range such as Boolean functions is of particular interest.

Chapter 3. In the case of indirect routing, we have provided a fast algorithm for path selection, however it holds only if the switch configurations are calculated separately. The problem of jointly deciding the switch configurations and indirect routing policies remains open. While submodular function optimization with nonlinear constraints is in general intractable, the specific constraints discussed in Section 3.3.1 perhaps have enough structure that they can be handled in a principled way. The key challenge here seems to be the computation of an appropriate sequence of matchings. Ideally, we would like a matching sequence that (i) offers good connectivity between nodes and (ii) reduces the number of hops packets have to travel to reach their destination. One can propose two approaches toward achieving this goal. The first idea is to use random bipartite matchings for the schedule and perform multicommodity flow over that. This seems like a reasonable baseline algorithm, given that random graphs demonstrate good expansion properties. The other approach is to have greedy algorithms, in the same vein as direct routing, for computing the matching sequence. While this has the advantage of being traffic matrix dependent (unlike random matchings), the exact gains obtained currently seem hard to analyze.

Chapter 4. A key real-world issue, that we have not addressed in our current model, is heterogeneity of peers. For example, user connections in the underlying IP-network could range from very fast fiber-optic networks to slow WiFi networks. The impact of such wide disparities in upload capacities and latencies on transient behavior is hard to predict analytically. Understanding this performance of the system-at-large (stability, delay and rate guarantees) in the limits of diverse heterogeneity through the framework of computer
simulations or a real world deployment is necessary. Another feature common in real systems, is the availability of multiple servers to source the stream. Analyzing the algorithm under other traffic patterns such as file-sharing and multiple multicast is also of interest.

Chapter 5. To be able to deploy Dandelion in the Bitcoin network, a few important factors have to be considered. First, we have analyzed honest-butcurious adversarial nodes. In practice, botnet nodes disobey protocol. In this case, anonymity can be negatively affected by nodes forwarding content inappropriately or misbehaving during graph construction. Hardening our protocols against such intrusions is critical. For example, noninteractive graph construction protocols offer some robustness by reducing opportunities for the adversary to lie in order to generate an advantageous anonymity graph. Alternatively, a system could use cryptographic proofs to ensure that nodes follow the graph construction protocol. Second, we have paid less attention to message latency by assuming the two-phase architecture of dandelion spreading. Understanding the anonymity-latency tradeoff is of fundamental interest.

## APPENDIX A

## PROOFS FOR CHAPTER 2

## A. 1 Background: Secret Key Agreement and Data Exchange

Our proofs draw from various techniques in cryptography and information theory. In particular, we rely heavily on recent results on information theoretic secret key agreement and data exchange, which are reviewed in this section together with the requisite background.

## A.1.1 Secret Key Agreement by Public Discussion

The problem of two-party secret key agreement by public discussion was alluded to in [201], but a proper formulation and an asymptotically optimal construction appeared first in [39, 40]. Consider two parties with the first and the second party, respectively, observing the random variable $X$ and $Y$. Using an interactive protocol $\pi$ and their local observations, the parties agree on a secret key. A random variable $K$ constitutes a secret key if the two parties form estimates that agree with $K$ with probability close to 1 and $K$ is concealed, in effect, from an eavesdropper with access to the transcript $\Pi$ and some side information $Z$. Formally, let $K_{\mathcal{X}}$ and $K_{\mathcal{Y}}$, respectively, be recoverable by $\pi$ for the first and the second party. Such random variables $K_{\mathcal{X}}$ and $K_{\mathcal{Y}}$ with common range $\mathcal{K}$ constitute an $\varepsilon$-secret key if the following condition is satisfied:

$$
d_{\text {var }}\left(\mathrm{P}_{K \mathcal{X} K \mathcal{Y} \Pi Z}, \mathrm{P}_{\text {unif }}^{(2)} \times \mathrm{P}_{\Pi Z}\right) \leq \varepsilon,
$$

where

$$
\mathrm{P}_{\text {unif }}^{(2)}\left(k_{\mathcal{X}}, k_{\mathcal{Y}}\right)=\frac{\mathbb{1}\left(k_{\mathcal{X}}=k_{\mathcal{Y}}\right)}{|\mathcal{K}|}
$$

and $\|\cdot\|$ is the variational distance. The condition above ensures both reliable recovery, requiring $\operatorname{Pr}\left(K_{\mathcal{X}} \neq K_{\mathcal{Y}}\right)$ to be small, and information theoretic secrecy, requiring the distribution of $K_{\mathcal{X}}$ (or $K_{\mathcal{Y}}$ ) to be almost independent of the eavesdropper's side information $(\Pi, Z)$ and to be almost uniform. See [22] for a discussion.

Definition A.1. Given $0 \leq \varepsilon<1$, the supremum over lengths $\log |\mathcal{K}|$ of an $\varepsilon$-secret key is denoted by $S_{\varepsilon}(X, Y \mid Z)$, and for the case when $Z$ is constant by $S_{\varepsilon}(X, Y)$.

By its definition, $S_{\varepsilon}(X, Y \mid Z)$ has the following monotonicity property.
Lemma A. 1 (Monotonicity). For any deterministic protocol $\pi$,

$$
S_{\varepsilon}(X, Y \mid Z) \geq S_{\varepsilon}(X \Pi, Y \Pi \mid Z \Pi)
$$

Furthermore, if $V_{\mathcal{X}}$ and $V_{\mathcal{Y}}$ can be recovered by $\pi$ for the first and the second party, respectively, then

$$
S_{\varepsilon}(X, Y \mid Z) \geq S_{\varepsilon}\left(X V_{\mathcal{X}}, V_{\mathcal{Y}} \mid Z \Pi\right)
$$

The claim holds since the two parties can generate a secret key by first running $\pi$ and then generating a secret key for the case when the first party observes $(X, \Pi)$, the second party observes $(Y, \Pi)$ and the eavesdropper observes $(Z, \Pi)$. Similarly, the second inequality holds since the parties can ignore a portion of their observations and generate a secret key from $\left(X, V_{\mathcal{X}}\right)$ and $\left(Y, V_{y}\right)$.

Leftover hash lemma. A key tool for generating secret keys is the leftover hash lemma [201, 202, 41, 42, 203] which, given a random variable $X$ and an $l$-bit eavesdropper's observation $Z$, allows us to extract roughly $H_{\min }\left(\mathrm{P}_{X}\right)-l$ bits of uniform bits, independent of $Z$. We shall use a slightly more general form. Given random variables $X$ and $Z$, let

$$
H_{\min }\left(\mathrm{P}_{X Z} \mid \mathrm{Q}_{Z}\right) \stackrel{\text { def }}{=} \sup _{x, z}-\log \frac{\mathrm{P}_{X Z}(x, z)}{\mathrm{Q}_{Z}(z)}
$$

We define ${ }^{1}$ the conditional min-entropy of $X$ given $Z$ by

$$
H_{\min }\left(\mathrm{P}_{X Z} \mid Z\right) \stackrel{\text { def }}{=} \sup _{\mathrm{Q}_{Z}: \operatorname{supp}\left(\mathrm{P}_{Z}\right) \subset \operatorname{supp}\left(\mathrm{Q}_{Z}\right)} H_{\min }\left(\mathrm{P}_{X Z} \mid \mathrm{Q}_{Z}\right) .
$$

Further, let $\mathcal{F}$ be a 2 -universal family of mappings $f: \mathcal{X} \rightarrow \mathcal{K}$, i.e., for each $x^{\prime} \neq x$, the family $\mathcal{F}$ satisfies

$$
\frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \mathbb{1}\left(f(x)=f\left(x^{\prime}\right)\right) \leq \frac{1}{|\mathcal{K}|}
$$

Lemma A. 2 (Leftover hash). Consider random variables $X, Z$ and $V$ taking values in countable sets $\mathcal{X}, \mathcal{Z}$, and a finite set $\mathcal{V}$, respectively. Let $S$ be a random seed such that $f_{S}$ is uniformly distributed over a 2-universal family $\mathcal{F}$. Then, for $K_{S}=f_{S}(X)$

$$
\mathbb{E}_{S}\left\{d_{\mathrm{var}}\left(\mathrm{P}_{K_{S} V Z}, \mathrm{P}_{\mathrm{unif}} \mathrm{P}_{V Z}\right)\right\} \leq \frac{1}{2} \sqrt{|\mathcal{K}||\mathcal{V}| 2^{-H_{\min }\left(\mathrm{P}_{X Z} \mid Z\right)}}
$$

where $\mathrm{P}_{\text {unif }}$ is the uniform distribution on $\mathcal{K}$.
The version above is a straightforward modification of the leftover hash lemma in, for instance, [203] and can be derived in a similar manner.

As an application of the leftover hash lemma above, we get the following useful result.

Lemma A.3. Consider random variables $X, Y, Z$ and $V$ taking values in countable sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, and a finite set $\mathcal{V}$, respectively. Then,

$$
S_{2 \varepsilon}(X, Y \mid Z V) \geq S_{\varepsilon}(X, Y \mid Z)-\log |\mathcal{V}|-2 \log (1 / 2 \varepsilon)
$$

The conditional independence testing upper bound for secret key lengths. Next, we recall the conditional independence testing upper bound for $S_{\varepsilon}(X, Y)$, which was established in [22, 23]. In fact, the general upper bound in $[22,23]$ is a single-shot upper bound on the secret key length for a multiparty secret key agreement problem with side information at the eavesdropper. Below, we recall a specialization of the general result for the

[^19]two-party case with no side information at the eavesdropper. In order to state the result, we need the following concept from binary hypothesis testing.

Consider a binary hypothesis testing problem with null hypothesis P and alternative hypothesis Q , where P and Q are distributions on the same alphabet $\mathcal{V}$. Upon observing a value $v \in \mathcal{V}$, the observer needs to decide if the value was generated by the distribution P or the distribution Q . To this end, the observer applies a stochastic test T , which is a conditional distribution on $\{0,1\}$ given an observation $v \in \mathcal{V}$. When $v \in \mathcal{V}$ is observed, the test T chooses the null hypothesis with probability $\mathrm{T}(0 \mid v)$ and the alternative hypothesis with probability $T(1 \mid v)=1-T(0 \mid v)$. For $0 \leq \varepsilon<1$, denote by $\beta_{\varepsilon}(\mathrm{P}, \mathrm{Q})$ the infimum of the probability of error of type II given that the probability of error of type I is less than $\varepsilon$, i.e.,

$$
\beta_{\varepsilon}(\mathrm{P}, \mathrm{Q}):=\inf _{\mathrm{T}: \mathrm{P}[\mathrm{~T}] \geq 1-\varepsilon} \mathrm{Q}[\mathrm{~T}],
$$

where

$$
\begin{aligned}
\mathrm{P}[\mathrm{~T}] & =\sum_{v} \mathrm{P}(v) \mathrm{T}(0 \mid v) \\
\mathrm{Q}[\mathrm{~T}] & =\sum_{v} \mathrm{Q}(v) \mathrm{T}(0 \mid v)
\end{aligned}
$$

The following upper bound for $S_{\varepsilon}(X, Y)$ was established in [22, 23].
Theorem A. 1 (Conditional independence testing bound). Given $0 \leq$ $\varepsilon<1,0<\eta<1-\varepsilon$, the following bound holds:

$$
S_{\varepsilon}(X, Y) \leq-\log \beta_{\varepsilon+\eta}\left(\mathrm{P}_{X Y}, \mathrm{Q}_{X} \mathrm{Q}_{Y}\right)+2 \log (1 / \eta)
$$

for all distributions $\mathrm{Q}_{X}$ and $\mathrm{Q}_{Y}$ on on $\mathcal{X}$ and $\mathcal{Y}$, respectively.
We close by noting a further upper bound for $\beta_{\varepsilon}(\mathrm{P}, \mathrm{Q})$, which is easy to derive.

Lemma A.4. For every $0 \leq \varepsilon<1$ and $\lambda$,

$$
-\log \beta_{\varepsilon}(\mathrm{P}, \mathrm{Q}) \leq \lambda-\log \left(\mathrm{P}\left(\log \frac{\mathrm{P}(X)}{\mathrm{Q}(X)}<\lambda\right)-\varepsilon\right)_{+}
$$

where $(x)_{+}=\max \{0, x\}$. As a corollary, we obtain the following upper bound
for $S_{\varepsilon}(X, Y)$ :
$S_{\varepsilon}(X, Y) \leq \lambda-\log \left(\operatorname{Pr}\left(\log \frac{\mathrm{P}_{X Y}(X, Y)}{\mathrm{Q}_{X}(X) \mathrm{Q}_{Y}(Y)}<\lambda\right)-\varepsilon-\eta\right)_{+}+2 \log (1 / \eta)$,
for all distributions $\mathrm{Q}_{X}$ and $\mathrm{Q}_{Y}$.

## A.1.2 The Data Exchange Problem

The next primitive that will be used in the reduction argument in our lower bound proof is a protocol for data exchange. The parties observing $X$ and $Y$ seek to know each other's data. What is the minimum length of interactive communication required? This basic problem, first studied in [204], is in effect a two-party extension of the classical Slepian-Wolf compression [27] (see [205] for a multiparty version). A recent work [51] derived tight lower and upper bounds for the length of a protocol that, for a given distribution $\mathrm{P}_{X Y}$, will facilitate data exchange with probability of error less than $\varepsilon$. We only review the proposed protocol and its performance here; first, we formally define the data exchange problem.

Definition A.2. For $0 \leq \varepsilon<1$, a protocol $\pi$ attains $\varepsilon$-data exchange if there exist $\hat{Y}$ and $\hat{X}$ which are recoverable by $\pi$ for the first and the second party, respectively, and satisfy

$$
\mathrm{P}(\hat{X}=X, \hat{Y}=Y) \geq 1-\varepsilon
$$

Note that data exchange corresponds to simulating a (deterministic) interactive protocol $\pi$ where $\Pi_{1}(X)=X$ and $\Pi_{2}=Y$; attaining $\varepsilon$-data exchange is tantamount to $\varepsilon$-simulation of $\pi$. In fact, the specific protocol for data exchange proposed in [51] can be recovered as a special case of our simulation protocol in Section A.3. The next result paraphrases [51, Theorem 2] and can also be recovered as a special case of Lemma A.11.

We paraphrase the result form [51] in a form that is more suited for our application here. The data exchange protocol proposed in [51] relies on slicing the spectrum of $h(X \mid Y)$ (or $h(Y \mid X)$ ). Let $\mathcal{E}_{\text {tail }}$ denote the tail event $h(X \mid Y) \notin\left[\lambda_{\text {min }}^{\prime}, \lambda_{\text {max }}^{\prime}\right]$. The protocol entails slicing the essential spectrum
$\left[\lambda_{\min }^{\prime}, \lambda_{\max }^{\prime}\right]$ into $N$ parts of length $\Delta$ each, i.e.,

$$
N=\frac{\lambda_{\max }^{\prime}-\lambda_{\min }^{\prime}}{\Delta}
$$

Theorem A. 2 ([51, Theorem 2], Lemma A.11). Given $\Delta>0, \xi>0$, and $N$ as above, there exists a deterministic protocol for $\varepsilon$-data exchange satisfying the following properties:
(i) denoting by $\mathcal{E}_{\text {error }}$ the error event, it holds that

$$
\mathrm{P}_{X Y}\left(\mathcal{E}_{\text {error }} \cap\{h(X \triangle Y) \leq \lambda\}\right) \leq \mathrm{P}_{X Y}\left(\mathcal{E}_{\text {tail }}\right)+N 2^{-\xi}
$$

which further yields that the probability of error $\varepsilon$ is bounded above as

$$
\varepsilon \leq \mathrm{P}_{X Y}(h(X \triangle Y)>\lambda)+\mathrm{P}_{X Y}\left(\mathcal{E}_{\text {tail }}\right)+N 2^{-\xi}
$$

(ii) the protocol communicates no more than $\lambda+\Delta+N+\xi$ bits;
(iii) for every $(X, Y)$ such that $\lambda_{\min }^{\prime}<h(X \mid Y)<\lambda_{\max }^{\prime}$, the transcript of the protocol can take no more than $2^{h(X \Delta Y)+\Delta+\xi}$ values.

Note that property (iii) above, though not explicitly stated in [51, Theorem 2] or in the general Lemma A. 11 below, follows simply from the proofs of these results. It makes the subtle observation that while, for each $(X, Y)$ such that $\lambda_{\min }^{\prime}<h(X \mid Y)<\lambda_{\max }^{\prime}, h(X \triangle Y)+\Delta+N+\xi$ bits are communicated to interactively generate the transcript, the number of (variable length) transcripts is no more than ${ }^{2} h(X \triangle Y)+\Delta+N+\xi$. Property (ii) above was crucial to establish the communication complexity results of [51]; property (iii) was not relevant in the context of that work. On the other hand, here we shall use the protocol of Theorem A. 2 in our reduction to secret key agreement in the next section and will treat the communication used in data exchange as eavesdropper's side information. As such, it suffices to bound the number of values taken by the transcript; the number of bits actually communicated in the interactive protocol is a loose upper bound on the former quantity.

[^20]It is perhaps interesting that our simulation protocol given in Section A. 3 is used both in our upper bound to compress a given protocol and in our lower bound to complete the reduction argument.

## A. 2 Section 2.2: Lower Bound

Our proof of Theorem 2.1 relies on a reduction argument that utilizes an $\varepsilon$-simulation to generate an information theoretically secure secret key for $X$ and $Y$. Heuristically, a protocol can be simulated using fewer bits of communication than its length because of the correlation in the observations $X$ and $Y$. Due to this correlation, when simulating the protocol, the parties agree on more bits (generate common randomness) than what they communicate. These extra bits can be extracted as an information theoretically secure secret key for the two parties. A lower bound on the number of bits communicated can be derived using an upper bound for the maximum possible length of a secret key that can be generated using interactive communication; the latter was derived recently in $[22,23]$. However, there are two caveats in this heuristic approach.

First, to extract secret keys from the generated common randomness we rely on the leftover hash lemma. In particular, the bits are extracted by applying a 2-universal hash family to the common randomness generated. However, the range-size of the hash family must be selected based on the min-entropy of the generated common randomness, which is not easy to estimate. To remedy this, we communicate more using a data-exchange protocol proposed in [51] to make the collective observations ( $X, Y$ ) available to both the parties; a good bound for the communication complexity of this protocol is available. The generated common randomness now includes $(X, Y)$ for which the min-entropy can be easily bounded and the size of the aforementioned extracted secret key can be tracked. A similar common randomness completion and decomposition technique was introduced in [206] to characterize a class of securely computable functions.

Second, our methodology described above requires bounds on various information densities in different directions. A direct application of this method will result in a gap equal to the effective length of various spectrums involved. To remedy this, we apply the methodology described above not to
the original distribution $\mathrm{P}_{X Y}$ but a conditional distribution $\mathrm{P}_{X Y \mid \mathcal{E}}$ where the event $\mathcal{E}$ is an appropriately chosen event contained in single slices of various spectrums involved. Such a conditioning is allowed since we are interested in the worst-case communication complexity of the simulation protocol.

We now describe the proof of Theorem 2.1 in detail. To make the exposition clear, we have divided the proof into steps.

Given a (private coin) protocol $\pi$, let $\pi_{\text {sim }}$ be its $\varepsilon$-simulation and $\Pi_{\mathcal{X}}$ and $\Pi_{\mathcal{Y}}$ be the corresponding estimates of the transcript $\Pi$ for Party 1 and Party 2 , respectively.

## A.2.1 From Simulation to Probability of Error

We first use a coupling argument to replace the $\varepsilon$-simulation condition with an $\varepsilon$ probability of error condition. Recall the maximal coupling lemma.

Lemma A. 5 (Maximal coupling lemma [207]). For any two distributions P and Q on the same set, there exists a joint distribution $\mathrm{P}_{X Y}$ with $X \sim \mathrm{P}$ and $Y \sim \mathrm{Q}$ such that

$$
\operatorname{Pr}(X \neq Y)=d_{\mathrm{var}}(\mathrm{P}, \mathrm{Q})
$$

Using the maximal coupling lemma, for each fixed $x, y$ there exists a joint distribution $\mathrm{P}_{\Pi \Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \mid X=x, Y=y}$ such that
$\operatorname{Pr}\left(\Pi=\Pi_{\mathcal{X}}=\Pi_{\mathcal{Y}} \mid X=x, Y=y\right)=1-d_{\mathrm{var}}\left(\mathrm{P}_{\Pi \Pi \mid X=x, Y=y}, \mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \mid X=x, Y=y}\right)$.

Consequently,

$$
\begin{align*}
\operatorname{Pr}\left(\Pi=\Pi_{\mathcal{X}}=\Pi_{\mathcal{Y}}\right) & =1-\sum_{x, y} \mathrm{P}_{X Y}(x, y) d_{\mathrm{var}}\left(\mathrm{P}_{\Pi \Pi \mid X=x, Y=y}, \mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \mid X=x, Y=y}\right) \\
& =1-d_{\mathrm{var}}\left(\mathrm{P}_{\Pi \Pi X Y}, \mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} X Y}\right) \\
& \geq 1-\varepsilon . \tag{A.1}
\end{align*}
$$

As pointed in Section 2.1, we restrict ourselves to public coin protocols $\pi_{\text {sim }}$ using shared public randomness $U$. For concreteness (and convenience of
proof), we define the joint distribution for $\left(\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} X Y U\right)$ as

$$
\begin{equation*}
\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \Pi X Y}=\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \Pi X Y} \mathrm{P}_{U \mid \Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} X Y} \tag{A.2}
\end{equation*}
$$

Note that the marginal $\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} X Y U}$ remains as in the original protocol. In particular, $(X, Y)$ is jointly independent of $U$.

## A.2.2 From Partial Knowledge to Omniscience

As explained in the heuristic proof above, instead of extracting a secret key from the common randomness generated by the protocol $\pi_{\text {sim }}$, we first use the data exchange protocol of Theorem A. 2 to make all the data available to both the parties, which was termed attaining omniscience ${ }^{3}$ in [205]. In particular, the parties run the protocol $\pi_{\text {sim }}$ followed by a data exchange protocol for $(X \Pi, Y \Pi)$ to recover $(X, Y)$ at both the parties. Once both the parties have access to $(X, Y)$, they can extract a secret key from $(X, Y)$ which will be used in the reduction in our final step.
Formally, with the notations introduced in Section A.1.2, let $\pi_{\mathrm{DE}}$ be the data exchange protocol of Theorem A. 2 with $X$ and $Y$ replaced by ( $Х \Pi$ ) and $(Y \Pi)$, respectively, with $N_{2}$ and $\Delta_{2}$ denoting $N$ and $\Delta$, respectively, and with $\lambda=\lambda_{\text {max }}^{(3)}, \lambda_{\text {min }}^{\prime}=\lambda_{\text {min }}^{(2)}, \lambda_{\text {max }}^{\prime}=\lambda_{\text {max }}^{(2)}$. Then, denoting by $\mathcal{E}_{\text {error }}$ the error event for the protocol $\pi_{\mathrm{DE}}$ Theorem A. 2 (i) yields

$$
\begin{equation*}
\operatorname{Pr}\left(\mathcal{E}_{\text {error }} \cap \mathcal{E}_{3}^{c}\right) \leq \operatorname{Pr}\left(\mathcal{E}_{2}\right)+N_{2} 2^{-\eta^{\prime}}, \tag{A.3}
\end{equation*}
$$

where $\mathcal{E}_{2}$ and $\mathcal{E}_{3}$ are as in Equation (2.3). Furthermore, for every realization $(X, Y) \notin \mathcal{E}_{3}$ the number possible transcripts $\Pi_{\mathrm{DE}}$ is no more than

$$
\begin{equation*}
2^{h(X \Pi \triangle Y \Pi)+\Delta_{2}+\xi} \tag{A.4}
\end{equation*}
$$

We seek to use $\pi_{\mathrm{DE}}$ for recovering $Y$ and $X$, respectively, at Party 1 and Party 2 by running $\pi_{\text {DE }}$ successively after $\pi_{\text {sim }}$. However, $\pi_{\text {sim }}$ yields $X \Pi_{\mathcal{X}}$ and $Y \Pi_{\mathcal{Y}}$ at Party 1 and Party 2, respectively, while the data exchange protocol $\pi_{\mathrm{DE}}$ facilitates data exchange when the two parties observe $X \Pi$ and $Y \Pi$. We

[^21]can easily fix this gap using Equation (A.1).
Specifically, denote by $\hat{X}$ and $\hat{Y}$ the estimates of $X$ and $Y$ formed at Party 2 and Party 1 in $\pi_{\mathrm{DE}}$. Note that $\pi_{\mathrm{DE}}$ is a deterministic protocol and $\hat{X}$ and $\hat{Y}$ are functions of $(X, Y, \Pi, \Pi)$. Denote by $\mathcal{A}$ the set
$$
\mathcal{A}=\left\{\left(\tau_{\mathcal{X}}, \tau_{\mathcal{Y}}, \tau, x, y\right): \tau_{\mathcal{X}}=\tau_{\mathcal{Y}}=\tau\right\}
$$
and by $\mathcal{B}$ the set
$$
\mathcal{B}=\left\{\left(\tau_{\mathcal{X}}, \tau_{\mathcal{Y}}, \tau, x, y\right): \hat{X}(x, y, \tau, \tau)=x, \hat{Y}(x, y, \tau, \tau)=y\right\}
$$
which is the same as $\mathcal{E}_{\text {error }}^{c}$ for $\mathcal{E}_{\text {error }}$ in Equation (A.3). Then, by Equations (A.1) and (A.3)
\[

$$
\begin{align*}
& \operatorname{Pr}\left(\left\{\hat{X}\left(X, Y, \Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right)=X, \hat{Y}\left(X, Y, \Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right)=Y\right\} \cap \mathcal{E}_{3}^{c}\right) \\
& \geq \mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \Pi X Y}\left(\mathcal{A} \cap \mathcal{B} \cap \mathcal{E}_{3}^{c}\right) \\
& \geq \mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \Pi X Y}(\mathcal{A})+\operatorname{Pr}\left(\mathcal{E}_{3}^{c}\right)-\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \Pi X Y}\left(\mathcal{B}^{c} \cap \mathcal{E}_{3}^{c}\right)-1 \\
& \geq 1-\varepsilon-\operatorname{Pr}\left(\mathcal{E}_{2}\right)-\operatorname{Pr}\left(\mathcal{E}_{3}\right)-N_{2} 2^{-\xi} . \tag{A.5}
\end{align*}
$$
\]

Note that it follows that the combined protocol $\left(\pi_{\text {sim }}, \pi_{\text {DE }}\right)$ attains $\left(\varepsilon+\operatorname{Pr}\left(\mathcal{E}_{2}\right)+\right.$ $N_{2} 2^{-\xi}$ )-data exchange for $X$ and $Y$. The partial bound in Equation (A.5) is more suited for our analysis since we shall account separately for the tail event $\mathcal{E}_{3}$.

## A.2.3 From Simulation to Secret Keys: A Sketch of the Reduction

The first step in our proof is to replace the simulation condition Equation (2.1) with the probability of error condition Equation (A.1) for the joint distribution $\mathrm{P}_{\Pi \Pi \Pi_{\mathcal{X}} \Pi_{\mathcal{Y} X Y U}}$ in Equation (A.2).

Next, as described in our heuristic arguments above, we "complete the common randomness," i.e., we communicate more to facilitate the recovery of $Y$ and $X$ at Party 1 and Party 2, respectively. To that end, upon executing $\pi_{\text {sim }}$, the parties run the data exchange protocol $\pi_{\mathrm{DE}}$ of Theorem A. 2 for $(X \Pi)$ and $(Y \Pi)$, with $\left(X, \Pi_{\mathcal{X}}\right)$ and $\left(Y, \Pi_{\mathcal{Y}}\right)$ in place of $(X \Pi)$ and $(Y \Pi)$, respectively. The condition in Equation (A.1) guarantees that the combined
protocol $\left(\pi_{\text {sim }}, \pi_{\mathrm{DE}}\right)$ recovers $Y$ and $X$ at Party 1 and Party 2 with probability of error less than $\varepsilon$.

We now sketch our reduction argument. Consider the secret key agreement for $X$ and $Y$ when the eavesdropper observes $U$. By the independence of ( $X, Y$ ) and $U, S_{\eta}(X, Y \mid U)=S_{\eta}(X, Y)$, and further, the result of [22] shows that $S_{\eta}(X, Y)$ is bounded above, roughly, by the mutual information density $i(X \wedge Y)=\log \mathrm{P}_{X Y}(X, Y) / \mathrm{P}_{X}(X) \mathrm{P}_{Y}(Y)$, i.e.,

$$
\begin{equation*}
S_{\eta}(X U, Y U \mid U)=S_{\eta}(X, Y) \lesssim i(X \wedge Y) \tag{A.6}
\end{equation*}
$$

On the other hand, we can generate a secret key using the following protocol:

1. Run the combined protocol $\left(\pi_{\text {sim }}, \pi_{\mathrm{DE}}\right)$ to attain data exchange for $X$ and $Y$, resulting in a common randomness of size roughly $h(X, Y \mid U)=$ $h(X, Y)$.
2. The data exchange protocol $\pi_{\mathrm{DE}}$ for $(X \Pi)$ and $(Y \Pi)$ communicates roughly $h(X \Pi \triangle Y \Pi)$ bits for every fixed realization $(X, Y, \Pi)$. Thus, the combined protocol $\left(\pi_{\text {sim }}, \pi_{\mathrm{DE}}\right)$, which allows both the parties to recover $(X, Y)$, communicates no more than $\log \left\|\pi_{\text {sim }}\right\|+h(X \Pi \triangle Y \Pi)$ bits for every fixed realization $(X, Y, \Pi)$. Using the leftover hash lemma, we can extract a secret key of rate roughly $h(X, Y)-\log \left\|\pi_{\text {sim }}\right\|-$ $h(X \Pi \triangle Y \Pi)$.

The following approximate inequalities summarize our reduction:

$$
\begin{align*}
S_{\eta}(X U, Y U \mid U) & \geq S_{\eta}\left(X U, Y U \mid \Pi_{\mathrm{sim}} \Pi_{\mathrm{DE}} U\right) \\
& \geq S_{\eta}\left(X \hat{Y}, \hat{X} Y \mid \Pi_{\mathrm{sim}} \Pi_{\mathrm{DE}} U\right) \\
& \gtrsim S_{\eta}(X \hat{Y}, \hat{X} Y \mid U)-\log \left\|\pi_{\mathrm{sim}}\right\|-h(X \Pi \triangle Y \Pi) \\
& \approx h(X, Y)-\log \left\|\pi_{\mathrm{sim}}\right\|-h(X \Pi \triangle Y \Pi), \tag{A.7}
\end{align*}
$$

where the first and the second inequalities hold since we can execute the protocol $\left(\pi_{\text {sim }}, \pi_{\mathrm{DE}}\right)$ and recover $\hat{Y}$ and $\hat{X}$ at Party 1 and Party 2, before generating a secret key for the $S_{\eta}(X U, Y U \mid U)$ problem.

We note that generation of secret keys from data exchange was first proposed in [205] in an amortized, IID setup and was shown to yield a secret key of asymptotically optimal rate.

From Equations (A.6) and (A.7) it follows that

$$
\log \left\|\pi_{\mathrm{sim}}\right\| \gtrsim h(X, Y)-h(X \Pi \triangle Y \Pi)-i(X \wedge Y)=\mathrm{ic}(\Pi ; X, Y)
$$

which is the required lower bound.
Of course, the steps above are not precise. We have used instantaneous communication and common randomness lengths in our bounds whereas a formal treatment will require us to use worst-case performance bounds for these quantities. Unfortunately, such worst-case bounds do not yield our desired lower bound for $D_{\varepsilon}(\pi)$. To fill this gap, we apply the arguments above not for the original distribution $\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \Pi X Y U}$ but for the conditional distribution $\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} \Pi X Y U \mid \mathcal{E}}$ where the event $\mathcal{E}$ is carefully constructed in such a manner that the aforementioned worst-case bounds are close to instantaneous bounds for all realizations. Specifically, $\mathcal{E}$ is selected by appropriately slicing the spectrums of the various information densities that appear in the worstcase bounds.

## A.2.4 From Original to Conditional Probabilities

To identify an appropriate critical event for conditioning, we take recourse to spectrum slicing. Specifically, we identify an appropriate subset of intersection of slices of spectrums (ii) and (iv) described in Section 2.2.1. For the combined protocol $\left(\pi_{\text {sim }}, \pi_{\text {DE }}\right)$ and the estimates $(\hat{X}, \hat{Y})$ as above, and $\lambda_{\text {min }}^{(i)}, \lambda_{\text {max }}^{(i)}, i=1,2,3$, as in Section 2.2.1, let

$$
\begin{aligned}
\mathcal{E}_{\mathrm{sim}} & =\left\{\Pi=\Pi_{\mathcal{X}}=\Pi_{\mathcal{Y}}\right\} \\
\mathcal{E}_{\mathrm{DE}} & =\left\{\hat{X}\left(X, Y, \Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right)=X, \hat{Y}\left(X, Y, \Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right)=Y\right\} \\
\mathcal{E}_{\lambda} & =\{\mathrm{ic}(\Pi ; X, Y) \geq \lambda\} \\
\mathcal{E}_{i}^{(1)} & =\left\{\lambda_{\min }^{(1)}+(i-1) \Delta_{1} \leq h(X, Y) \leq \lambda_{\min }^{(1)}+i \Delta_{1}\right\}, \quad 1 \leq i \leq N_{1}, \\
\mathcal{E}_{j}^{(3)} & =\left\{\lambda_{\min }^{(3)}+(j-1) \Delta_{3} \leq h(X \Pi \triangle Y \Pi) \leq \lambda_{\min }^{(3)}+j \Delta_{3}\right\}, \quad 1 \leq j \leq N_{3},
\end{aligned}
$$

where

$$
N_{1}=\frac{\lambda_{\max }^{(1)}-\lambda_{\min }^{(1)}}{\Delta_{1}} \text { and } N_{3}=\frac{\lambda_{\max }^{(3)}-\lambda_{\min }^{(3)}}{\Delta_{3}}
$$

Note that $\cup_{i} \mathcal{E}_{i}^{(1)}=\mathcal{E}_{1}^{c}$ and $\cup_{j} \mathcal{E}_{j}^{(3)}=\mathcal{E}_{3}^{c}$, where the events $\mathcal{E}_{1}$ and $\mathcal{E}_{3}$ are as in Equation (2.3). Finally, define the event $\mathcal{E}_{i j}$ as follows:

$$
\mathcal{E}_{i j}=\mathcal{E}_{\mathrm{sim}} \cap \mathcal{E}_{\mathrm{DE}} \cap \mathcal{E}_{\lambda} \cap \mathcal{E}_{i}^{(1)} \cap \mathcal{E}_{j}^{(3)}, \quad 1 \leq i \leq N_{1}, 1 \leq j \leq N_{3} .
$$

The next lemma says that (at least) one of the events $\mathcal{E}_{i j}$ has significant probability, and this particular event will be used as the critical event in our proofs.

Lemma A.6. There exists $i, j$ such that

$$
\begin{equation*}
\operatorname{Pr}\left(\mathcal{E}_{i j}\right) \geq \frac{\operatorname{Pr}\left(\mathcal{E}_{\lambda}\right)-\varepsilon-\varepsilon_{\text {tail }}-N_{2} 2^{-\xi}}{N_{1} N_{3}} \stackrel{\text { def }}{=} \alpha . \tag{A.8}
\end{equation*}
$$

Proof. Note that the event $\mathcal{E}_{\text {sim }} \cap \mathcal{E}_{\text {DE }} \cap \mathcal{E}_{3}^{c}$ is the same as the event $\mathcal{A} \cap \mathcal{B} \cap \mathcal{E}_{3}^{c}$ of Equation (A.5). Therefore,

$$
\begin{aligned}
\operatorname{Pr}\left(\mathcal{E}_{\mathrm{sim}} \cap \mathcal{E}_{\mathrm{DE}} \cap \mathcal{E}_{\lambda} \cap \mathcal{E}_{1}^{c} \cap \mathcal{E}_{3}^{c}\right) \geq & \operatorname{Pr}\left(\mathcal{E}_{\lambda}\right)+\operatorname{Pr}\left(\mathcal{E}_{\mathrm{sim}} \cap \mathcal{E}_{\mathrm{DE}} \cap \mathcal{E}_{3}^{c}\right)+\operatorname{Pr}\left(\mathcal{E}_{1}^{c}\right)-2 \\
\geq & \operatorname{Pr}\left(\mathcal{E}_{\lambda}\right)-\varepsilon-\operatorname{Pr}\left(\mathcal{E}_{2}\right)-\operatorname{Pr}\left(\mathcal{E}_{3}\right)-N_{2} 2^{-\xi} \\
& -\operatorname{Pr}\left(\mathcal{E}_{1}\right) \\
\geq & \operatorname{Pr}\left(\mathcal{E}_{\lambda}\right)-\varepsilon-\varepsilon_{\text {tail }}-N_{2} 2^{-\xi},
\end{aligned}
$$

where the second inequality uses Equation (A.5) and and the third uses Equation (2.3). The proof is completed upon noting that $\left\{\mathcal{E}_{i j}\right\}_{i, j}$ constitutes a partition of $\mathcal{E}_{\text {sim }} \cap \mathcal{E}_{\text {DE }} \cap \mathcal{E}_{\lambda} \cap \mathcal{E}_{1}^{c} \cap \mathcal{E}_{3}^{c}$ with $N_{1} N_{3}$ parts.

## A.2.5 From Simulation to Secret Keys: The Formal Reduction Proof

We are now in a position to complete the proof of our lower bound. For brevity, let $\mathcal{E}$ denote the event $\mathcal{E}_{i j}$ of Lemma A. 6 satisfying $\operatorname{Pr}(\mathcal{E}) \geq \alpha$.

Our proof essentially formalizes the steps outlined in Section A.2.3, but for the conditional distribution given $\mathcal{E}$. With an abuse of notation, let $S_{\eta}(X, Y \mid Z, \mathcal{E})$ denote the maximum length of an $\eta$-secret key for two parties observing $X$ and $Y$, and the eavesdropper's side information $Z$, when the distribution of $(X, Y, Z)$ is given by $\mathrm{P}_{X Y Z \mid \mathcal{E}}$. Then, using Lemma A. 4 with $\mathrm{Q}_{X}=\mathrm{P}_{X}$ and $\mathrm{Q}_{Y}=\mathrm{P}_{Y}$, we get the following bound in place of Equa-
tion (A.6):

$$
\begin{align*}
& S_{2 \eta}(X, Y \mid \mathcal{E}) \leq \gamma-\log \left(\operatorname{Pr}\left(\left.\left\{(x, y): \log \frac{\mathrm{P}_{X Y \mid \mathcal{E}}(x, y)}{\mathrm{P}_{X}(x) \mathrm{P}_{Y}(y)}<\gamma\right\} \right\rvert\, \mathcal{E}\right)-3 \eta\right)_{+} \\
&+2 \log (1 / \eta) \\
& \leq \gamma-\log \left(\operatorname{Pr}\left(\left.\left\{(x, y): \log \frac{\mathrm{P}_{X Y}(x, y)}{\mathrm{P}_{X}(x) \mathrm{P}_{Y}(y)}<\gamma+\log \alpha\right\} \right\rvert\, \mathcal{E}\right)-3 \eta\right)_{+} \\
&+2 \log (1 / \eta) \tag{А.9}
\end{align*}
$$

where $0<\eta<1 / 3$ is arbitrary and in the previous inequality we have used

$$
\mathrm{P}_{X Y \mid \mathcal{E}}(x, y \mid \mathcal{E}) \leq \frac{\mathrm{P}_{X Y}(x, y)}{\operatorname{Pr}(\mathcal{E})} \leq \frac{\mathrm{P}_{X Y}(x, y)}{\alpha}
$$

To replace Equation (A.7), first note that the following inequalities hold simply by the definition of $S_{\eta}(X, Y \mid \mathcal{E})$

$$
\begin{align*}
S_{2 \eta}(X, Y \mid \mathcal{E}) & \geq S_{2 \eta}(X U, Y U \mid U, \mathcal{E}) \\
& \geq S_{2 \eta}\left(X \Pi_{\mathrm{sim}} \Pi_{\mathrm{DE}}, Y \Pi_{\mathrm{sim}} \Pi_{\mathrm{DE}} \mid U, \Pi_{\mathrm{sim}}, \Pi_{\mathrm{DE}}, \mathcal{E}\right) \\
& \geq S_{2 \eta}\left(X \hat{Y}, \hat{X} Y \mid U, \Pi_{\mathrm{sim}}, \Pi_{\mathrm{DE}}, \mathcal{E}\right), \tag{A.10}
\end{align*}
$$

where we have used the fact that we can always generate a secret key for $X$ and $Y$ by first running the combined protocol $\left(\pi_{\mathrm{sim}}, \pi_{\mathrm{DE}}\right)$, recovering estimates $\hat{Y}$ and $\hat{X}$ at Party 1 and Party 2, and finally, since now Party 1 observes $(X, \hat{Y})$, Party 2 observes $(\hat{X}, Y)$ and the eavesdropper observes $\left(U, \Pi_{\mathrm{sim}}, \Pi_{\mathrm{DE}}\right)$, generating a secret key attaining $S_{2 \eta}\left(X \hat{Y}, \hat{X} Y \mid U, \Pi_{\text {sim }} \Pi_{\mathrm{DE}} \mathcal{E}\right)$.

Next, note that by Equation (A.4) the transcript $\Pi_{\text {sim }} \Pi_{D E}$ takes no more than $\left\|\pi_{\text {sim }}\right\| 2^{h(X \Pi \triangle Y \Pi)+\Delta_{2}+\xi}$ values for every realization $(X, Y) \notin \mathcal{E}_{3}$. However, when the event $\mathcal{E}=\mathcal{E}_{i j}$ holds, $h(X \Pi \triangle Y \Pi) \leq \lambda_{\text {min }}^{(3)}+j \Delta_{3}$. It follows by Lemma A. 3 that

$$
\begin{align*}
& S_{2 \eta}\left(X \hat{Y}, \hat{X} Y \mid U \Pi_{\mathrm{sim}} \Pi_{\mathrm{DE}}, \mathcal{E}\right) \\
& \geq S_{\eta}(X \hat{Y}, \hat{X} Y \mid U, \mathcal{E})-\log \left\|\pi_{\mathrm{sim}}\right\|-\lambda_{\min }^{(3)}-j \Delta_{3}-\Delta_{2}-\xi-2 \log (1 / 2 \eta) \tag{A.11}
\end{align*}
$$

Also, since $\{X=\hat{X}, Y=\hat{Y}\}$ holds when we condition on $\mathcal{E}$,

$$
\begin{align*}
S_{\eta}(X \hat{Y}, \hat{X} Y \mid U, \mathcal{E}) & =S_{\eta}(X Y, X Y \mid U, \mathcal{E}) \\
& \geq H_{\min }\left(\mathrm{P}_{X Y U \mid \mathcal{E}} \mid U\right)-2 \log (1 / 2 \eta) \tag{A.12}
\end{align*}
$$

where the previous inequality is by the leftover hash lemma. Furthermore, by using

$$
\mathrm{P}_{X Y U \mid \mathcal{E}}(x, y, u) \leq \frac{\mathrm{P}_{X Y U}(x, y, u)}{\operatorname{Pr}(\mathcal{E})} \leq \frac{\mathrm{P}_{X Y U}(x, y, u)}{\alpha}
$$

we can bound $H_{\min }\left(\mathrm{P}_{X Y U \mid \mathcal{E}} \mid U\right)$ as follows:

$$
\begin{align*}
H_{\min }\left(\mathrm{P}_{X Y U \mid \mathcal{E}} \mid U\right) & \geq \min _{x, y, u}-\log \frac{\mathrm{P}_{X Y U \mid \mathcal{E}}(x, y, u)}{\mathrm{P}_{U}(u)} \\
& \geq \min _{x, y, u}-\log \frac{\mathrm{P}_{X Y U}(x, y, u) \mathbb{1}\left(\mathrm{P}_{X Y U \mid \mathcal{E}}(x, y, u)>0\right)}{\alpha \mathrm{P}_{U}(u)} \\
& =\min _{x, y \in \mathcal{E}_{i}^{(1)}} h_{\mathrm{P}_{X Y}}(x, y)+\log \alpha \\
& \geq \lambda_{\min }^{(1)}+(i-1) \Delta_{1}+\log \alpha . \tag{A.13}
\end{align*}
$$

Thus, on combining Equations (A.10)-(A.13), we get

$$
\begin{align*}
S_{2 \eta}(X, Y \mid \mathcal{E}) \geq & {\left[\lambda_{\min }^{(1)}+(i-1) \Delta_{1}-\lambda_{\min }^{(3)}-j \Delta_{3}+\log \alpha\right]-\Delta_{2}-\xi } \\
& -4 \log (1 / 2 \eta)-\log \left\|\pi_{\text {sim }}\right\| . \tag{A.14}
\end{align*}
$$

To get a matching form of the upper bound Equation (A.9) for $S_{2 \eta}(X, Y \mid \mathcal{E})$, note that since ${ }^{4}$

$$
-\mathrm{i} \mathrm{c}_{\mathrm{P}_{\Pi X Y}}(\tau ; x, y)=i_{\mathrm{P}_{X Y}}(x \wedge y)-h_{\mathrm{P}_{X Y}}(x, y)+h_{\mathrm{P}_{\Pi X Y}}((x, \tau) \triangle(y, \tau)),
$$

and since under $\mathcal{E}$

$$
\begin{aligned}
h_{\mathrm{P}_{X Y}}(x, y) & \leq \lambda_{\min }^{(1)}+i \Delta_{1}, \\
h_{\mathrm{P}_{X Y \Pi}}((x, \tau) \triangle(y, \tau)) & \geq \lambda_{\min }^{(3)}+(j-1) \Delta_{3},
\end{aligned}
$$

[^22]it holds that
\[

$$
\begin{aligned}
& \operatorname{Pr}\left(\left\{(x, y): i_{\mathrm{P}_{X Y}}(x \wedge y)<\gamma+\log \alpha\right\} \mid \mathcal{E}\right) \\
& \geq \operatorname{Pr}\left(\left\{(x, y, \tau):-i{c_{\mathrm{P}_{X Y \Pi}}(x, y, \tau)<\gamma-\lambda_{\min }^{(1)}-i \Delta_{1}+\lambda_{\min }^{(3)}+(j-1) \Delta_{3}}_{\quad+\log \alpha\} \mid \mathcal{E}) .} .\right.\right.
\end{aligned}
$$
\]

On choosing

$$
\gamma=-\lambda+\lambda_{\min }^{(1)}+i \Delta_{1}-\lambda_{\min }^{(3)}-(j-1) \Delta_{3}-\log \alpha
$$

it follows from Equation (A.9) that

$$
\begin{align*}
& S_{2 \eta}(X, Y \mid \mathcal{E}) \\
& \leq-\lambda+\left[\lambda_{\min }^{(1)}+i \Delta_{1}-\lambda_{\min }^{(3)}-(j-1) \Delta_{3}-\log \alpha\right]-\log \left(\operatorname{Pr}\left(\mathcal{E}_{\lambda} \mid \mathcal{E}\right)-3 \eta\right)_{+} \\
& \quad+2 \log (1 / \eta) \\
& \leq-\lambda+\left[\lambda_{\min }^{(1)}+i \Delta_{1}-\lambda_{\min }^{(3)}-(j-1) \Delta_{3}-\log \alpha\right]-\log (1-3 \eta)+2 \log (1 / \eta) \tag{A.15}
\end{align*}
$$

where the equality holds since $\operatorname{Pr}\left(\mathcal{E}_{\lambda} \mid \mathcal{E}\right)=1$.
Thus, by Equations (A.14) and (A.15), we get

$$
\begin{aligned}
\log \left\|\pi_{\text {sim }}\right\| \geq & \lambda+2 \log \alpha-\Delta_{1}-\Delta_{2}-\Delta_{3}-\xi-6 \log (1 / \eta)+\log (1-3 \eta)+4 \\
= & \lambda+2 \log \left(\operatorname{Pr}\left(\mathcal{E}_{\lambda}\right)-\varepsilon-\varepsilon_{\text {tail }}-\eta\right)-2 \log N_{1} N_{3} \\
& -\left(\Delta_{1}+\Delta_{2}+\Delta_{3}\right)-\log N_{2}-7 \log (1 / \eta)+\log (1-3 \eta)+4
\end{aligned}
$$

where the equality holds for $\eta^{\prime}=-\log \eta+\log N_{2}$. Note that the maximum value of the right-side above, when maximized over $N_{i}$ and $\Delta_{i}$ under the constraint $N_{i} \Delta_{i}=\Lambda_{i}, i=1,2,3$, occurs for $\Delta_{1}=\Delta_{3}=2$ and $\Delta_{2}=1$. Substituting this choice of parameters, we get

$$
\begin{aligned}
\log \left\|\pi_{\text {sim }}\right\| \geq & \lambda+2 \log \left(\operatorname{Pr}\left(\mathcal{E}_{\lambda}\right)-\varepsilon-\varepsilon_{\text {tail }}-\eta\right)-2 \log \Lambda_{1} \Lambda_{3}-\log \Lambda_{2} \\
& -7 \log (1 / \eta)+\log (1-3 \eta)+3 \\
\geq & \lambda-2 \log \Lambda_{1} \Lambda_{3}-\log \Lambda_{2}-9 \log (1 / \eta)+\log (1-3 \eta)+3
\end{aligned}
$$

where the final inequality holds for every $\lambda$ such that $\operatorname{Pr}\left(\mathcal{E}_{\lambda}\right) \geq \varepsilon+\varepsilon_{\text {tail }}+2 \eta$;

Theorem 2.1 follows upon maximizing the right side-over all such $\lambda$.

## A. 3 Section 2.2: Upper Bound

In this section, we formally present an $\varepsilon$-simulation of a given interactive protocol $\pi$ with bounded rounds. For clarity, we build the simulation protocol in steps.

## A.3.1 Sending $X$ Using One-Sided Communication

We start with the well-known Slepian-Wolf compression problem [27] where Party 1 wants to transmit $X$ itself to Party 2 using as few bits as possible. This corresponds to simulating the deterministic protocol $\Pi=\Pi_{1}=X$. See Remark 1 in Section 2.1 for a discussion on simulation of deterministic protocols.

For encoder, we use a hash function that is randomly chosen from a 2 universal hash family $\mathcal{F}_{l}(\mathcal{X})$; for decoder, we use a kind of joint typical decoder [11]. Let the typical set $\mathcal{T}_{\mathrm{P}_{X \mid Y}}$ be given by

$$
\begin{equation*}
\mathcal{T}_{\mathrm{P}_{X \mid Y}}=\left\{(x, y): h_{\mathrm{P}_{X \mid Y}}(x \mid y) \leq l-\gamma\right\}, \tag{A.16}
\end{equation*}
$$

for a slack parameter $\gamma>0$. Our first protocol is given below:

```
Protocol 10: Slepian-Wolf compression
    Input: Observations \(X\) and \(Y\), uniform public randomness \(U_{\text {hash }}\), and
            a parameter \(l\)
    Output: Estimate \(\hat{X}\) of \(X\) at party 2
    Both parties use \(U_{\text {hash }}\) to select \(f\) from \(\mathcal{F}_{l}(\mathcal{X})\)
    Party 1 sends \(\Pi_{\text {sim }, 1}=f(X)\)
    if Party 2 finds the unique \(x \in \mathcal{T}_{\mathrm{P}_{X \mid Y}}\) with hash value \(f(x)=\Pi_{\text {sim }, 1}\)
    then
        set \(\hat{X}=x\)
    else
        protocol declares an error
    end
```

The following result is from [50], [32, Lemma 7.2.1] (see, also, [208]).

Lemma A. 7 (Performance of protocol 10). For every $\gamma>0$, the protocol above satisfies

$$
\operatorname{Pr}(X \neq \hat{X}) \leq \mathrm{P}_{X Y}\left(\mathcal{T}_{\mathrm{P}_{X \mid Y}}^{c}\right)+2^{-\gamma}
$$

Essentially, the result above says that Party 1 can send $X$ to Party 2 with probability of error less than $\varepsilon$ using roughly as many bits as the $\varepsilon$-tail of $h_{\mathrm{P}_{X \mid Y}}(X \mid Y)$.

In fact, the use of the typical set in Equation (A.16) is not crucial in Protocol 10 and its performance analysis: For a given measure $\mathrm{Q}_{X Y}$, we can define another typical set $\mathcal{T}_{\mathrm{Q}_{X \mid Y}}$ by replacing $h_{\mathrm{P}_{X \mid Y}}(x \mid y)$ with $h_{\mathrm{Q}_{X \mid Y}}(x \mid y)$ in Equation (A.16) even though the underlying distribution of $(X, Y)$ is $\mathrm{P}_{X Y}$. Then, the error probability is bounded as

$$
\operatorname{Pr}(X \neq \hat{X}) \leq \mathrm{P}_{X Y}\left(\mathcal{T}_{Q_{X \mid Y}}^{c}\right)+2^{-\gamma}
$$

which implies that $X$ can be sent by using roughly as many bits as the $\varepsilon$-tail of $h_{\mathrm{Q}_{X \mid Y}}(X \mid Y)$ under $\mathrm{P}_{X Y}$. This modification is crucial for simplifying our performance analysis of the more involved protocols in the following sections.

## A.3.2 Sending $X$ Using Interactive Communication

Protocol 10 aims at minimizing the worst-case communication length over all realization of $(X, Y)$. However, our goal here is to simulate a multiround interactive protocol and as such, we need not account for the worst-case communication length in each round. Instead, we shall optimize the worst-case communication length for the combined interactive protocol. The protocol below is a modification of Protocol 10 and uses roughly $h(X \mid Y)$ bits for transmitting $X$ instead of its $\varepsilon$-tail.

The new protocol proceeds as the previous one but relies on spectrumslicing to adapt the length of communication to the specific realization of $(X, Y)$ : It increases the size of the hash output gradually, starting with $\lambda_{1}=\lambda_{\text {min }}$ and increasing the size $\Delta$-bits at a time until either Party 2 decodes $X$ or $\lambda_{\max }$ bits have been sent. After each transmission, Party 2 sends either an ACK-NACK feedback signal. The protocol stops when an ACK symbol is received.

Fix an auxiliary distribution $\mathrm{Q}_{X Y}$. For $\lambda_{Q_{X \mid Y}}^{\min }, \lambda_{Q_{X \mid Y}}^{\max }, \Delta_{Q_{X \mid Y}}>0$ with $\lambda_{Q_{X \mid Y}}^{\max }>\lambda_{Q_{X \mid Y}}^{\min }$, let

$$
N_{\mathrm{Q}_{X \mid Y}}=\frac{\lambda_{\mathrm{Q}_{X \mid Y}}^{\max }-\lambda_{\mathrm{Q}_{X \mid Y}}^{\min }}{\Delta_{\mathrm{Q}_{X \mid Y}}}
$$

and

$$
\lambda_{Q_{X \mid Y}}^{(i)}=\lambda_{Q_{X \mid Y}}^{\min }+(i-1) \Delta_{Q_{X \mid Y}}, \quad 1 \leq i \leq N_{Q_{X \mid Y}}
$$

Further, let

$$
\begin{equation*}
\mathcal{T}_{Q_{X \mid Y}}^{(0)}:=\left\{(x, y) \mid h_{Q_{X \mid Y}}(x \mid y) \geq \lambda_{Q_{X \mid Y}}^{\max } \text { or } h_{Q_{X \mid Y}}(x \mid y)<\lambda_{Q_{X \mid Y}}^{\min }\right\} \tag{A.17}
\end{equation*}
$$

and for $1 \leq i \leq N_{\mathrm{Q}_{X \mid Y}}$, let $\mathcal{T}_{\mathrm{Q}_{X \mid Y}}^{(i)}$ denote the $i$ th slice of the spectrum given by

$$
\mathcal{T}_{Q_{X \mid Y}}^{(i)}=\left\{(x, y) \mid \lambda_{Q_{X \mid Y}}^{(i)} \leq h_{Q_{X \mid Y}}(x \mid y)<\lambda_{Q_{X \mid Y}}^{(i)}+\Delta_{Q_{X \mid Y}}\right\} .
$$

Note that $\mathcal{T}_{Q_{X \mid Y}}^{(0)}$ corresponds to $\mathcal{T}_{Q_{X \mid Y}}^{c}$ in the previous section and will be counted as an error event.

Our protocol is described in Protocol 11. For every $(x, y) \in \mathcal{T}_{\mathrm{Q}_{X \mid Y}}^{(i)}, 1 \leq$ $i \leq N_{\mathrm{Q}_{X \mid Y}}$, the following lemma provides a bound on the error.

Lemma A. 8 (Performance of protocol 11). For $(x, y) \in \mathcal{T}_{Q_{X \mid Y}}^{(i)}, 1 \leq$ $i \leq N_{Q_{X \mid Y}}$, denoting by $\hat{X}=\hat{X}(x, y)$ the estimate of $x$ at Party 2 at the end of the protocol (with the convention that $\hat{X}=\emptyset$ if an error is declared), Protocol 11 sends at most $\left(l+(i-1) \Delta_{\mathrm{Q}_{X \mid Y}}+i\right)$ bits and has probability of error bounded above as follows:

$$
\operatorname{Pr}(\hat{X} \neq x \mid X=x, Y=y) \leq i 2^{\lambda_{Q_{X \mid Y}}^{\min }+\Delta_{Q_{X \mid Y}}-l}
$$

Proof. Since $(x, y) \in \mathcal{T}_{Q_{X \mid Y}}^{(i)}$, an error occurs if there exists a $\hat{x} \neq x$ such that $(\hat{x}, y) \in \mathcal{T}_{Q_{X \mid Y}}^{(j)}$ and $\Pi_{\text {sim }, 2 k-1}=f_{2 k-1}(\hat{x})$ for $1 \leq k \leq j$ for some $j \leq i$. Therefore, the probability of error is bounded above as
$\operatorname{Pr}(\hat{X} \neq x \mid X=x, Y=y) \leq \sum_{j=1}^{i} \sum_{\hat{x} \neq x} \operatorname{Pr}\left(f_{2 k-1}(x)=f_{2 k-1}(\hat{x}), \forall 1 \leq k \leq j\right)$

```
Protocol 11: Interactive Slepian-Wolf compression
    Input: Observations \(X\) and \(Y\) with distribution \(\mathrm{P}_{X Y}\), uniform public
            randomness \(U_{\text {hash }}\), auxiliary distribution \(\mathrm{Q}_{X Y}\), and parameters
                    \(\gamma, \lambda_{Q_{X \mid Y}}^{\min }, \Delta_{Q_{X \mid Y}}, N_{Q_{X \mid Y}}\), and \(l\)
    Output: Estimate \(\hat{X}\) of \(X\) at party 2
    Both parties use \(U_{\text {hash }}\) to select \(f_{1}\) from \(\mathcal{F}_{l}(\mathcal{X})\)
    Party 1 sends \(\Pi_{\text {sim }, 1}=f_{1}(X)\)
    if Party 2 finds the unique \(x \in \mathcal{T}_{\mathrm{Q}_{X \mid Y}}^{(1)}\) with hash value \(f_{1}(x)=\Pi_{\text {sim }, 1}\)
    then
        set \(\hat{X}=x\)
        send back \(\Pi_{\text {sim }, 2}=\mathrm{ACK}\)
    else
        send back \(\Pi_{\text {sim }, 2}=\) NACK
    end
    while \(2 \leq i \leq N_{Q_{X \mid Y}}\) and party 2 did not send an \(A C K\) do
        Both parties use \(U_{\text {hash }}\) to select \(f_{i}\) from \(\mathcal{F}_{\Delta_{Q_{X \mid Y}}}(\mathcal{X})\), independent of
        \(f_{1}, \ldots, f_{i-1}\)
        Party 1 sends \(\Pi_{\text {sim }, 2 i-1}=f_{i}(X)\)
        if Party 2 finds the unique \(x \in \mathcal{T}_{Q_{X \mid Y}}^{(i)}\) with hash value
        \(f_{j}(x)=\Pi_{\text {sim }, 2 j-1}, \forall 1 \leq j \leq i\) then
            set \(\hat{X}=x\)
            send back \(\Pi_{\text {sim }, 2 i}=\mathrm{ACK}\)
        else
            if More than one such \(x\) found then
                protocol declares an error
            else
                    send back \(\Pi_{\text {sim }, 2 i}=\) NACK
            end
        end
        Reset \(i \rightarrow i+1\)
    end
    if No \(\hat{X}\) found at party 2 then
        | Protocol declares an error
    end
```

$$
\begin{aligned}
& \mathbb{1}\left((\hat{x}, y) \in \mathcal{T}_{\mathrm{Q}_{X \mid Y}}^{(j)}\right) \\
\leq & \sum_{j=1}^{i} \sum_{\hat{x} \neq x} \frac{1}{2^{l+(j-1) \Delta_{Q_{X \mid Y}}}} \mathbb{1}\left((\hat{x}, y) \in \mathcal{T}_{\mathrm{Q}_{X \mid Y}}^{(j)}\right) \\
= & \sum_{j=1}^{i} \sum_{\hat{x} \neq x} \frac{1}{2^{l+(j-1) \Delta_{Q_{X \mid Y}}}}\left|\left\{\hat{x} \mid(\hat{x}, y) \in \mathcal{T}_{Q_{X \mid Y}}^{(j)}\right\}\right|
\end{aligned}
$$

$$
\leq i 2^{\lambda_{Q_{X \mid Y}}^{\min }+\Delta_{Q_{X \mid Y}}-l}
$$

where the first inequality follows from the union bound, the second inequality follows from the property of 2-universal hash family, and the third inequality follows from the fact that $\left|\left\{\hat{x} \mid(\hat{x}, y) \in \mathcal{T}_{Q_{X \mid Y}}^{(j)}\right\}\right| \leq 2^{\lambda_{Q_{X \mid Y}}^{(j)}+\Delta_{Q_{X \mid Y}}}$. Note that the protocol sends $l$ bits in the first transmission, and $\Delta_{Q_{X \mid Y}}$ bits and 1bit feedback in every subsequence transmission. Therefore, no more than $\left(l+(i-1) \Delta_{\mathbb{Q}_{X \mid Y}}+i\right)$ bits are sent.

Corollary A.1. Protocol 11 with $l=\lambda_{Q_{X \mid Y}}^{\min }+\Delta_{Q_{X \mid Y}}+\gamma$ sends at most $\left(h_{\mathrm{Q}_{X \mid Y}}(X \mid Y)+\Delta_{\mathrm{Q}_{X \mid Y}}+\gamma+N_{\mathrm{Q}_{X \mid Y}}\right)$ bits when the observations are ${ }^{5}(X, Y) \notin$ $\mathcal{T}_{Q_{X \mid Y}}^{(0)}$, and has probability of error less than

$$
\operatorname{Pr}(\hat{X} \neq X) \leq \operatorname{Pr}\left((X, Y) \in \mathcal{T}_{Q_{X \mid Y}}^{(0)}\right)+N_{Q_{X \mid Y}} 2^{-\gamma}
$$

## A.3.3 Simulation of $\Pi_{1}$ Using Interactive Communication

We now proceed to simulating the first round of our given interactive protocol $\pi$. Note that using Protocol 11, we can send $\Pi_{1}$ using roughly $h\left(\Pi_{1} \mid Y\right)$ bits. This protocol uses a public randomness $U_{\text {hash }}$ only to choose hash functions, which is convenient for our probability of error analysis, and can be easily derandomized. We now present a scheme which uses another independent portion of public randomness $U_{\text {sim }}$ to reduce the rate of the communication further. However, the scheme will only allow the parties to simulate $\Pi_{1}$ (rather than recover it with small probability of error) and cannot be derandomized.

Specifically, our next protocol uses $X$ and $U=\left(U_{\text {hash }}, U_{\text {sim }}\right)$ to simulate $\Pi_{1}$ in such a manner that $U_{\text {sim }}$ can be treated, in effect, as a portion of the communication used in Protocol 11. Note that since $U_{\text {sim }}$ is independent of $(X, Y)$, the portion of communication which is equivalent to $U_{\text {sim }}$ must as well be almost independent of $(X, Y)$. Such a portion can be guaranteed by noting that the communication used in Protocol 11 is simply a random hash of $\Pi_{1}$ drawn from a 2-universal family, and therefore, its appropriately small portion can have the desired independence property by the leftover

[^23]hash lemma. In fact, since the Markov condition $\Pi_{1} \rightarrow X \mapsto Y$ holds, it suffices guarantee the independent of $X$ instead of $(X, Y)$.

> Protocol 12: Simulation of $\Pi_{1}$
> Input: Observations $X$ and $Y$ with distribution $\mathrm{P}_{X Y}$, uniform public randomness $U=\left(U_{\text {hash }}, U_{\text {sim }}\right)$, auxiliary distribution $\mathrm{Q}_{\Pi_{1} Y}$, and parameters $\gamma, \lambda_{Q_{\Pi_{1} \mid Y}}^{\min }, \Delta_{Q_{\Pi_{1} \mid Y}}, N_{Q_{\Pi_{1} \mid Y}}$ and $k$

Output: Estimates $\Pi_{1 \mathcal{X}}$ and $\Pi_{1 \mathcal{Y}}$ of $\Pi_{1}$

1. Two parties share $k$ random bits $U_{\text {sim }}$ and an $h$ chosen from $\mathcal{H}_{k}\left(\operatorname{supp}\left(\Pi_{1}\right)\right)$ using $U_{\text {hash }}$
2. Party 1 generates a sample $\Pi_{1 \mathcal{X}}$ using $\mathrm{P}_{\Pi_{1} \mid X f\left(\Pi_{1}\right)}\left(\cdot \mid X, U_{\text {sim }}\right)$
3. Parties use Protocol 11 with auxiliary distribution $\mathrm{Q}_{\Pi_{1} Y}$, and parameters $\gamma, \lambda_{Q_{\Pi_{1} \mid Y}}^{\min }, \Delta_{Q_{\Pi_{1} \mid Y}}, N_{Q_{\Pi_{1} \mid Y}}$, and $l=\lambda_{Q_{\Pi_{1} \mid Y}}^{\min }+\Delta_{Q_{\Pi_{1} \mid Y}}+\gamma$ to send $\Pi_{1 \mathcal{X}}$ to Party 2 by treating $U_{\text {sim }}$ as the first $k$ bits of communication obtained via the hash function $f$

Our simulation protocol is described in Protocol 12. Let the quantities such as $\lambda_{Q_{\Pi_{1} \mid Y}}^{\min }, \Delta_{Q_{\Pi_{1} \mid Y}}$, and $N_{Q_{\Pi_{1} \mid Y}}$ be defined analogously to the corresponding quantities in Section A.3.2 with $\Pi_{1}$ replacing $X$. The following lemma provides a bound on the simulation error for Protocol 12.

Lemma A. 9 (Performance of protocol 12). Protocol 12 sends at most

$$
\left(h_{\mathrm{Q}_{\Pi_{1} \mid Y}}\left(\Pi_{1 \mathcal{X}} \mid Y\right)+\Delta_{\mathrm{Q}_{\Pi_{1} \mid Y}}+N_{\mathrm{Q}_{\Pi_{1} \mid Y}}+\gamma-k\right)_{+}
$$

bits when $\left(\Pi_{1 \mathcal{X}}, Y\right) \notin \mathcal{T}_{Q_{\Pi_{1} \mid Y}}^{(0)}$, and has simulation error

$$
\begin{aligned}
d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{1 \chi} \Pi_{1} \mathcal{X} X}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y}\right) \leq & \operatorname{Pr}\left(\left(\Pi_{1}, Y\right) \in \mathcal{T}_{\mathrm{Q}_{\Pi_{1} \mid Y}}^{(0)}\right)+N_{\mathrm{Q}_{\Pi_{1} \mid Y}} 2^{-\gamma} \\
& +\frac{1}{2} \sqrt{2^{k-H_{\min }\left(\mathrm{P}_{\Pi_{1} X} \mid \mathrm{Q}_{X}\right)}}
\end{aligned}
$$

for any auxiliary distribution $\mathrm{Q}_{X}$ on $\mathcal{X}$.
Proof. Consider the following simple protocol for simulating $\Pi_{1}$ at Party 2:

1. Party 1 generates a sample $\Pi_{1}$ using $\mathrm{P}_{\Pi_{1} \mid X}(\cdot \mid X)$.
2. Both parties use Protocol 11 with auxiliary distribution $\mathrm{Q}_{\Pi_{1} Y}$, and parameters $\gamma, \lambda_{Q_{\Pi_{1} \mid Y}}^{\min }, \Delta_{Q_{\Pi_{1} \mid Y}}, N_{Q_{\Pi_{1} \mid Y}}$, and $l=\lambda_{Q_{\Pi_{1} \mid Y}}^{\min }+\Delta_{Q_{\Pi_{1} \mid Y}}+\gamma$ to generate an estimate $\mathrm{Pr}_{\mathrm{r}}^{\mathrm{t}}{ }_{1}$ of $\Pi_{1}$ at Party 2.

In this protocol, $l_{\text {wst }}=\lambda_{\mathrm{Q}_{\Pi_{1} \mid Y}}^{\min }+N_{\mathrm{Q}_{\Pi_{1} \mid Y}} \Delta_{\mathrm{Q}_{\Pi_{1} \mid Y}}+\gamma$ bits of hash values will be sent for the worst $\left(\Pi_{1}, Y\right)$. We divide these $l_{\text {wst }}$ hash values into two parts, the fist $k$ bits and the last $l_{\text {wst }}-k$ bits; let $f$ and $f^{\prime}$, respectively, denote the hash function producing the first and the second parts. Protocol 12 replaces, in effect, $f$ with shared randomness $U_{\text {sim }}$ for an appropriately chosen value of $k$.

Note that the joint distribution of the random variables involved in the simple protocol above satisfies ${ }^{6}$

$$
\begin{align*}
\mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \hat{\Pi}_{1} X Y}(v, & \left.v^{\prime}, \tau, \hat{\tau}, x, y\right) \\
= & \mathrm{P}_{f\left(\Pi_{1}\right) X}(v, x) \mathrm{P}_{\Pi_{1} \mid X f\left(\Pi_{1}\right)}(\tau \mid x, v) \mathrm{P}_{f^{\prime}\left(\Pi_{1}\right) \mid \Pi_{1}}\left(v^{\prime} \mid \tau\right) \mathrm{P}_{Y \mid X}(y \mid x) \\
& \mathrm{P}_{\hat{\Pi}_{1} \mid f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} X Y}\left(\hat{\tau} \mid v, v^{\prime}, \tau, x, y\right) . \tag{A.18}
\end{align*}
$$

Note that the simple protocol above is deterministic and therefore by Remark 2.1

$$
\begin{array}{r}
d_{\text {var }}\left(\mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \hat{\Pi}_{1} X Y}, \mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \Pi_{1} X Y}\right)=\operatorname{Pr}\left(\Pi_{1} \neq \hat{\Pi}_{1}\right) \\
\leq \operatorname{Pr}\left(\left(\Pi_{1}, Y\right) \in \mathcal{T}_{\mathrm{Q}_{\Pi_{1} \mid Y}}^{(0)}\right)+N_{\mathrm{Q}_{\Pi_{1} \mid Y}} 2^{-\gamma} \tag{A.19}
\end{array}
$$

where the inequality is by Corollary A.1.
On the other hand, the joint distribution of RVs involved in Protocol 12 can be factorized as

$$
\begin{align*}
& \mathrm{P}_{U_{\text {sim }} f^{\prime}\left(\Pi_{1 \mathcal{X}}\right) \Pi_{1 \mathcal{X}} \Pi_{1 y} X Y}\left(u, u^{\prime}, \tau, \hat{\tau}, x, y\right) \\
& =\mathrm{P}_{U_{\text {sim }}}(u) \mathrm{P}_{X}(x) \mathrm{P}_{\Pi_{1} \mid X f\left(\Pi_{1}\right)}(\tau \mid x, u) \mathrm{P}_{f^{\prime}\left(\Pi_{1}\right) \mid \Pi_{1}}\left(u^{\prime} \mid \tau\right) \mathrm{P}_{Y \mid X}(y \mid x) \\
& \quad \mathrm{P}_{\hat{\Pi}_{1} \mid f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} X Y}\left(\hat{\tau} \mid u, u^{\prime}, \tau, x, y\right) . \tag{A.20}
\end{align*}
$$

Therefore, the simulation error for Protocol 12 is bounded as

$$
\begin{aligned}
& d_{\text {var }}\left(\mathrm{P}_{\Pi_{1 \chi} \not \Pi_{1 \mathcal{}} X Y}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y}\right) \\
& \leq d_{\text {var }}\left(\mathrm{P}_{U_{\text {sim }} f^{\prime}\left(\Pi_{1}\right) \Pi_{1 \mathcal{}} \Pi_{1} \mathcal{Y} X}, \mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \Pi_{1} X Y}\right) \\
& \leq d_{\text {var }}\left(\mathrm{P}_{U_{\text {sim }} f^{\prime}\left(\Pi_{1}\right) \Pi_{1 \chi} \not \Pi_{1 \mathcal{Y} X Y}}, \mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \hat{\Pi}_{1} X Y}\right) \\
& \quad+d_{\text {var }}\left(\mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \hat{\Pi}_{1} X Y}, \mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \Pi_{1} X Y}\right)
\end{aligned}
$$

[^24]\[

$$
\begin{aligned}
& =d_{\mathrm{var}}\left(\mathrm{P}_{U_{\text {sim }}} \mathrm{P}_{X}, \mathrm{P}_{f\left(\Pi_{1}\right) X}\right)+d_{\mathrm{var}}\left(\mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \hat{\Pi}_{1} X Y}, \mathrm{P}_{f\left(\Pi_{1}\right) f^{\prime}\left(\Pi_{1}\right) \Pi_{1} \Pi_{1} X Y}\right) \\
& \leq d_{\mathrm{var}}\left(\mathrm{P}_{U_{\text {sim }}} \mathrm{P}_{X}, \mathrm{P}_{f\left(\Pi_{1}\right) X}\right)+\operatorname{Pr}\left(\left(\Pi_{1}, Y\right) \in \mathcal{T}_{\mathrm{Q}_{\Pi_{1} \mid Y}}^{(0)}\right)+N_{\mathrm{Q}_{\Pi_{1} \mid Y}} 2^{-\gamma},
\end{aligned}
$$
\]

where the first inequality is by the monotonicity of $\|\cdot\|$, the second inequality is by the triangular inequality, the equality is by the fact that replacing $\mathrm{P}_{U_{\text {sim }}} \mathrm{P}_{X}$ with $\mathrm{P}_{f\left(\Pi_{1}\right) X}$ is the only difference between the factorizations in Equations (A.20) and (A.18), and the final inequality is by Equation (A.19). The desired bound on simulation error for Protocol 12 follows by using Lemma A. 2 to get

$$
d_{\mathrm{var}}\left(\mathrm{P}_{U_{\mathrm{sim}}} \mathrm{P}_{X}, \mathrm{P}_{f\left(\Pi_{1}\right) X}\right) \leq \frac{1}{2} \sqrt{2^{k-H_{\min }\left(\mathrm{P}_{\Pi_{1} X} \mid \mathrm{Q}_{X}\right)}} .
$$

Since Protocol 12 uses shared randomness $U_{\text {sim }}$ instead of sending $f\left(\Pi_{1}\right)$, it communicates $k$ fewer bits in comparison with the simple protocol above, which completes the proof.

## A.3.4 Improved Simulation of $\Pi_{1}$

In Protocol 12 we were able to reduce the communication by roughly
$H_{\text {min }}\left(\mathrm{P}_{\Pi_{1} X} \mid \mathrm{Q}_{X}\right)$ bits by simulating a $\Pi_{1}$ such that if we use Protocol 11 for sending $\Pi_{1}$ to Party 2 , a portion of the required communication can be treated as shared public randomness. However, this is the least reduction in communication we can obtain in the worst case. In this section, we slice the spectrum of $h_{\mathrm{P}_{\Pi_{1} \mid X}}\left(\Pi_{1} \mid X\right)$ to obtain an instantaneous reduction of roughly $h_{\mathrm{P}_{\Pi_{1} \mid X}}\left(\Pi_{1} \mid X\right)$ bits.

Denote by $J$ a random variable which takes the value $j \in\left\{0,1, \ldots, N_{\mathrm{P}_{\Pi_{1} \mid X}}\right\}$ if $\left(\Pi_{1}, X\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid X}}^{(j)}$. In our modified protocol, Party 1 first samples $J$ and sends it to Party 2. Then, they proceed with Protocol 12 for $\mathrm{P}_{\Pi_{1} X Y \mid J=j}$ by selecting $k$ to be less than $H_{\text {min }}\left(\mathrm{P}_{\Pi_{1} X \mid J=j} \mid \mathrm{Q}_{X}\right)$ for an appropriately chosen $\mathrm{Q}_{X}$. Let $\mathcal{J}_{\mathrm{g}}$ be the set of "good" indices $j>0$ with

$$
\mathrm{P}_{J}(j) \geq \frac{1}{N_{\mathrm{P}_{\Pi_{1} \mid X}}^{2}}
$$

it holds that

$$
\mathrm{P}_{J}\left(\mathcal{J}_{\mathrm{g}}^{c}\right)<\operatorname{Pr}\left(\left(\Pi_{1}, X\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid X}}^{(0)}\right)+\frac{1}{N_{\mathrm{P}_{\Pi_{1} \mid X}}}
$$

Note that for $j \in \mathcal{J}_{\mathrm{g}}$, with $\mathrm{Q}_{X}=\mathrm{P}_{X}$, we have

$$
\begin{aligned}
H_{\min }\left(\mathrm{P}_{\Pi_{1} X \mid J=j} \mid \mathrm{P}_{X}\right) & =\min _{\tau, x}-\log \frac{\mathrm{P}_{\Pi_{1} X \mid J}(\tau, x \mid j)}{\mathrm{P}_{X}(x)} \\
& =\min _{\tau, x}-\log \frac{\mathrm{P}_{\Pi_{1} \mid X}(\tau \mid x)}{\mathrm{P}_{J}(j)} \\
& \geq \lambda_{\mathrm{P}_{\Pi_{1} \mid X}}^{\min }+(j-1) \Delta_{\mathrm{P}_{\Pi_{1} \mid X}}-2 \log N_{\mathrm{P}_{\Pi_{1} \mid X}}
\end{aligned}
$$

```
Protocol 13: Improved simulation of \(\Pi_{1}\)
    Input: Observations \(X\) and \(Y\) with distribution \(\mathrm{P}_{X Y}\), uniform public
            randomness \(U=\left(U_{\text {hash }}, U_{\text {sim }}\right)\), and parameters \(\lambda_{\mathrm{P}_{\Pi_{1} \mid Y}}^{\min }, \Delta_{\mathrm{P}_{\Pi_{1} \mid Y}}\),
            \(N_{\mathrm{P}_{\Pi_{1} \mid Y}}, \lambda_{\mathrm{P}_{\Pi_{1} \mid X}}^{\min }, \Delta_{\mathrm{P}_{\Pi_{1} \mid X}}, N_{\mathrm{P}_{\Pi_{1} \mid X}}\), and \(\gamma\)
    Output: Estimates \(\Pi_{1 \mathcal{X}}\) and \(\Pi_{1 \mathcal{Y}}\) of \(\Pi_{1}\)
    Party 1 generate \(J \sim \mathrm{P}_{J \mid X}(\cdot \mid X)\), and send it to Party 2.
    if \(J=j \in \mathcal{J}_{g}\) then
        Parties use Protocol 12 with auxiliary distribution \(\mathrm{P}_{\Pi_{1} Y}\),
        parameters \(\gamma, \lambda_{\mathrm{P}_{\Pi_{1} \mid Y}}^{\min }, \Delta_{\mathrm{P}_{\Pi_{1} \mid Y}}, N_{\mathrm{P}_{\Pi_{1} \mid Y}}\), and
        \(k=\lambda_{\mathrm{P}_{\Pi_{1} \mid X}}^{\min }+(j-1) \Delta_{\mathrm{P}_{\Pi_{1} \mid X}}-2 \log N_{\mathrm{P}_{\Pi_{1} \mid X}}-2 \gamma+2\) to simulate \(\Pi_{1 \mathcal{X}}\)
        and \(\Pi_{1 \mathcal{Y}}\) for the distribution \(\mathrm{P}_{\Pi_{1} X Y \mid J=j}\)
    else
        protocol declares an error
    end
```

Our modified simulation protocol is described in Protocol 13. The following lemma provides a bound on the simulation error.

Lemma A. 10 (Performance of protocol 13). Protocol 13 sends at most $\left(h_{\mathrm{P}_{\Pi_{1} \mid Y}}\left(\Pi_{1 \mathcal{X}} \mid Y\right)-h_{\mathrm{P}_{\Pi_{1} \mid X}}\left(\Pi_{1 \mathcal{X}} \mid X\right)+N_{\mathrm{P}_{\Pi_{1} \mid Y}}+3 \log N_{\mathrm{P}_{\Pi_{1} \mid X}}+\Delta_{\mathrm{P}_{\Pi_{1} \mid Y}}+\Delta_{\mathrm{P}_{\Pi_{1} \mid X}}\right.$ $+3 \gamma)_{+}$bits when $\left(\Pi_{1 \mathcal{X}}, Y\right) \notin \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid Y}}^{(0)}$, and has simulation error

$$
\begin{aligned}
& d_{\text {var }}\left(\mathrm{P}_{\Pi_{1 X} \Pi_{1} X X Y}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y}\right) \\
& \leq \operatorname{Pr}\left(\left(\Pi_{1}, Y\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid Y}}^{(0)}\right)+\operatorname{Pr}\left(\left(\Pi_{1}, X\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid X}}^{(0)}\right)+\left(N_{\mathrm{P}_{\Pi_{1} \mid Y}}+1\right) 2^{-\gamma} \\
& \quad+\frac{1}{N_{\mathrm{P}_{\Pi_{1} \mid X}}}
\end{aligned}
$$

Proof. First, we have

$$
\begin{aligned}
& d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{1 \mathcal{X}} \Pi_{1 \mathcal{}} X Y}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y}\right) \\
& \leq d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{1 \mathcal{X}} \Pi_{1 \mathcal{V}} X Y J}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y J}\right) \\
& =\sum_{j} \mathrm{P}_{J}(j) d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{1 \mathcal{X}} \Pi_{1 \mathcal{V}} X Y \mid J=j}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y \mid J=j}\right) \\
& \leq \sum_{j \in \mathcal{J}_{g}} \mathrm{P}_{J}(j) d_{\text {var }}\left(\mathrm{P}_{\Pi_{1 \mathcal{X}} \Pi_{1 \mathcal{}} X Y \mid J=j}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y \mid J=j}\right)+\mathrm{P}_{J}\left(\mathcal{J}_{\mathrm{g}}^{c}\right) \\
& \leq \sum_{j \in \mathcal{J}_{g}} \mathrm{P}_{J}(j) d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{1 \mathcal{X}} \Pi_{1 \mathcal{Y}} X Y \mid J=j}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y \mid J=j}\right)+\operatorname{Pr}\left(\left(\Pi_{1}, X\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid X}}^{(0)}\right) \\
& +\frac{1}{N_{\mathrm{P}_{\Pi_{1} \mid X}}} .
\end{aligned}
$$

Then, we apply Lemma A. 9 with $\mathrm{Q}_{X}=\mathrm{P}_{X}$ for each $j \in \mathcal{J}_{\mathrm{g}}$, and get

$$
\begin{align*}
& d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{1 \mathcal{X}} \Pi_{1 y} X Y \mid J=j}, \mathrm{P}_{\Pi_{1} \Pi_{1} X Y \mid J=j}\right) \\
& \leq \operatorname{Pr}\left(\left(\Pi_{1}, Y\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid Y}}^{(0)} \mid J=j\right)+N_{\mathrm{P}_{\Pi_{1} \mid Y}} 2^{-\gamma}+\frac{1}{2} \sqrt{2^{k-H_{\min }\left(\mathrm{P}_{\Pi_{1} X \mid J=j} \mid \mathrm{P} X\right)}} \\
& \leq \operatorname{Pr}\left(\left(\Pi_{1}, Y\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{1} \mid Y}}^{(0)} \mid J=j\right)+\left(N_{\mathrm{P}_{\Pi_{1} \mid Y}}+1\right) 2^{-\gamma} \tag{A.21}
\end{align*}
$$

Thus, we have the desired bound on simulation error.
Next, we prove the claimed bound on the number of bits sent by the protocol. By Lemma A.9, the fact that $J$ can be sent by using at most $\log N_{\mathrm{P}_{\Pi_{1} \mid X}}+1$ bits and the choice of $k$ in Protocol 13 , for $J=j$ the protocol above communicates at most

$$
\begin{aligned}
& h_{\mathrm{Q}_{\Pi_{1} \mid Y}}\left(\Pi_{1 \mathcal{X}} \mid Y\right)+\Delta_{\mathrm{Q}_{\Pi_{1} \mid Y}}+N_{\mathrm{Q}_{\Pi_{1} \mid Y}}+\gamma+\log N_{\mathrm{P}_{\Pi_{1} \mid X}}+2-k \\
& \leq h_{\mathrm{Q}_{\Pi_{1} \mid Y}}\left(\Pi_{1 \mathcal{X}} \mid Y\right)-\lambda_{\mathrm{P}_{\Pi_{1} \mid X}}^{\min } \\
& \quad+(j-1) \Delta_{\mathrm{P}_{\Pi_{1} \mid X}}+\Delta_{\mathrm{Q}_{\Pi_{1} \mid Y}}+N_{\mathrm{Q}_{\Pi_{1} \mid Y}} \\
& \quad+3 \log N_{\mathrm{P}_{\Pi_{1} \mid X}}+3 \gamma \\
& \leq h_{\mathrm{Q}_{\Pi_{1} \mid Y}}\left(\Pi_{1 \mathcal{X}} \mid Y\right)-h_{\mathrm{P}_{\Pi_{1} \mid X}}\left(\Pi_{1 \mathcal{X}} \mid X\right)+\Delta_{\mathrm{P}_{\Pi_{1} \mid X}}+\Delta_{\mathrm{Q}_{\Pi_{1} \mid Y}}+N_{\mathrm{Q}_{\Pi_{1} \mid Y}} \\
& \quad+3 \log N_{\mathrm{P}_{\Pi_{1} \mid X}}+3 \gamma,
\end{aligned}
$$

where the previous inequality holds since for $\Pi_{1 \mathcal{X}}$ generated by $\mathrm{P}_{\Pi_{1} \mid X f\left(\Pi_{1}\right) J}\left(\cdot \mid X, U_{\text {sim }}, j\right)$,

$$
\lambda_{\mathrm{P}_{\Pi_{1} \mid X}}^{\min }+j \Delta_{\mathrm{P}_{\Pi_{1} \mid X}} \geq h_{\mathrm{P}_{\Pi_{1} \mid X}}\left(\Pi_{1 \mathcal{X}} \mid X\right)
$$

for each $j \in \mathcal{J}_{\mathrm{g}}$.

## A.3.5 Simulation of $\Pi$

We are now in a position to describe our complete simulation protocol. Consider an interactive protocol $\pi$ with maximum number of rounds $r_{m} a x=d<$ $\infty$. We simply apply Protocol 13 for each round $\Pi_{t}$ of $\Pi$. Our overall simulation protocol is described in Protocol 14. In each round we use Protocol 13 assuming that the simulation up to the previous round has succeeded, where, for the rounds with even numbers, we use Protocol 13 by interchanging the role of Party 1 and Party 2.

```
Protocol 14: Simulation of \(\Pi\)
    Input: Observations \(X\) and \(Y\) with distribution \(\mathrm{P}_{X Y}\), uniform public
            randomness \(U=\left(U_{t, \text { hash }}, U_{t, \text { sim }}: t=1, \ldots, d\right)\), and parameters
            \(\lambda_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}^{\min }, \Delta_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}, N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}, \lambda_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}^{\min }, \Delta_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}\),
            \(N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}\) for \(t=1, \ldots, d\) and \(\gamma\).
    Output: Estimates \(\Pi_{\mathcal{X}}\) and \(\Pi_{\mathcal{Y}}\) of \(\Pi\)
    while Total communication is less than \(l_{\max }\) bits, and simulation not
    ended do
        Party 1 and Party 2, respectively, use estimates \(\Pi_{\mathcal{X}}^{t-1}\) and \(\Pi_{\mathcal{Y}}^{t-1}\) for
        \(\Pi^{t-1}\);
        Parties use Protocol 13 for simulating \(\mathrm{P}_{\Pi_{t}\left(X \Pi^{t-1}\right)\left(Y \Pi^{t-1}\right)}\) with
        parameters \(\lambda_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}^{\min }, \Delta_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}, N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}, \lambda_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}^{\min }\),
        \(\Delta_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}, N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}\) and \(\gamma ;\)
        Update \(t \rightarrow t+1\)
    end
    if Total communication exceeds \(l_{\max }\) bits then
        | Declare an error
    end
```

The following lemma provides a bound on the simulation error.
Lemma A. 11 (Performance of protocol 14). Protocol 14 sends at most $l_{\max }$ bits, and has simulation error $d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} X Y}, \mathrm{P}_{\Pi \Pi X Y}\right) \leq$

$$
\begin{aligned}
& \operatorname{Pr}\left(\mathrm{ic}(\Pi ; X, Y)+\sum_{t=1}^{d} \delta_{t}>l_{\max }\right) \\
& +\sum_{t=1}^{d}\left[4 \operatorname{Pr}\left(\left(\Pi_{t},\left(Y, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}^{(0)}\right)+4 \operatorname{Pr}\left(\left(\Pi_{t},\left(X, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{\Pi}_{\Pi_{t} \mid X \Pi^{t-1}}}^{(0)}\right)\right.
\end{aligned}
$$

$$
\left.+3\left(N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+2\right) 2^{-\gamma}+\frac{3}{N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}}+\frac{3}{N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}}\right]
$$

where
$\delta_{t}=\left\{\begin{array}{l}N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+3 \log N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+\Delta_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+\Delta_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+3 \gamma \\ N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+3 \log N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+\Delta_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+\Delta_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+3 \gamma \\ \text { even } t\end{array}\right.$.

Remark 1. The fudge parameters $\varepsilon^{\prime}$ and $\lambda^{\prime}$ can be explicitly given by

$$
\begin{aligned}
\varepsilon^{\prime}= & \sum_{t=1}^{d}\left[4 \operatorname{Pr}\left(\left(\Pi_{t},\left(Y, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}^{(0)}\right)\right. \\
& +4 \operatorname{Pr}\left(\left(\Pi_{t},\left(X, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}^{(0)}}^{(0)}\right) \\
& \left.+3\left(N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+2\right) 2^{-\gamma}+\frac{3}{N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}}+\frac{3}{N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}}\right] \\
\lambda^{\prime}= & \sum_{t=1}^{d} \delta_{t}
\end{aligned}
$$

Proof. Let us consider a virtual protocol in which the protocol does not terminate even if the total number of bits exceed $l_{\text {max }}$; let us denote the output of this protocol by $\bar{\Pi}_{X}=\left(\bar{\Pi}_{1 \mathcal{X}}, \ldots, \bar{\Pi}_{d \mathcal{X}}\right)$ and $\bar{\Pi}_{Y}=\left(\bar{\Pi}_{1 \mathcal{Y}}, \ldots, \bar{\Pi}_{d \mathcal{Y}}\right)$. Then, we have

$$
\begin{align*}
& d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y} X Y}}, \mathrm{P}_{\Pi \Pi X Y}\right) \\
& \leq d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{\mathcal{X}} \Pi_{\mathcal{Y}} X Y}, \mathrm{P}_{\bar{\Pi}_{\mathcal{X}} \bar{\Pi}_{\mathcal{Y} X Y}}\right)+d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{\mathcal{X}} \bar{\Pi}_{\mathcal{Y} X} X}, \mathrm{P}_{\Pi \Pi X Y}\right) \\
& \leq \operatorname{Pr}\left(\left(\Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right) \neq\left(\bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right)\right)+d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{\mathcal{X}} \bar{\Pi}_{\mathcal{Y} X Y}}, \mathrm{P}_{\Pi \Pi X Y}\right) . \tag{A.23}
\end{align*}
$$

First, we bound the second term of Equation (A.23). By using triangular inequality repeatedly and by using Lemma A.10, we have

$$
\begin{aligned}
& d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{\mathcal{X}} \bar{\Pi}_{\mathcal{Y} X Y}}, \mathrm{P}_{\Pi \Pi X Y}\right) \\
& \leq d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{1 X} \bar{\Pi}_{1 y} \cdots \cdots \bar{\Pi}_{(d-1) \mathcal{X}} \bar{\Pi}_{(d-1) \mathcal{H}} \bar{\Pi}_{d X} \bar{\Pi}_{d y} X Y}, \mathrm{P}_{\Pi_{1} \Pi_{1} \cdots \Pi_{(d-1)} \Pi_{(d-1)} \bar{\Pi}_{d X} \bar{\Pi}_{d y} X Y}\right) \\
& +d_{\text {var }}\left(\mathrm{P}_{\Pi_{1} \Pi_{1} \cdots \Pi_{(d-1)} \Pi_{(d-1)} \bar{\Pi}_{d X} \bar{\Pi}_{d y} X Y}, \mathrm{P}_{\Pi_{1} \Pi_{1} \cdots \Pi_{(d-1)} \Pi_{(d-1)} \Pi_{d} \Pi_{d} X Y}\right) \\
& =d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{1 \mathcal{X}} \bar{\Pi}_{1 \mathcal{Y}} \cdots \bar{\Pi}_{(d-1) \mathcal{X}} \bar{\Pi}_{(d-1) \mathcal{Y}} X Y}, \mathrm{P}_{\Pi_{1} \Pi_{1} \cdots \Pi_{(d-1)} \Pi_{(d-1)} X Y}\right)
\end{aligned}
$$

$$
\begin{align*}
& +d_{\text {var }}\left(\mathrm{P}_{\bar{\Pi}_{d X}} \bar{\Pi}_{d y}\left(X \Pi^{d-1}\right)\left(Y \Pi^{d-1}\right)\right. \\
= & \left.\mathrm{P}_{\Pi_{d} \Pi_{d}\left(X \Pi^{d-1}\right)\left(Y \Pi^{d-1}\right)}\right) \\
\vdots & \\
= & \sum_{t=1}^{d} d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{t \chi} \bar{\Pi}_{t y}\left(X \Pi^{t-1}\right)\left(Y \Pi^{t-1}\right)}, \mathrm{P}_{\Pi_{t} \Pi_{t}\left(X \Pi^{t-1}\right)\left(Y \Pi^{t-1}\right)}\right) \\
\leq & \sum_{t: \text { odd }}\left[\operatorname{Pr}\left(\left(\Pi_{t},\left(Y, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}^{(0)}\right)\right. \\
& \left.+\operatorname{Pr}\left(\left(\Pi_{t},\left(X, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}^{(0)}\right)\right] \\
& \left.+\left(N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+1\right) 2^{-\gamma}+\frac{1}{N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}}\right] \\
& +\sum_{t: \mathrm{even}}\left[\operatorname{Pr}\left(\left(\Pi_{t},\left(Y, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}\right)\right. \\
& \left.+\operatorname{Pr}\left(\left(\Pi_{t},\left(X, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}^{(0)}\right)\right] \\
& \left.+\left(N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+1\right) 2^{-\gamma}+\frac{1}{N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}}\right] \\
\leq & \sum_{t=1}^{d}\left[\operatorname{Pr}\left(\left(\Pi_{t},\left(Y, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}^{(0)}\right)+\operatorname{Pr}\left(\left(\Pi_{t},\left(X, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}^{(0)}\right)\right.  \tag{A.24}\\
& \left.+\left(N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}+N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}+2\right) 2^{-\gamma}+\frac{1}{N_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}}+\frac{1}{N_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}}\right] .(\mathrm{A} .24)
\end{align*}
$$

By denoting

$$
\begin{aligned}
l\left(X, Y, \bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right):= & \sum_{t: \text { odd }} h_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}\left(\bar{\Pi}_{t \mathcal{X}} \mid Y, \bar{\Pi}_{\mathcal{Y}}^{t-1}\right)-h_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}\left(\bar{\Pi}_{t \mathcal{X}} \mid X, \bar{\Pi}_{\mathcal{X}}^{t-1}\right) \\
& +\sum_{t: \text { even }} h_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}\left(\bar{\Pi}_{t \mathcal{Y}} \mid X, \bar{\Pi}_{\mathcal{X}}^{t-1}\right)-h_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}}\left(\bar{\Pi}_{t \mathcal{Y}} \mid Y, \bar{\Pi}_{\mathcal{Y}}^{t-1}\right) .
\end{aligned}
$$

Since $\left(\Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right)$ coincides with $\left(\bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right)$ when the accumulated message length of the protocol generating $\left(\bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right)$ does not exceed $l_{\text {max }}$, and since the message length of each round is bounded by each term of $l\left(X, Y, \bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right)$ plus $\delta_{t}$ by Lemma A. 10 unless $\left(\bar{\Pi}_{t \mathcal{X}},\left(Y, \bar{\Pi}_{\mathcal{Y}}^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}$ or $\left(\bar{\Pi}_{t \mathcal{Y}},\left(X, \bar{\Pi}_{\mathcal{X}}^{t-1}\right)\right) \in$ $\mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}}^{(0)}$, we have

$$
\operatorname{Pr}\left(\left(\Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right) \neq\left(\bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right)\right)
$$

$$
\begin{align*}
\leq & \operatorname{Pr}\left(l\left(X, Y, \bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right)+\sum_{t=1}^{d} \delta_{t}>l_{\max }\right) \\
& +\operatorname{Pr}\left(\bigcup_{t: o \mathrm{odd}}\left(\bar{\Pi}_{t \mathcal{X}},\left(Y, \bar{\Pi}_{\mathcal{Y}}^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}^{(0)}\right. \text { or } \\
& \left.\bigcup_{t: \text { even }}\left(\bar{\Pi}_{t \mathcal{Y}},\left(X, \bar{\Pi}_{\mathcal{X}}^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{\Pi}_{\Pi_{t} \mid X \Pi^{t-1}}^{(0)}}\right) \tag{A.25}
\end{align*}
$$

Since

$$
\operatorname{Pr}\left(\left(X, Y, \bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right) \in \mathcal{E}\right) \leq \operatorname{Pr}((X, Y, \Pi, \Pi) \in \mathcal{E})+d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{\mathcal{X}} \bar{\Pi}_{\mathcal{Y}} X Y}, \mathrm{P}_{\Pi \Pi X Y}\right)
$$

for any event $\mathcal{E}$, it follows from Equation (A.25) that

$$
\begin{align*}
\operatorname{Pr} & \left(\left(\Pi_{\mathcal{X}}, \Pi_{\mathcal{Y}}\right) \neq\left(\bar{\Pi}_{\mathcal{X}}, \bar{\Pi}_{\mathcal{Y}}\right)\right) \\
\leq & \operatorname{Pr}\left(l(X, Y, \Pi, \Pi)+\sum_{t=1}^{d} \delta_{t}>l_{\max }\right) \\
& +\operatorname{Pr}\left(\bigcup_{t: \text { odd }}\left(\Pi_{t},\left(Y, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}^{(0)} \bigcup_{t: \text { even }}\left(\Pi_{t},\left(X, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}^{(0)}}^{(0)}\right) \\
& +2 d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{\mathcal{X}} \bar{\Pi}_{\mathcal{Y}} X Y}, \mathrm{P}_{\Pi \Pi X Y}\right) \\
\leq & \operatorname{Pr}\left(l(X, Y, \Pi, \Pi)+\sum_{t=1}^{d} \delta_{t}>l_{\max }\right) \\
& +\sum_{t=1}^{d}\left[\operatorname{Pr}\left(\left(\Pi_{t},\left(Y, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid Y \Pi^{t-1}}^{(0)}}^{(0)}\right)\right. \\
& \left.+\operatorname{Pr}\left(\left(\Pi_{t},\left(X, \Pi^{t-1}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t} \mid X \Pi^{t-1}}^{(0)}}^{(0)}\right)\right] \\
& +2 d_{\mathrm{var}}\left(\mathrm{P}_{\bar{\Pi}_{\mathcal{X}}} \bar{\Pi}_{\mathcal{Y} X Y}, \mathrm{P}_{\Pi \Pi X Y}\right) . \tag{A.26}
\end{align*}
$$

Thus, by combining this bound with Equations (A.23) and (A.24), and by noting

$$
l(X, Y, \Pi, \Pi)=\mathrm{ic}(\Pi ; X, Y)
$$

we have the desired bound on simulation error.

## A. 4 Section 2.2: Asymptotic Optimality

We now present the proofs of Theorem 2.3 and Theorem 2.4 using singleshot bounds given in Theorem 2.1 and Theorem 2.2. Both the proofs rely on carefully choosing the slice-sizes in the lower and upper bounds.

## A.4.1 Proof of Theorem 2.3

We start with the upper bound. Note that, for IID random variables $\left(\Pi^{n}, X^{n}, Y^{n}\right)$, the spectrums of $h\left(\Pi_{t}^{n} \mid Z^{n},\left(\Pi^{t-1}\right)^{n}\right)$ for ${ }^{7} Z=X$ or $Y$ have width $O(\sqrt{n})$. Therefore, the parameters $\Delta \mathrm{s}$ and $N \mathrm{~s}$ that appear in the fudge parameters can be chosen as $O\left(n^{1 / 4}\right)$. Specifically, by standard measure concentration bounds (for bounded random variables), for every $\nu>0$, there exists a constant ${ }^{8} c>0$ such that with

$$
\begin{aligned}
& \lambda_{\mathrm{P}_{\Pi_{t}^{n} \mid Z^{n}\left(\Pi^{t-1}\right)^{n}}^{\min }}=n H\left(\Pi_{t} \mid Z, \Pi^{t-1}\right)-c \sqrt{n}, \\
& \lambda_{\mathrm{P}_{\Pi_{t}^{n} \mid Z^{n}\left(\Pi^{t-1}\right)^{n}}^{\max }}=n H\left(\Pi_{t} \mid Z, \Pi^{t-1}\right)+c \sqrt{n},
\end{aligned}
$$

the following bound holds:

$$
\begin{equation*}
\operatorname{Pr}\left(\left(\Pi_{t}^{n},\left(Z^{n},\left(\Pi^{t-1}\right)^{n}\right)\right) \in \mathcal{T}_{\mathrm{P}_{\Pi_{t}^{n} \mid Z^{n}\left(\Pi^{t-1}\right)^{n}}^{(0)}}^{(0)} \leq \nu\right. \tag{A.27}
\end{equation*}
$$

Let $T$ denote the third central moment of the random variable ic $(\Pi ; X, Y)$. For

$$
\lambda_{n}=n \mathrm{IC}(\pi)+\sqrt{n \mathrm{~V}(\pi)} Q^{-1}\left(\varepsilon-9 d \nu-\frac{T^{3}}{2 \mathrm{~V}(\pi)^{3 / 2} \sqrt{n}}\right)
$$

choosing $\Delta_{\mathrm{P}_{\Pi_{t}^{n} \mid Z^{n}\left(\Pi^{t-1}\right)^{n}}}=N_{\mathrm{P}_{\Pi_{t}^{n} \mid Z^{n}\left(\Pi^{t-1}\right)^{n}}}=\gamma=\sqrt{2 c} n^{1 / 4}$, and $l_{\max }=\lambda_{n}+$ $\sum_{t=1}^{d} \delta_{t}$ in Theorem 2.2, we get a protocol of length $l_{\max }$ and satisfying

$$
\begin{aligned}
& d_{\mathrm{var}}\left(\mathrm{P}_{\Pi_{\mathcal{X}}^{n} \Pi_{\mathcal{Y}}^{n} X^{n} Y^{n}}, \mathrm{P}_{\Pi^{n} \Pi^{n} X^{n} Y^{n}}\right) \\
& \leq \operatorname{Pr}\left(\sum_{i=1}^{n} \mathrm{ic}\left(\Pi_{i} ; X_{i}, Y_{i}\right)>\lambda_{n}\right)+9 d \nu
\end{aligned}
$$

[^25]for sufficiently large $n$. By its definition given in Equation (A.22), $\delta_{t}=$ $O\left(n^{1 / 4}\right)$ for the choice of parameters above. Thus, the Berry-Esséen theorem (cf. [209]) and the observation above gives a protocol of length $l_{\text {max }}$ attaining $\varepsilon$-simulation. Therefore, using the Taylor approximation of $Q(\cdot)$ yields the achievability of the claimed protocol length.

For the lower bound, we fix sufficiently small constant $\delta>0$, and we set

$$
\begin{aligned}
& \lambda_{\min }^{(1)}=n(H(X, Y)-\delta), \quad \lambda_{\max }^{(1)}=n(H(X, Y)+\delta), \\
& \lambda_{\min }^{(2)}=n(H(X \mid Y, \Pi)-\delta), \quad \lambda_{\max }^{(2)}=n(H(X \mid Y, \Pi)+\delta), \\
& \lambda_{\min }^{(3)}=n(H(X \Pi \triangle Y \Pi)-\delta), \quad \lambda_{\max }^{(3)}=n(H(X \Pi \triangle Y \Pi)+\delta) .
\end{aligned}
$$

Then, by standard measure concentration bounds imply that the tail probability $\varepsilon_{\text {tail }}$ in Equation (2.3) is bounded above by $\frac{c}{n}$ for some constant $c>0$. We also set $\eta=\frac{1}{n}$. For these choices of parameters, we note that the fudge parameter is $\lambda^{\prime}=O(\log n)$. Thus, by setting

$$
\begin{aligned}
\lambda & =\lambda_{n} \\
& =n \mathrm{IC}(\pi)+\sqrt{n \mathrm{~V}(\pi)} Q^{-1}\left(\varepsilon+\frac{c+2}{n}+\frac{T^{3}}{2 \mathrm{~V}(\pi)^{3 / 2} \sqrt{n}}\right) \\
& =n \mathrm{IC}(\pi)+\sqrt{n \mathrm{~V}(\pi)} Q^{-1}(\varepsilon)+O(\log n),
\end{aligned}
$$

where the final equality is by the Tailor approximation, an application of the Berry-Esséen theorem to the bound in Equation (2.2) gives the desired lower bound on the protocol length.

## A.4.2 Proof of Theorem 2.4

Theorem 2.1 implies that if a protocol $\pi_{\text {sim }}$ is such that

$$
\begin{equation*}
\log \left\|\pi_{\text {sim }}\right\|<\lambda-\lambda^{\prime} \tag{A.28}
\end{equation*}
$$

then its simulation error must be larger than

$$
\begin{equation*}
\operatorname{Pr}\left(\mathrm{ic}\left(\Pi^{n} ; X^{n}, Y^{n}\right)>\lambda\right)-\varepsilon^{\prime} . \tag{A.29}
\end{equation*}
$$

To compute fudge parameters, we set

$$
\begin{aligned}
& \lambda_{\min }^{(1)}=n(H(X, Y)-\delta), \quad \lambda_{\max }^{(1)}=n(H(X, Y)+\delta), \\
& \lambda_{\min }^{(2)}=n(H(X \mid Y, \Pi)-\delta), \quad \lambda_{\max }^{(2)}=n(H(X \mid Y, \Pi)+\delta), \\
& \lambda_{\min }^{(3)}=n(H(X \Pi \triangle Y \Pi)-\delta), \quad \lambda_{\max }^{(3)}=n(H(X \Pi \triangle Y \Pi)+\delta) .
\end{aligned}
$$

By the Chernoff bound, there exists $E_{1}>0$ such that

$$
\varepsilon_{\mathrm{tail}} \leq 2^{-E_{1} n}
$$

Furthermore, $\Lambda_{i}=O(n)$ for $i=1,2,3$. We set $\eta=2^{-\frac{\delta}{27} n}$. It follows that

$$
\begin{equation*}
\varepsilon^{\prime} \leq 2^{-E_{1} n}+2^{-\frac{\delta}{27} n} \tag{A.30}
\end{equation*}
$$

and

$$
\begin{equation*}
\lambda^{\prime} \leq \frac{\delta}{3} n+O(\log n) \tag{A.31}
\end{equation*}
$$

Finally, upon setting

$$
\begin{equation*}
\lambda=n \operatorname{IC}(\pi)-\frac{\delta}{3}, \tag{А.32}
\end{equation*}
$$

and applying the Chernoff bound once more, we obtain a constant $E_{2}>0$ such that

$$
\begin{equation*}
\operatorname{Pr}\left(\text { ic }\left(\Pi^{n} ; X^{n}, Y^{n}\right)>\lambda\right) \geq 1-2^{-E_{2} n} . \tag{A.33}
\end{equation*}
$$

The result follows upon combining Equations (A.28)-(A.33).

## APPENDIX B

## PROOFS FOR CHAPTER 3

## B. 1 Section 3.2: Direct Routing

## B.1.1 Proof of Theorem 3.1

Proof. We first note that for the throughput of an empty schedule is zero, i.e. $f\left(\})=0\right.$. Also for any $S \subseteq S^{\prime} \in 2^{\mathbb{Z} \times \mathcal{M}}$ we have $\min \left\{\sum_{(\alpha, M) \in S} \alpha M, T\right\} \leq$ $\min \left\{\sum_{(\alpha, M) \in S^{\prime}} \alpha M, T\right\}$ implying $f(S) \leq f\left(S^{\prime}\right)$. Hence $f$ is normalized and monotone. Next, using the identify

$$
\min (a+b, c)=\min (a, c)+\min (b, c-\min (a, c))
$$

for non-negative reals $a, b, c$, we have for $S \in 2^{\mathbb{Z} \times \mathcal{M}}$ and $\left(\alpha_{0}, M_{0}\right) \notin S$,

$$
\begin{align*}
& f\left(S \cup\left\{\left(\alpha_{0}, M_{0}\right)\right\}\right)=\left\|\min \left(\sum_{(\alpha, M) \in S} \alpha M+\alpha_{0} M_{0}, T\right)\right\|_{1} \\
& \quad=\left\|\min \left(\sum_{(\alpha, M) \in S} \alpha M, T\right)+\min \left(\alpha_{0} M_{0}, T-\min \left(\sum_{(\alpha, M) \in S} \alpha M, T\right)\right)\right\|_{1} \\
& \quad \Rightarrow f_{S}\left(\left(\alpha_{0}, M_{0}\right)\right)=\left\|\min \left(\alpha_{0} M_{0}, T-\min \left(\sum_{(\alpha, M) \in S} \alpha M, T\right)\right)\right\|_{1}, \tag{B.1}
\end{align*}
$$

where $f_{S}\left(\left(\alpha_{0}, M_{0}\right)\right)$ denotes the incremental marginal value of adding $\left(\alpha_{0}, M_{0}\right)$ to the set $S$ (see Section 3.2.2). Finally, for $S \subseteq S^{\prime} \in 2^{\mathbb{Z} \times \mathcal{M}}$ and $\left(\alpha_{0}, M_{0}\right) \notin S^{\prime}$ we have

$$
T-\min \left(\sum_{i \in S^{\prime}} \alpha_{i} M_{i}, T\right) \leq T-\min \left(\sum_{i \in S} \alpha_{i} M_{i}, T\right) .
$$

Combining the above equation with Equation (B.1) we get

$$
f_{S^{\prime}}\left(\left\{\left(\alpha_{0}, M_{0}\right)\right\}\right) \leq f_{S}\left(\left\{\left(\alpha_{0}, M_{0}\right)\right\}\right),
$$

or in other words $f$ is submodular.

## B.1.2 Proof of Theorem 3.2

Proof. Recall the submodular sum-throughput function $f$ defined in Equation (3.2). Let $\left\{\left(\alpha_{1}, M_{1}\right), \ldots,\left(\alpha_{k}, M_{k}\right)\right\}$ be the schedule returned by Algorithm 2. Let $S_{i}=\left\{\left(\alpha_{1}, M_{1}\right), \ldots,\left(\alpha_{i}, M_{i}\right)\right\}$ denote the schedule computed at the end of $i$ iterations of the while loop and let $S^{*}$ denote the optimal schedule. Now, since in the $i+1$-th iteration $\left(\alpha_{i+1}, M_{i+1}\right)$ maximizes $\frac{\min \left(\alpha M, T_{\text {rem }}(i+1)\right) \|_{1}}{(\alpha+\delta)}=\frac{f_{S_{i}}(\{(\alpha, M)\})}{(\alpha+\delta)}$ we have for any $(\alpha, M) \notin S_{i}$,

$$
\begin{align*}
& \frac{f_{S_{i}}(\{(\alpha, M)\})}{(\alpha+\delta)} \leq \frac{f_{S_{i}}\left(\left\{\left(\alpha_{i+1}, M_{i+1}\right)\right\}\right)}{\left(\alpha_{i+1}+\delta_{i+1}\right)} \\
\Rightarrow & f_{S_{i}}(\{(\alpha, M)\}) \leq \frac{(\alpha+\delta)}{\left(\alpha_{i+1}+\delta_{i+1}\right)} f_{S_{i}}\left(\left\{\left(\alpha_{i+1}, M_{i+1}\right)\right\}\right) . \tag{B.2}
\end{align*}
$$

Now consider OPT $-f\left(S_{i}\right)$ for some $i<k$. Since $f$ is monotone we have

$$
\begin{align*}
\mathrm{OPT}-f\left(S_{i}\right) & =f\left(S^{*}\right)-f\left(S_{i}\right) \leq f\left(S_{i} \cup S^{*}\right)-f\left(S_{i}\right) \\
& \leq \sum_{(\alpha, M) \in J^{*}} f_{S_{i}}(\{(\alpha, M)\}) \\
& \leq \sum_{(\alpha, M) \in J^{*}} \frac{(\alpha+\delta)}{\left(\alpha_{i+1}+\delta_{i+1}\right)} f_{S_{i}}\left(\left\{\left(\alpha_{i+1}, M_{i+1}\right)\right\}\right)  \tag{B.3}\\
& \leq \frac{W}{\left(\alpha_{i+1}+\delta_{i+1}\right)} f_{S_{i}}\left(\left\{\left(\alpha_{i+1}, M_{i+1}\right)\right\}\right), \tag{B.4}
\end{align*}
$$

where $J^{*}:=S^{*} \backslash S_{i}$ denotes the set of matchings that are present in the optimal solution but not in $S_{i}$, Equation (B.3) follows from Equation (B.2), and Equation (B.4) follows because $\sum_{(\alpha, M) \in J^{*}}(\alpha+\delta) \leq \sum_{(\alpha, M) \in S^{*}}(\alpha+\delta) \leq$ $W$. Next, observe that

$$
\begin{aligned}
f\left(S_{i+1}\right) & =f\left(S_{i}\right)+f_{S_{i}}\left(\left\{\left(\alpha_{i+1}, M_{i+1}\right)\right\}\right) \\
\Rightarrow \mathrm{OPT}-f\left(S_{i+1}\right) & =\mathrm{OPT}-f\left(S_{i}\right)-f_{S_{i}}\left(\left\{\left(\alpha_{i+1}, M_{i+1}\right)\right\}\right)
\end{aligned}
$$

$$
\begin{align*}
& \leq\left(\mathrm{OPT}-f\left(S_{i}\right)\right)\left(1-\frac{\left(\alpha_{i+1}+\delta\right)}{W}\right)  \tag{B.5}\\
& \leq\left(\mathrm{OPT}-f\left(S_{0}\right)\right) \prod_{i^{\prime}=1}^{i+1}\left(1-\frac{\left(\alpha_{i^{\prime}}+\delta\right)}{W}\right) \\
& \leq \mathrm{OPT} \times e^{-\sum_{i^{\prime}=1}^{i+1}\left(\alpha_{i^{\prime}}+\delta\right) / W} \tag{B.6}
\end{align*}
$$

where Equation (B.5) follows from Equation (B.4) and Equation (B.6) follows because of the identity $1-x \leq e^{-x}$. Now, since after the $k$-th iteration the while loop terminates, this implies $\sum_{i^{\prime}=1}^{k}\left(\alpha_{i^{\prime}}+\delta\right)>W$. However, if the entries of the input traffic matrix $T$ are bounded by $\epsilon W+\delta$, then no matching has a duration longer than $\epsilon W$. In particular $\alpha_{k}+\delta \leq \epsilon W \Rightarrow$ $\sum_{i^{\prime}=1}^{k-1}\left(\alpha_{i^{\prime}}+\delta\right) \geq W(1-\epsilon)$. Thus, setting $i=k-2$ in Equation (B.6) we have

$$
\begin{aligned}
& \mathrm{OPT}-f\left(S_{k-1}\right) \leq \mathrm{OPT} \times e^{-\sum_{i^{\prime}=1}^{k-1}\left(\alpha_{i^{\prime}}+\delta\right) / W} \leq \mathrm{OPT} \times e^{-(1-\epsilon)} \\
& \Rightarrow \mathrm{OPT}-\mathrm{ALG} 2 \leq \mathrm{OPT} \times e^{-(1-\epsilon)} .
\end{aligned}
$$

Hence we conclude ALG2 $\geq \operatorname{OPT}\left(1-e^{-(1-\epsilon)}\right)$.

## B.1.3 Correctness

Consider any traffic matrix $T \in \mathbb{Z}^{n \times n}$. Let $\mathcal{T}=\{T(i, j): i, j \leq[n]\}$ denote the distinct entries in the matrix $T$. Then, in the following, we show that the maximizer in

$$
\begin{equation*}
\max _{\alpha \in \mathbb{Z}, M \in \mathcal{M}} \frac{\|\min (T, \alpha M)\|_{1}}{\alpha+\delta} \tag{B.7}
\end{equation*}
$$

occurs for $\alpha \in \mathcal{T}$. To do this, for any matching $M \in \mathcal{M}$ let us define $f_{M}(\alpha) \triangleq\|\min (\alpha M, T)\|_{1}$ and let $f(\alpha) \triangleq \max _{M \in \mathcal{M}} \frac{f_{M}(\alpha)}{\alpha+\delta}$. We then have the following proposition.

Proposition B.1. $f_{M}(\alpha)$ is (i) non-decreasing, (ii) piece-wise linear where the corner points are from $\mathcal{T}$ and (iii) concave.

Proof. It is easy to see (i) because if $\alpha_{1} \leq \alpha_{2}$ then $\min \left(\alpha_{1} M, T\right) \leq \min \left(\alpha_{2} M, T\right)$ entrywise and hence $f_{M}\left(\alpha_{1}\right) \leq f_{M}\left(\alpha_{2}\right)$. To see (ii) consider any $t_{1}<t_{2} \in \mathcal{T}$ such that no other element of $\mathcal{T}$ is between $t_{1}$ and $t_{2}$. Then for $t_{1} \leq \alpha \leq t_{2}$
we have

$$
\begin{align*}
f_{M}(\alpha) & =\|\min (\alpha M, T)\|_{1} \\
& =\sum_{\substack{(i, j) \in M \\
T(i, j) \leq t_{1}}} \min (\alpha, T(i, j))+\sum_{\substack{(i, j) \in M \\
T(i, j) \geq t_{1}}} \min (\alpha, T(i, j)) \\
& =\sum_{\substack{(i, j) \in M \\
T(i, j) \leq t_{1}}} T(i, j)+\sum_{\substack{(i, j) \in M \\
T(i, j) \geq t_{1}}} \alpha \\
& =\sum_{\substack{(i, j) \in M \\
T(i, j) \leq t_{1}}} T(i, j)+\left|\left\{(i, j) \in M: T(i, j) \geq t_{1}\right\}\right| \alpha . \tag{B.8}
\end{align*}
$$

Thus $f_{M}(\cdot)$ is linear for $t_{1} \leq \alpha \leq t_{2}$ and (ii) follows. (iii) also follows from Equation (B.8) by observing that

$$
\left|\left\{(i, j) \in M: T(i, j) \geq t_{1}\right\}\right| \geq\left|\left\{(i, j) \in M: T(i, j) \geq t_{2}\right\}\right|
$$

for any $t_{1}<t_{2} \in \mathcal{T}$. Hence the slope of the piece-wise linear function $f_{M}(\alpha)$ is non-increasing as $\alpha$ increases. In other words, $f_{M}(\alpha)$ is concave.

Next, we consider a case where the matching is fixed.
Proposition B.2. For a fixed matching $M$, we have $\arg \max _{\alpha} \frac{f_{M}(\alpha)}{\alpha+\delta} \in \mathcal{T}$.
Proof. This follows from Proposition B.1-(ii). Let $f_{M}(\alpha)$ be linear for $\alpha \in$ $\left[t_{1}, t_{2}\right]$. Then it can be written as $f_{M}(\alpha)=f_{M}\left(t_{1}\right)+m\left(\alpha-t_{1}\right)$ for some slope $m \geq 0$. Now, consider the derivation of the function $f_{M}(\alpha) /(\alpha+\delta)$ in the interval $\left[t_{1}, t_{2}\right]$ :

$$
\begin{align*}
\frac{d}{d \alpha}\left(\frac{f_{M}(\alpha)}{\alpha+\delta}\right) & =\frac{d}{d \alpha}\left(\frac{f_{M}\left(t_{1}\right)+m\left(\alpha-t_{1}\right)}{\alpha+\delta}\right) \\
& =\frac{(\alpha+\delta)(m)-\left(f_{M}\left(t_{1}\right)+m\left(\alpha-t_{1}\right)\right)}{(\alpha+\delta)^{2}} \\
& =\frac{\delta m-f_{M}\left(t_{1}\right)+m t_{1}}{(\alpha+\delta)^{2}} . \tag{B.9}
\end{align*}
$$

Note that the numerator of Equation (B.9) is independent of $\alpha$ and the denominator is strictly positive. Hence the sign (i.e., $>0,<0$ or $=0$ ) of the slope of $f_{M}(\alpha) /(\alpha+\delta)$ is the same in the interval $\left[t_{1}, t_{2}\right]$. This proves that the maximum must occur at either of the extreme points $t_{1}$ or $t_{2}$. By

Proposition B.1-(ii) we know that the $f_{M}(\alpha)$ is piece-wise linear with the corner points from the set $\mathcal{T}$. Thus we can conclude that the maximum must occur at one of the points in $\mathcal{T}$.

We are now ready to show that the maximizer of Equation (B.7) occurs for $\alpha \in \mathcal{T}$.

Theorem B.1. $\arg \max _{\alpha} f(\alpha) \in \mathcal{T}$.
Proof. This follows directly from Proposition B.2. Notice that

$$
\max _{\alpha} f(\alpha)=\max _{\alpha} \max _{M} \frac{f_{M}(\alpha)}{\alpha+\delta}=\max _{M}\left(\max _{\alpha} \frac{f_{M}(\alpha)}{\alpha+\delta}\right)
$$

But the maximizer of $f_{M}(\alpha) /(\alpha+\delta)$ belongs to $\mathcal{T}$ for any $M$. Hence we conclude that the maximizer of $f(\alpha)$ also belongs to $\mathcal{T}$ and the theorem follows.

## APPENDIX C

## PROOFS FOR CHAPTER 4

## C. 1 Section 4.5.1: Converse

We first prove the following Proposition C.1, which bounds the depth of any tree with a give degree distribution.

Proposition C.1. Any directed tree with $n$ nodes, and where $d^{(i)}$ fraction of the nodes have an out-degree of $i$ for $i=0,1, \ldots, l$, has a depth $D$ that is bounded as

$$
\begin{equation*}
D \geq \frac{d^{(1)}}{d^{(0)}}+\log _{l}\left(1+\sum_{k=2}^{l} n d^{(k)}(k-1)\right)-(l-2) \log _{l}\left(\frac{l!}{2}\right) . \tag{C.1}
\end{equation*}
$$

Proof. It is clear that the tree with the lowest depth, for a given $\left(d^{(0)}, \ldots, d^{(l)}\right)$, has the largest degree nodes on the very top followed by the second largest degree nodes and so on. Let us call a layer of nodes at a particular depth as an $i$-layer if the largest degree node present in that layer has the degree $i$. Further, let $d_{i}, i=1, \ldots, l$ denote the number of of the $i$-layers in the tree. Therefore,

$$
\begin{equation*}
D=\sum_{i=0}^{l} d_{i} \tag{C.2}
\end{equation*}
$$

gives the depth of the tree. The proof proceeds by bounding the depth of each layer. The number of nodes in the topmost layer of the graph, layer $l$, can be bounded as ${ }^{1}$

$$
\begin{equation*}
1+l+\ldots+l^{d_{l}-2} \leq n d^{(l)} \leq 1+l+\ldots+l^{d_{l}-1} . \tag{C.3}
\end{equation*}
$$

[^26]This yields

$$
\begin{align*}
\log _{l}\left(n d^{(l)}(l-1)+1\right) & \leq d_{l}  \tag{C.4}\\
l^{d_{l}} & \leq\left(n d^{(l)}(l-1)+1\right) l . \tag{C.5}
\end{align*}
$$

Now, in the second layer where there are nodes of degree $l-1$ (or possibly lesser), since $l^{d_{l}}$ constitutes an upper bound on the number of degree $l$ parents of degree $l-1$ nodes and $l^{d_{l}-1}(l-1)$ constitutes a lower bound, we must have

$$
\begin{array}{r}
l^{d_{l}-1}(l-1)\left(1+(l-1)+\ldots+(l-1)^{d_{l-1}-2}\right) \leq n d^{(l-1)}, \\
n d^{(l-1)} \leq l^{d_{l}}\left(1+(l-1)+\ldots+(l-1)^{d_{l-1}-1}\right) \\
\Rightarrow l^{d_{l}-1}\left(1+(l-1)+\ldots+(l-1)^{d_{l-1}-2}\right) \leq n d^{(l-1)}, \\
n d^{(l-1)} \leq l^{d_{l}}\left(1+(l-1)+\ldots+(l-1)^{d_{l-1}-1}\right) . \tag{C.7}
\end{array}
$$

This yields

$$
\begin{align*}
& \log _{l-1}\left(\frac{n d^{(l-1)}(l-2)}{l^{d_{l}}}+1\right) \leq d_{l-1}  \tag{C.8}\\
& l^{d_{l}}(l-1)^{d_{l-1}} \leq\left(n d^{(l-1)}(l-2)+n d^{(l)}(l-1)+1\right)(l)(l-1) \tag{C.9}
\end{align*}
$$

Using Equation (C.5) in Equation (C.8) we have,

$$
\begin{equation*}
\log _{l-1}\left(\frac{n d^{(l-1)}(l-2)}{\left(n d^{(l)}(l-1)+1\right) l}+1\right) \leq d_{l-1} \tag{C.10}
\end{equation*}
$$

Similarly, we have in the $(l-2)$ th layer,

$$
\begin{array}{r}
l^{d_{l}-1}(l-1)(l-1)^{d_{l-1}-1}(l-2)\left(1+(l-2)+\ldots+(l-2)^{d_{l-2}-2}\right) \leq n d^{(l-2)} \\
\Rightarrow l^{d_{l}-1}(l-1)^{d_{l-1}-1}\left(1+(l-2)+\ldots+(l-2)^{d_{l-2}-2}\right) \leq n d^{(l-2)} \\
\Rightarrow n d^{(l-2)} \leq l^{d_{l}}(l-1)^{d_{l-1}}\left(1+(l-2)+\ldots+(l-2)^{d_{l-2}-1}\right), \tag{C.11}
\end{array}
$$

yielding

$$
\begin{align*}
\log _{l-2}\left(\frac{n d^{(l-2)}(l-3)}{l^{d_{l}}(l-1)^{d_{l-1}}}+1\right) & \leq d_{l-2}  \tag{C.12}\\
l^{d_{l}}(l-1)^{d_{l-1}}(l-2)^{d_{l-2}} & \leq\left(n d^{(l-2)}(l-3)\right. \\
& +n d^{(l-1)}(l-2)+n d^{(l)}(l-1)
\end{align*}
$$

$$
\begin{equation*}
+1)(l)(l-1)(l-2) \tag{C.13}
\end{equation*}
$$

Using Equation (C.9) we have:

$$
\begin{equation*}
\log _{l-2}\left(\frac{n d^{(l-2)}(l-3)}{\left(n d^{(l-1)}(l-2)+n d^{(l)}(l-1)+1\right)(l)(l-1)}+1\right) \leq d_{l-2} . \tag{C.14}
\end{equation*}
$$

We continue this process for all the $i$-layers for $i \geq 2$. Finally, in the last layer the number of degree one chains is equal to the number of the leaves. As such, we must have

$$
\begin{equation*}
\frac{n d^{(1)}}{n d^{(0)}}-1 \leq d_{1}, \text { and } d_{0}=1 \tag{C.15}
\end{equation*}
$$

Therefore, from Equation (C.2) we have depth

$$
\begin{equation*}
D \geq \frac{d^{(1)}}{d^{(0)}}+\sum_{k=2}^{l} \log _{k}\left(1+\frac{n d^{(k)}(k-1)}{\left(1+\sum_{k^{\prime}=k+1}^{l} n d^{\left(k^{\prime}\right)}\left(k^{\prime}-1\right)\right) \prod_{k^{\prime \prime}=k+1}^{l}\left(k^{\prime \prime}\right)}\right) \tag{C.16}
\end{equation*}
$$

Now, the second term in the right-hand side of Equation (C.16), denoted by $T$, can be lower bounded as

$$
\begin{align*}
T & \geq \sum_{k=2}^{l} \log _{l}\left(1+\frac{n d^{(k)}(k-1)}{\left(1+\sum_{k^{\prime}=k+1}^{l} n d^{\left(k^{\prime}\right)}\left(k^{\prime}-1\right)\right) \prod_{k^{\prime \prime}=k+1}^{l}\left(k^{\prime \prime}\right)}\right) \\
& =\log _{l} \prod_{k=2}^{l}\left(1+\frac{n d^{(k)}(k-1)}{\left(1+\sum_{k^{\prime}=k+1}^{l} n d^{\left(k^{\prime}\right)}\left(k^{\prime}-1\right)\right) \prod_{k^{\prime \prime}=k+1}^{l}\left(k^{\prime \prime}\right)}\right) \\
& \geq \log _{l} \prod_{k=2}^{l}\left(\frac{\left(1+\sum_{k^{\prime}=k+1}^{l} n d^{\left(k^{\prime}\right)}\left(k^{\prime}-1\right)\right)+n d^{(k)}(k-1)}{\left(1+\sum_{k^{\prime}=k+1}^{l} n d^{\left(k^{\prime}\right)}\left(k^{\prime}-1\right)\right) \prod_{k^{\prime \prime}=k+1}^{l}\left(k^{\prime \prime}\right)}\right) \\
& =\log _{l}\left(\left(1+\sum_{k^{\prime}=2}^{l} n d^{\left(k^{\prime}\right)}\left(k^{\prime}-1\right)\right) \prod_{k=2}^{l}\left(\frac{1}{\prod_{k^{\prime \prime}=k+1}^{l}\left(k^{\prime \prime}\right)}\right)\right) \\
& \geq \log _{l}\left(1+\sum_{k^{\prime}=2}^{l} n d^{\left(k^{\prime}\right)}\left(k^{\prime}-1\right)\right)-(l-2) \log _{l}\left(\frac{l!}{2}\right) \tag{C.17}
\end{align*}
$$

thus proving the claim.
We now prove Theorem 4.2.
Proof of Theorem 4.2. As mentioned in Section 4.5.1, without loss of gener-
ality let us consider $T$ trees with the $i$ th tree carrying a rate of $r_{i}$. Let $d_{i}^{(j)}$ denote the fraction of nodes having an out-degree of $j$ in tree $i$. Clearly,

$$
\begin{equation*}
d_{i}^{(0)}+d_{i}^{(1)}+\ldots+d_{i}^{(l)}=1, \quad \forall i=1, \ldots, T \tag{C.18}
\end{equation*}
$$

Since any tree with $n$ nodes has $n-1$ edges, we have

$$
\begin{array}{ll}
n\left(d_{i}^{(1)}+2 d_{i}^{(2)}+\ldots+(l-1) d_{i}^{(l-1)}+l d_{i}^{(l)}\right)=n-1, & \forall i=1, \ldots, T \\
\Rightarrow d_{i}^{(1)}+2 d_{i}^{(2)}+\ldots+(l-1) d_{i}^{(l-1)}+l d_{i}^{(l)}=1-\frac{1}{n}, & \forall i=1, \ldots, T \tag{C.20}
\end{array}
$$

Now, every degree $i$ node for $i \geq 2$ needs at least $i-1$ redundant edges because of the capacity requirement of the theorem. As such, the cumulative node capacity constraint becomes

$$
\begin{align*}
& \sum_{i=1}^{T}(n-1) r_{i}+n\left(d_{i}^{(2)}+2 d_{i}^{(3)}+\ldots+(l-1) d_{i}^{(l)}\right) r_{i} \leq n \\
\Rightarrow & \sum_{i=1}^{T}\left(1-\frac{1}{n}+d_{i}^{(2)}+2 d_{i}^{(3)}+\ldots+(l-1) d_{i}^{(l)}\right) r_{i} \leq 1 . \tag{C.21}
\end{align*}
$$

The proof essentially obtains a lower bound for the expression in Equation (C.1) based on above Equations (C.18), (C.20) and (C.21). Subtracting Equation (C.18) from (C.20) gives

$$
\begin{equation*}
d_{i}^{(2)}+2 d_{i}^{(3)}+\ldots+(l-1) d_{i}^{(l)}=d_{i}^{(0)}-\frac{1}{n}, \quad \forall i=1, \ldots, T \tag{C.22}
\end{equation*}
$$

From the above, we have

$$
\begin{equation*}
d_{i}^{(j)} \leq \frac{1}{j-1}\left(d_{i}^{(0)}-\frac{1}{n}\right) \tag{C.23}
\end{equation*}
$$

and combined with Equation (C.18) we get

$$
\begin{aligned}
1 & \leq d_{i}^{(0)}+d_{i}^{(1)}+\left(d_{i}^{(0)}-\frac{1}{n}\right)\left(1+\frac{1}{2}+\ldots+\frac{1}{l-1}\right) \\
& \leq d_{i}^{(0)}+d_{i}^{(1)}+\left(d_{i}^{(0)}-\frac{1}{n}\right)\left(\log _{e}(l-1)+1\right) \\
\Rightarrow d_{i}^{(1)} & \geq 1-d_{i}^{(0)}\left(\log _{e}(l-1)+2\right)+\frac{1}{n}\left(\log _{e}(l-1)+1\right)
\end{aligned}
$$

$$
\begin{equation*}
\Rightarrow \frac{d_{i}^{(1)}}{d_{i}^{(0)}} \geq \frac{1}{d_{i}^{(0)}}-\left(\log _{e}(l-1)+2\right)+\frac{1}{n d_{i}^{(0)}}\left(\log _{e}(l-1)+1\right) \tag{C.24}
\end{equation*}
$$

Also, the second term in the delay lower bound in Equation (C.1) becomes

$$
\begin{equation*}
\log _{l}\left(1+\sum_{k=2}^{l} n d^{(k)}(k-1)\right)=\log _{l}\left(1+n d_{i}^{(0)}\right) \tag{C.25}
\end{equation*}
$$

As such, using Equations (C.24), (C.25) and (C.1) the delay for the $i$-th tree $D_{i}$ can now be lower bounded as

$$
\begin{equation*}
D_{i} \geq \frac{1}{d_{i}^{(0)}}-\left(\log _{e}(l-1)+2\right)+\log _{l}\left(1+n d_{i}^{(0)}\right)-(l-2) \log _{l}\left(\frac{l!}{2}\right) \tag{C.26}
\end{equation*}
$$

for all $i=1, \ldots, T$. The derivative of the right-hand side above in Equation (C.26) with respect to $d_{i}^{(0)}$ is given by

$$
\begin{equation*}
-\frac{1}{\left(d_{i}^{(0)}\right)^{2}}+\frac{n}{\left(1+n d_{i}^{(0)}\right) \log l}, \tag{C.27}
\end{equation*}
$$

which is strictly negative in $0<d_{i}^{(0)}<1$. As such, the minima in the righthand side of Equation (C.26) is achieved by the largest achievable $d_{i}^{(0)}$. Now, using Equations (C.18) and (C.20) in (C.21) we get

$$
\begin{align*}
& \sum_{i=1}^{T}\left(1-\frac{2}{n}+d_{i}^{(0)}\right) r_{i} \leq 1 \\
& \Rightarrow \min _{i} d_{i}^{(0)} \leq \frac{1}{R}-1+\frac{2}{n} \tag{C.28}
\end{align*}
$$

Letting $i^{*}=\arg \min d_{i}^{(0)}$, the overall delay for the system can be bounded by the delay of the $i^{*}$-th tree. Hence, substituting Equation (C.28) in (C.26) we have

$$
\begin{align*}
D \geq & \frac{1}{d_{i^{*}}^{(0)}}-\left(\log _{e}(l-1)+2\right)+\log _{l}\left(1+n d_{i^{*}}^{(0)}\right)-(l-2) \log _{l}\left(\frac{l!}{2}\right)  \tag{C.29}\\
\geq & \frac{1}{\frac{1}{R}-1+\frac{2}{n}}+\log _{l}\left(1+n\left(\frac{1}{R}-1+\frac{2}{n}\right)\right) \\
& -(l-2) \log _{l}\left(\frac{l!}{2}\right)-\log _{e}(l-1)-2 \tag{C.30}
\end{align*}
$$

$$
\begin{align*}
& \geq \log _{l} n+\frac{R}{2(1-R)}+\log _{l}\left(\frac{2(1-R)}{R}\right)-(l-2) \log _{l}\left(\frac{l!}{2}\right) \\
& \quad-\log _{e}(l-1)-2 \tag{C.31}
\end{align*}
$$

for $n \geq 3 R /(1-R)$. For $l=\Delta$ and a node capacity of $C$ (rather than 1 ) replacing $R$ by $R / C$, we get the desired theorem. Hence we can conclude that the steady state delay in our algorithm, Theorem (4.1), is order optimal for the class of algorithms satisfying the conditions of Theorem 4.2.

## APPENDIX D

## PROOFS FOR CHAPTER 5

## D. 1 Section 5.2: Anonymity Metric Properties

## D.1.1 Proof of Theorem 5.1

Consider any realization of the network, in which the messages $\mathcal{X}$ are mapped to the servers $V_{H}$ according to mapping rule M . Then from the definition of precision and recall at any node $v$ (Equations (5.2), (5.3)), we have

$$
\begin{equation*}
D_{\mathrm{M}}(v)=\frac{\mathbb{1}\left\{\mathrm{M}\left(X_{v}\right)=v\right\}}{\sum_{w \in V_{H}} \mathbb{1}\left\{\mathrm{M}\left(X_{w}\right)=v\right\}} \leq \mathbb{1}\left\{\mathrm{M}\left(X_{v}\right)=v\right\}=R_{\mathrm{M}}(v) . \tag{D.1}
\end{equation*}
$$

Hence it follows that the macro-averaged precision $D_{\mathrm{M}}$ is at most the macroaveraged recall $R_{\mathrm{M}}$, implying $\mathbf{D}_{\mathrm{M}} \leq \mathbf{R}_{\mathrm{M}}$.

To prove inequality (b), let $V_{\mathrm{M}}=\left\{v \in V_{H}: \mathrm{M}\left(X_{v}\right)=v\right\}$ denote the set of servers whose corresponding messages are correctly mapped by M. Further, for each such node $v \in V_{\mathrm{M}}$, let $I_{v}=\left\{x \in \mathcal{X}: \mathrm{M}(x)=v, x \neq X_{v}\right\}$ denote all the messages (other than $v$ 's own message $X_{v}$ ) that are mapped to $v$. Then, by definition we have $R_{\mathrm{M}}=\left|V_{\mathrm{M}}\right| / \tilde{n}$ and

$$
\begin{align*}
\tilde{n} D_{\mathrm{M}} & =\sum_{v \in V_{H}} D_{\mathrm{M}}(v)=\sum_{v \in V_{\mathrm{M}}} \frac{1}{\left|I_{v}\right|+1} \\
& \geq \frac{\left|V_{\mathrm{M}}\right|^{2}}{\sum_{v \in V_{\mathrm{M}}}\left(\left|I_{v}\right|+1\right)} \geq \frac{\left|V_{\mathrm{M}}\right|^{2}}{\tilde{n}}=\tilde{n} R_{\mathrm{M}}^{2}, \tag{D.2}
\end{align*}
$$

where Equation (D.2) follows from the arithmetic-mean harmonic-mean (A.MH.M) inequality and $\sum_{v \in V_{\mathrm{M}}}\left(\left|I_{v}\right|+1\right) \leq \tilde{n}$. Hence we have $R_{\mathrm{M}} \leq \sqrt{D_{\mathrm{M}}}$, which upon taking expectation and using Jensen's inequality, yields $\mathbf{R}_{M} \leq \sqrt{\mathbf{D}_{M}}$.

## D.1.2 Proof of Theorem 5.2

Recall that for honest server $v$, the tuple $\left(x, u, T_{u}(x)\right)$ is contained in $S_{v}$ if $v$ forwards message $x$ to adversarial node $u$ at time $T_{u}(x)$. Let us now define a related quantity $\bar{S}_{v}$ to denote the set of messages $x \in \mathcal{X}$ forwarded by $v$ to some adversary such that $x$ was not received by any adversarial node previously. This quantity $\bar{S}_{v}$ is useful in analyses involving the first-spy estimator. $\overline{\mathbf{S}}$ denotes the vector of all $\bar{S}_{v}$ 's.

Lemma D.1. If $v \in V_{H}$ is a honest server node in a network with a fraction $p$ of adversaries, then the recall of the first-spy estimator is $\mathbf{R}_{\mathrm{FS}}(v)=\mathbb{P}\left(X_{v} \in\right.$ $\left.\bar{S}_{v}\right) \geq p$.

Proof. Let $U \in \Gamma(v)$ denote the node to which $v$ first sends its message $X_{v}$. Then,

$$
\begin{align*}
\mathbb{P}\left(U \in V_{A}\right) & =\sum_{u \in V, u \neq v} \mathbb{P}(U=u) \mathbb{P}\left(U \in V_{A} \mid U=u\right) \\
& =\sum_{u \in V, u \neq v} \frac{1}{n-1} \mathbb{P}\left(U \in V_{A} \mid U=u\right)=\frac{n p}{n-1} \geq p \tag{D.3}
\end{align*}
$$

due to uniform distribution among the remaining nodes $V \backslash\{v\}$. Therefore we have

$$
\begin{equation*}
\mathbb{P}\left(X_{v} \in \bar{S}_{v}\right) \geq \mathbb{P}\left(U \in V_{A}\right) \geq p \tag{D.4}
\end{equation*}
$$

Thus $v$ 's message is contained in $\bar{S}_{v}$ with probability at least $p$. The case where $v$ simultaneously broadcasts $X_{v}$ to multiple nodes can also be similarly bounded as above, and hence the lemma follows.

To show Equation (5.7), note that $\mathbf{R}_{0 \mathrm{PT}} \geq \mathbf{R}_{\mathrm{FS}}(v) \geq p$, by Lemma D.1. Next, we show that the first-spy estimator also has a precision of at least $p^{2}$ regardless of the topology or spreading scheme. Consider a random realization $\overline{\mathbf{S}}$, in which the adversaries observe a set of first-received messages $S_{v} \subseteq \mathcal{X}$ from each node $v \in V$. Now, supposing in these observations there exists a subset of $t$ server nodes $\left\{v_{1}, v_{2}, \ldots, v_{t}\right\}$ whose own messages are included in the respective forwarded sets, i.e., $X_{v_{i}} \in \bar{S}_{v_{i}} \forall i=1,2, \ldots, t$. The
macro-averaged precision in this case is

$$
\begin{equation*}
D_{\mathrm{FS}}=\frac{1}{\tilde{n}} \sum_{i=1}^{t} \frac{1}{\left|\bar{S}_{v_{i}}\right|} \geq \frac{t^{2}}{\tilde{n} \sum_{i=1}^{\tilde{n}}\left|\bar{S}_{v_{i}}\right|} \geq \frac{t^{2}}{\tilde{n}^{2}} \tag{D.5}
\end{equation*}
$$

where the first inequality above is due to the arithmetic-mean harmonicmean (A.M-H.M) inequality, and the second inequality is because the total number of messages is at most $\tilde{n}$. Equation (D.5) in turn implies that

$$
\begin{equation*}
\mathbb{E}\left[D_{\mathrm{FS}} \mid T=t\right] \geq \frac{t^{2}}{\tilde{n}^{2}} \tag{D.6}
\end{equation*}
$$

The overall expected detection precision can then be bounded as

$$
\begin{align*}
\mathbf{D}_{\mathrm{FS}} & =\mathbb{E}\left[D_{\mathrm{FS}}\right]=\sum_{t=0}^{\tilde{n}} \mathbb{P}(T=t) \mathbb{E}\left[D_{\mathrm{FS}} \mid T=t\right] \\
& \geq \sum_{t=0}^{\tilde{n}} \mathbb{P}(T=t) \frac{t^{2}}{\tilde{n}^{2}}=\frac{\mathbb{E}\left[T^{2}\right]}{\tilde{n}^{2}} \geq \frac{\mathbb{E}[T]^{2}}{\tilde{n}^{2}} \\
& =\frac{\mathbb{E}\left[\sum_{v \in V_{H}} \mathbf{1}_{X_{v} \in \bar{S}_{v}}\right]^{2}}{\tilde{n}^{2}} \geq \frac{(p \tilde{n})^{2}}{\tilde{n}^{2}}=p^{2}, \tag{D.7}
\end{align*}
$$

where the inequality in Equation (D.7) follows from Lemma D.1. Finally by definition we have $\mathbf{D}_{\mathrm{OPT}} \geq \mathbf{D}_{\mathrm{FS}}$ and hence the theorem follows.

## D.1.3 Proof of Theorem 5.3

Let us first prove that the optimal mapping must be a matching. Supposing otherwise, consider a mapping $M \in \mathcal{M}$ that is not a matching. Then there exists a server $v$ that is mapped to the most number of messages $\left\{x_{1}, x_{2}, \ldots, x_{k}\right\}$ $(k>1)$ in M . This also implies there exists another node $u \in V_{H}$ such that no message is mapped to $u$. Now, the expected precision at server $v$ is given by

$$
\begin{equation*}
\mathbb{E}\left[D_{\mathbb{M}}(v) \mid \mathbf{O}\right]=\frac{\sum_{i=1}^{k} \mathbb{P}\left(X_{v}=x_{i} \mid \mathbf{O}\right)}{k} \leq \max _{i \in\{1, \ldots, k\}} \mathbb{P}\left(X_{v}=x_{i} \mid \mathbf{O}\right) \tag{D.8}
\end{equation*}
$$

On the other hand, the expected precision at $u$ is zero. Now, consider an alternative mapping $\mathrm{M}^{\prime} \in \mathcal{M}$ in which all messages $x \in \mathcal{X}$ are mapped to servers exactly as in M except for the message $x_{i^{*}}$ where $i^{*}=\operatorname{argmin}_{i \in\{1, \ldots, k\}} \mathbb{P}\left(X_{v}=\right.$
$\left.x_{i} \mid \mathbf{O}\right)$ which is mapped to server $u$. In this case, the expected precision at $v$ becomes

$$
\begin{equation*}
\mathbb{E}\left[D_{\mathrm{M}^{\prime}}(v) \mid \mathbf{O}\right]=\frac{\sum_{i=1, i \neq i^{*}}^{k} \mathbb{P}\left(X_{v}=x_{i} \mid \mathbf{O}\right)}{k-1} \geq \mathbb{E}\left[D_{\mathrm{M}}(v) \mid \mathbf{O}\right] \tag{D.9}
\end{equation*}
$$

while the expected precision at $u$ is

$$
\begin{equation*}
\mathbb{E}\left[D_{M^{\prime}}(u) \mid \mathbf{O}\right]=\mathbb{P}\left(X_{u}=x_{i^{*}} \mid \mathbf{O}\right) \geq 0 \tag{D.10}
\end{equation*}
$$

As such the total expected precision at servers $u$ and $v$ is

$$
\begin{align*}
\mathbb{E}\left[D_{M^{\prime}}(v)\right. & \left.+D_{\mathrm{M}^{\prime}}(u) \mid \mathbf{O}\right]
\end{align*} \quad \geq \mathbb{E}\left[D_{\mathrm{M}}(v)+D_{\mathrm{M}}(u) \mid \mathbf{O}\right] .
$$

Thus we have constructed a new mapping $\mathrm{M}^{\prime}$ whose expected precision is at least as much as $M$ and in which the maximum number of messages mapped to any server is smaller by $1 .{ }^{1}$ Continuing this process, we conclude that for any mapping $\mathrm{M} \in \mathcal{M}$ there exists another matching mapping $\mathrm{M}^{\prime}$ such that $\mathbb{E}\left[D_{\mathrm{M}^{\prime}} \mid \mathbf{O}\right] \geq \mathbb{E}\left[D_{\mathrm{M}} \mid \mathbf{O}\right]$. Thus the optimizing mapping is achieved by a matching.

Now, let $\mathcal{M}^{*}$ denote the set of all matchings in the bipartite graph $\left(V_{H}, \mathcal{X}\right)$. By the first part of the theorem above, we can restrict our search to $\mathcal{M}^{*}$ for finding the optimal mapping. As such,

$$
\begin{equation*}
\mathbb{E}\left[D_{\mathrm{OPT}} \mid \mathbf{O}\right]=\max _{\mathrm{M} \in \mathcal{M}^{*}} \mathbb{E}\left[D_{\mathrm{M}} \mid \mathbf{O}\right]=\max _{\mathrm{M} \in \mathcal{M}^{*}} \sum_{(v, x) \in \mathrm{M}} \mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right), \tag{D.12}
\end{equation*}
$$

implying that the optimum is achieved by a maximum weight matching.

## D.1.4 Proof of Corollary 5.1

Let $\mathrm{M} \in \mathcal{M}$ be any mapping under observations $\mathbf{O}=(\mathbf{S}, \boldsymbol{\Gamma})$. Consider a server $v$ and let $\left\{x_{1}, x_{2}, \ldots, x_{k}\right\}$ be the set of messages that are mapped to $v$

[^27]in M. Then,
\[

$$
\begin{align*}
\mathbb{E}\left[D_{\mathrm{M}}(v) \mid \mathbf{O}\right] & \leq \frac{\sum_{i=1}^{k} \mathbb{P}\left(X_{v}=x_{i} \mid \mathbf{O}\right)}{k} \\
& \leq \max _{i \in\{1, \ldots, k\}} \mathbb{P}\left(X_{v}=x_{i} \mid \mathbf{O}\right) \\
& \leq \max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right) \tag{D.13}
\end{align*}
$$
\]

Since the above Equation (D.13) holds for any mapping M, it must hold for the optimal mapping as well.

## D.1.5 Proof of Theorem 5.4

We want to prove that the optimal mapping must map each message $x \in \mathcal{X}$ to a server $v$ that maximizes $\mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right)$. Supposing otherwise, let us consider a mapping $\mathrm{M} \in \mathcal{M}$ where there exists a server $w$ that is mapped to a set of messages $\left\{x_{1}, x_{2}, \ldots, x_{k}\right\}(k \geq 1)$, where w.l.o.g. $w \notin \operatorname{argmax} \mathbb{P}\left(X_{v}=x_{1} \mid \mathbf{O}\right)$. The expected recall at server $w$ is given by

$$
\mathbb{E}\left[R_{\mathbb{B}}(w) \mid \mathbf{O}\right]=\sum_{i=1}^{k} \mathbb{P}\left(X_{w}=x_{i} \mid \mathbf{O}\right)
$$

Further, consider another node $u \in V_{H}$ such that $u \in \underset{v \in V_{H}}{\operatorname{argmax}} \mathbb{P}\left(X_{v}=x_{1} \mid \mathbf{O}\right)$. Suppose it is mapped to a different set of messages $\left\{y_{1}, \ldots, y_{j}\right\}$. The expected recall for node $u$ is

$$
\mathbb{E}\left[R_{\mathrm{B}}(u) \mid \mathbf{O}\right]=\sum_{i=1}^{j} \mathbb{P}\left(X_{u}=y_{i} \mid \mathbf{O}\right)
$$

Now, consider an alternative mapping $\mathrm{M}^{\prime} \in \mathcal{M}$ in which all messages $x \in \mathcal{X}$ are mapped to servers exactly as in M except for the message $x_{1}$, which is mapped to server $u$. In this case, the expected recall at $w$ becomes

$$
\mathbb{E}\left[R_{M^{\prime}}(w) \mid \mathbf{O}\right]=\sum_{i=2}^{k} \mathbb{P}\left(X_{w}=x_{i} \mid \mathbf{O}\right)
$$

while the expected recall at $u$ is

$$
\begin{equation*}
\mathbb{E}\left[R_{\mathbb{M}^{\prime}}(u) \mid \mathbf{O}\right]=\mathbb{E}\left[R_{\mathbb{M}}(u) \mid \mathbf{O}\right]+\mathbb{P}\left(X_{u}=x_{1} \mid \mathbf{O}\right) \tag{D.14}
\end{equation*}
$$

As such the total expected precision at servers $u$ and $v$ is

$$
\begin{align*}
\mathbb{E}\left[D_{\mathrm{M}^{\prime}}(w)+D_{\mathrm{M}^{\prime}}(u) \mid \mathbf{O}\right] & \geq \mathbb{E}\left[D_{\mathrm{M}}(w)+D_{\mathrm{M}}(u) \mid \mathbf{O}\right] \\
\Rightarrow \mathbb{E}\left[D_{\mathrm{M}^{\prime}} \mid \mathbf{O}\right] & \geq \mathbb{E}\left[D_{\mathrm{M}} \mid \mathbf{O}\right] \tag{D.15}
\end{align*}
$$

Thus we have constructed a new mapping $\mathrm{M}^{\prime}$ whose expected precision is at least as much as $M$ and in which the number of messages mapped to servers with sub-maximal likelihood is reduced by one. Continuing this process, we conclude that for any mapping $\mathrm{M} \in \mathcal{M}$ there exists another mapping $\mathrm{M}^{\prime}$ such that each message $x$ is mapped to a server $v^{*} \in \underset{v \in V_{H}}{\operatorname{argmax}} \mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right)$.

## D. 2 Section 5.3: Baseline Algorithms

## D.2.1 Proof of Proposition 5.1

Consider the broadcasting experiment on a random realization $G$ of the network topology. In the static case this topology is completely known to the adversary. As defined in Section D.1.2, for any honest server $v \in V_{H}$, let $\bar{S}_{v}$ denote the set of transactions $x \in \mathcal{X}$ such that $x$ is directly forwarded by $v$ to some adversary and $x$ has not been received by any adversary previously. By our assumption on flooding, this means a server $v$ 's message is contained in $\bar{S}_{u}$ if and only if
(i) $u$ is reachable from $v$,
(ii) $u$ has an out-going edge to an adversary and
(iii) no other node $u^{\prime} \in V_{H}$ that satisfies the previous two conditions is strictly closer to $v$ than $u$.

Thus by looking at the graph $G$, the adversary can construct a bipartite graph $B\left(V_{H}, V_{H}\right)$ in which there is edge $(u, v) \in V_{H} \times V_{H}$ if and only if $X_{v}$ will be contained in $\bar{S}_{u}$. Further for each $u \in V_{H}$, let $W_{u}=\left\{v \in V_{H}:(u, v) \in\right.$
$B, \nexists u^{\prime} \neq u$ s.t. $\left.\left(u^{\prime}, v\right) \in B\right\}$ denote the set of server nodes whose messages reach only the message set $\bar{S}_{u}$. Note that for servers $v \in V_{H}$ that have an out-going edge to an adversary, we must have $v \in W_{v}$.

Now, once the messages have been broadcast in $G$, consider the following mapping strategy M for the adversary. First, for each set $\bar{S}_{v}$ we compute a subset $\bar{S}_{v}^{\prime}=\left\{x \in \bar{S}_{v}: x \notin \bar{S}_{u} \forall u \neq v\right\}$. Such a set $\bar{S}_{v}^{\prime}$ corresponds to the messages that were delivered to the adversaries only by server $v$ and no other server. Thus, the messages in $\bar{S}_{v}^{\prime}$ must precisely belong to the servers in $W_{v}$. As such, the adversary's mapping strategy can be: (i) for each $v \in V_{H}$ pick a random matching mapping between $\bar{S}_{v}^{\prime}$ and $W_{v}$ and (ii) assign any remaining messages randomly to the remaining server nodes. Let $\mathcal{E}_{v}$ denote the event that $v$ has an out-going edge to an adversary. The payoff can then be bounded as

$$
\begin{align*}
\mathbb{E}\left[\tilde{n} D_{\mathrm{M}} \mid G\right] & \geq \mathbb{E}\left[\sum_{\substack{v \in V_{H}: \\
\left|W_{v}\right| \geq 1}} \sum_{u \in W_{v}} \mathbb{1}\left\{\mathrm{M}\left(X_{u}\right)=u\right\} \mid G\right] \\
& =\sum_{\substack{v \in V_{H}: \\
\left|W_{v}\right| \geq 1}} \sum_{u \in W_{v}} \mathbb{E}\left[\mathbb{1}\left\{\mathrm{M}\left(X_{u}\right)=u\right\} \mid G\right] \\
& =\sum_{\substack{v \in V_{H}: \\
\left|W_{v}\right| \geq 1}} \sum_{u \in W_{v}} \frac{1}{\left|W_{v}\right|}=\sum_{v \in V_{H}} \mathbb{1}\left\{\left|W_{v}\right| \geq 1\right\} \tag{D.16}
\end{align*}
$$

where Equation (D.16) follows because in a random matching any message in $S_{v}^{\prime}$ is likely to be assigned to its true server in $W_{v}$ with probability $1 /\left|W_{v}\right|$. Hence the total average precision is bounded by

$$
\begin{equation*}
\tilde{n} \mathbf{D}_{\mathrm{M}} \geq \mathbb{E}\left[\sum_{v \in V_{H}} \mathbb{1}\left\{\left|W_{v}\right| \geq 1\right\}\right] \geq \sum_{v \in V_{H}} \mathbb{P}\left(\mathcal{E}_{v}\right)=\tilde{n}\left(1-(1-p)^{d}\right) \tag{D.17}
\end{equation*}
$$

and we have the proposition.

## D.2.2 Proof Sketch of Proposition 5.2

Before we begin the proof, first notice that in the dynamic setting the adversaries are directly connected to at most $d p n$ (i.e., roughly a fraction $p$ ) honest servers, while the rest of the server locations are unknown to the adversary.


Figure D.1: Comparison of the number of adversarial nodes receiving a message in two rounds following first reception when the source is (i) directly connected to an adversary (left) and (ii) away from the adversary (right). The propagation of the message is shown in red; darkened nodes are adversarial.

Since the hidden servers can only be trivially deanonymized, in order to obtain our claimed average precision of $O(p)$ each of the servers visible to the adversary must be deanonymized with a high precision close to 1 . Indeed, in the following we describe a simple mapping scheme $M$ that achieves this high precision.

For a server $v \in V_{H}$, let $\mathcal{E}_{v}$ denote the event that at least one of $v$ 's outgoing edges is connected to an adversary. Consider then, the spreading of $v$ 's message $X_{v}$ in the graph under event $\mathcal{E}_{v}$. Since $G$ is a random $d$-regular graph, using the result in [210], there is almost surely a regular tree of depth at least $\frac{1}{2} \log _{d-1} n$ rooted at $v$. For simplicity, let us consider $d=4$ in which case there is a tree of depth at least $\frac{1}{4} \log n$ rooted at $v$ almost surely. Thus $v$ 's message propagates along this tree, reaching two nodes (at least one of which is an adversary, due to $\mathcal{E}_{v}$ ) in the first round and subsequently reaching $2^{i}$ new nodes in the $i$-th round for each $i<\frac{1}{4} \log n$. Since a fraction $p$ of the nodes are adversarial, this implies in the $i$-th round we expect roughly $p 2^{i}$ adversarial nodes to receive $X_{v}$. On the other hand, if some other server $u$ upstream of $v$ had started broadcasting its message $X_{u}$, then more adversarial nodes would have received $X_{u}$ in the $i$-th round following reception by the first adversarial node (see Figure D.1).

The above observation then, naturally motivates a mapping $M$ as described in Algorithm 15. In this strategy, the adversary simply counts the number of adversarial nodes that received a particular message $x \in \mathcal{X}$ at a time $\frac{1}{4} \log n-1$ rounds after the message was first received by some adversarial node $a$. If this number is small $\left(<2 p n^{1 / 4}\right)$ then we conclude the source of $x$ to be the server $v$ that sent the message to the first adversary. Otherwise

```
Algorithm 15: Mapping algorithm under flooding for a dynamic 4-
regular graph.
    Input: Time-stamp \(T_{v}(x)\) and sender \(S_{v}(x)\) for each message \(x \in \mathcal{X}\)
            received by adversary \(v\) for all \(v \in V_{A}\).
    Output: Mapping M from \(\mathcal{X}\) to \(V_{H}\)
    \(I \leftarrow V_{H}, J \leftarrow \mathcal{X}\)
    for each \(x \in \mathcal{X}\) do
        \(a_{\text {init }} \leftarrow \underset{v \in V_{A}}{\operatorname{argmin}} T_{v}(x)\)
        \(T_{\text {init }} \leftarrow T_{a_{\text {init }}}(x)\)
        \(v_{\text {init }} \leftarrow S_{a_{\text {init }}}(x)\)
        \(\eta \leftarrow\left|\left\{v \in V_{A}: T_{v}(x)=T_{\text {init }}+\frac{1}{4} \log n-1\right\}\right|\)
        if \(\eta<2 p n^{1 / 4}\) then
            \(\mathrm{M}(x) \leftarrow v_{\text {init }}\)
            \(I \leftarrow I \backslash\left\{v_{\text {init }}\right\}\)
            \(J \leftarrow J \backslash\{x\}\)
        end
    end
    Randomly assign messages in \(J\) to servers in \(I\)
    return M
```

the message is randomly assigned to an unassigned server at the end.
The algorithm works because if $v$ were truly the source of $x$, then in the $\left(\frac{1}{4} \log n-1\right)$-th round following reception by $a$, the number of adversarial nodes to receive $x$ is less than $2 p n^{1 / 4}$ with a probability at least $1-2^{-\log (4 / e) p n^{1 / 4}}$ by the Chernoff bound. On the other hand if $v$ were not the true source of $x$, then $x$ was initially broadcast at a time at least 2 rounds before $a$ received it. This implies at least $2 p n^{1 / 4}$ adversarial nodes receive the message at a time $\frac{1}{4} \log n-1$ rounds following reception by $a$. Thus the total probability of error can be bounded by the union bound, to yield that whenever a server $v$ 's out-going edges are connected to at least one adversarial node, $X_{v}$ is mapped to $v$ with precision 1 with high probability. Such an event $\mathcal{E}_{v}$ happens with a probability at least $p$ to conclude the proposition.

## D.2.3 Proof of Proposition 5.3

For any message $X_{u}$, let $\Pi_{u}=\left(\Pi_{1, u}, \Pi_{2, u}, \ldots, \Pi_{L_{u}, u}\right)$ be the path taken by a message from its source $u\left(=\Pi_{1, u}\right)$ until it reaches an adversarial node $\Pi_{L_{u}, u}$ for the first time ( $L_{u}$ denotes the length of the path). Further for any two
nodes $v, u \in V_{H}$, let $\mathcal{E}_{u, v}$ denote the event that $u$ 's message $X_{u}$ reaches the adversary through server $v$, i.e., $\Pi_{1, u}=u, \Pi_{2, u} \notin V_{A}, \Pi_{3, u} \notin V_{A}, \ldots, \Pi_{k-2, u} \notin$ $V_{A}, \Pi_{k-1, u}=v$ and $\Pi_{k, u} \in V_{A}$. Then by counting over paths of all possible lengths, we can evaluate probability of $\mathcal{E}_{u, u}$ as

$$
\begin{equation*}
\mathbb{P}\left(\mathcal{E}_{u, u}\right)=\sum_{l \geq 2} \mathbb{P}\left(L_{k}=l, \mathcal{E}_{u, u}\right)=\left(\frac{n p}{n}\right)+\sum_{l \geq 3}\left(\frac{\tilde{n}}{n}\right)^{l-3}\left(\frac{1}{n}\right)\left(\frac{n p}{n}\right)=p+\frac{1}{n} . \tag{D.18}
\end{equation*}
$$

Similarly, for $u \in V_{H}, u \neq v$,

$$
\begin{equation*}
\mathbb{P}\left(\mathcal{E}_{u, v}\right)=\sum_{l \geq 3}\left(\frac{\tilde{n}}{n}\right)^{l-3}\left(\frac{1}{n}\right)\left(\frac{n p}{n}\right)=\frac{1}{n} . \tag{D.19}
\end{equation*}
$$

Further, since the messages are all forwarded independently the set of events $\left\{\mathcal{E}_{v, u}: v \in V_{H}\right\}$ are mutually independent for each server $u \in V_{H}$. Hence the expected cost incurred at a server under the first-spy estimator can be written as

$$
\begin{equation*}
\mathbf{D}_{\mathrm{FS}}(v)=\left(p+\frac{1}{n}\right) \mathbb{E}\left[\frac{1}{1+Z_{v}}\right]=\left(p+\frac{1}{n}\right) \frac{1}{\tilde{n} \frac{1}{n}}\left(1-\left(1-\frac{1}{n}\right)^{\tilde{n}}\right) \tag{D.20}
\end{equation*}
$$

where $Z_{v}=\sum_{u \in V_{H}, u \neq v} \mathbb{1}\left\{\mathcal{E}_{u, v}\right\}$ is the number of messages, other than $X_{v}$, that reach the adversary through $v$ and $Z_{v} \sim \operatorname{Binom}\left(\tilde{n}-1, \frac{1}{n}\right)$ because of independence of messages and Equation (D.19). The last equation above can be further simplified to yield the bound

$$
\begin{equation*}
\mathbf{D}_{\mathrm{FS}}(v) \geq \frac{p}{1-p}\left(1-e^{p-1}\right) \tag{D.21}
\end{equation*}
$$

which when averaged over all honest nodes $v \in V_{H}$ gives us the desired result.

## D. 3 Section 5.4: Main Result - Dandelion

## D.3.1 Proof of Theorem 5.5

We first show that the first-spy estimator is recall-optimal for dandelion spreading, then that the first-spy estimator has an expected recall of $p$.

To show the first step, i.e., $\mathbf{R}_{0 \mathrm{PT}}=\mathbf{R}_{\mathrm{FS}}$, Theorem 5.4 implies that we must show that for every message $x$, its exit node $z$ (i.e., the node implicated by the first-spy estimator) maximizes $\mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right)$. For any message $X_{u}$, let $\Pi_{u}=\left(\Pi_{1, u}, \Pi_{2, u}, \ldots, \Pi_{L_{u}, u}\right)$ be the path taken by a message from its source $u\left(=\Pi_{1, u}\right)$ until it reaches an adversarial node $\Pi_{L_{u}, u}$ for the first time ( $L_{u}$ denotes the length of the path). From the adversary's observation $\mathbf{S}, \Pi_{L_{u}-1, u}$ and $\Pi_{L_{u}, u}$ are fixed as the exit node $z$ and the first spy for $X_{u}$, respectively. Due to the specification of dandelion spreading (Algorithm 8), the likelihood of this path, $\mathcal{L}\left(\Pi_{u}\right)$, is $\mathcal{L}\left(\Pi_{u}\right)=\prod_{i=1}^{L_{u}-1} \frac{1}{\operatorname{deg}\left(\Pi_{i, u}\right)}$, where $\operatorname{deg}(v)$ denotes the out-degree of $v$. Assuming a uniform prior over candidate sources, we have $\mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right) \propto \mathcal{L}\left(\Pi_{v}\right)$. Since each node is assumed to have an outdegree of at least 1 , this likelihood is maximized by taking the shortest path possible. That is, the maximum-likelihood path over all paths originating at honest candidate sources gives $z \in \underset{v \in V_{H}}{\operatorname{argmax}} \mathbb{P}\left(X_{v}=x \mid \mathbf{O}\right)$. Hence the first-spy estimator is also a maximum-recall estimator.

Now we analyze the recall of the first-spy estimator. Let $\mathcal{P}_{v}$ denote the event that $v$ 's parent (i.e., the next node in the line) is adversarial. Then the expected recall is

$$
\begin{aligned}
\mathbf{R}_{0 \mathrm{PT}} & =\mathbb{E}\left[R_{\mathrm{FS}} \mid \mathbf{S}, G\right]=\frac{1}{\tilde{n}} \mathbb{E}\left[\sum_{v \in V_{H}} \mathbb{1}\left\{\mathcal{P}_{v}\right\}\right] \\
\Rightarrow \mathbf{R}_{0 \mathrm{PT}} & =\frac{1}{\tilde{n}} \sum_{v \in V_{H}} \mathbb{P}\left(\mathcal{P}_{v}\right)=\frac{1}{\tilde{n}} \sum_{v \in V_{H}}\left(\frac{n p}{n-1}\right)=p+O\left(\frac{1}{n}\right) .
\end{aligned}
$$

## D.3.2 Proof of Proposition 5.4

For any honest node $v$, let $W_{v}=\left\{u \in V_{H}: X_{u} \in S_{v}\right\}$ denote the ward under node $v$ and let $\mathbf{W}$ denote the set of all wards. Note that in the tree topology, the wards $W_{v}$ can be completely determined from knowledge of the graph $G$. Let $I(v)$ denote the node $u \in V_{H}$ such that $X_{v} \in S_{u}$, i.e., node $v$ belongs to the ward of node $I(v)$. Then, the expected cost at a node $v$ under observations $\mathbf{O}=(\mathbf{S}, G)$ by the adversary can be written as

$$
\begin{equation*}
\mathbb{E}\left[D_{\mathrm{M}}(v) \mid \mathbf{S}, G\right]=\mathbb{E}\left[D_{\mathrm{M}}(v) \mid \mathbf{S}, G, \mathbf{W}\right]=\frac{1}{\left|W_{I(v)}\right|} \tag{D.22}
\end{equation*}
$$

This follows because the matching estimator MAT assigns the messages in $S_{v}$ to the nodes in $W_{v}$ as a random matching, and hence the probability of a node receiving the correct message is $1 /\left|W_{v}\right|$. Summing Equation (D.22) over all honest nodes and averaging, we have

$$
\begin{equation*}
\mathbb{E}\left[D_{\mathrm{MAT}} \mid \mathbf{S}, G\right]=\frac{1}{\tilde{n}} \sum_{v \in V_{H}} \frac{1}{\left|W_{I(v)}\right|}=\frac{|W|}{\tilde{n}}, \tag{D.23}
\end{equation*}
$$

where $|W|=\left|\left\{v: W_{v} \neq \emptyset\right\}\right|$ denotes the number of non-empty wards, and $\emptyset$ denotes the null set. Now, let $\mathcal{P}_{v}$ denote the event that either $v$ 's parent is adversarial or $v$ is the root of the tree. Since a ward under a node $v$ is non-empty iff $v$ 's parent is adversarial or $v$ is a root node, Equation (D.23) above becomes

$$
\begin{align*}
\mathbb{E}\left[D_{\mathrm{MAT}} \mid \mathbf{S}, G\right] & =\frac{1}{\tilde{n}} \mathbb{E}\left[\sum_{v \in V_{H}} \mathbb{1}\left\{\mathcal{P}_{v}\right\}\right]  \tag{D.24}\\
\Rightarrow \mathbf{D}_{\mathrm{MAT}}=\frac{1}{\tilde{n}} \sum_{v \in V_{H}} \mathbb{P}\left(\mathcal{P}_{v}\right) & =\frac{1}{\tilde{n}} \sum_{v \in V_{H}}\left(\frac{1}{n}+\frac{n-1}{n} \frac{n p}{n-1}\right) \geq p .
\end{align*}
$$

## D.3.3 Proof of Proposition 5.5

First note that since the tree is dynamic, the adversary's observations consists of $\mathbf{O}=\left(\mathbf{S}, \Gamma\left(V_{A}\right)\right)$, i.e., the transaction logs and the local neighborhood of adversarial nodes. Now for any honest node $v \in V_{H}$, let $\mathcal{E}_{v}$ denote the event that (i) $v$ occurs at a position in $G$ which is a leaf of the tree and (ii) v's parent is an adversary. Similarly let $\mathcal{I}_{v}$ denote the event that $v \in V_{H}$ occurs at the interior of the tree. We first show that whenever $\mathcal{E}_{v}$ happens, $v$ is detected with certainty under the first-spy estimator, i.e.,

$$
\begin{equation*}
\mathbb{E}\left[D_{\mathrm{FS}}(v) \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}\right]=1 \tag{D.25}
\end{equation*}
$$

This is because $D_{\mathrm{FS}}(v)=\frac{\sum_{x \in S_{v}} \mathbb{1}\left\{X_{v}=x\right\}}{\left|S_{v}\right|}$ in the first-spy estimator and $S_{v}=$ $\left\{X_{v}\right\}$ whenever $\mathcal{E}_{v}$ happens. As such,

$$
\begin{align*}
\mathbb{E}\left[D_{\mathrm{FS}}(v) \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}\right] & =\mathbb{E}\left[\mathbb{1}\left\{X_{v}=X_{v}\right\} \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}\right]=1 \\
\Rightarrow \mathbb{E}\left[D_{\mathrm{FS}}(v) \mid \mathcal{E}_{v}\right] & =1 \tag{D.26}
\end{align*}
$$

Hence the expected payoff becomes

$$
\begin{align*}
\mathbf{D}_{\mathrm{FS}}(v) & =\mathbb{P}\left(\mathcal{E}_{v}\right) \mathbb{E}\left[D_{\mathrm{FS}}(v) \mid \mathcal{E}_{v}\right]+\mathbb{P}\left(\mathcal{I}_{v}\right) \mathbb{E}\left[D_{\mathrm{FS}}(v) \mid \mathcal{I}_{v}\right]  \tag{D.27}\\
& \geq \mathbb{P}\left(\mathcal{E}_{v}\right) \mathbb{E}\left[D_{\mathrm{FS}}(v) \mid \mathcal{E}_{v}\right]=\frac{1}{2} \frac{n p}{(n-1)} \geq \frac{p}{2} \tag{D.28}
\end{align*}
$$

since at least half of the nodes are leaves in a perfect $d$-ary tree. Summing over all honest nodes gives the result.

## D.3.4 Proof of Theorem 5.6

As in the case of dynamic trees, the adversary's observations consists of $\mathbf{O}=$ $\left(\mathbf{S}, \Gamma\left(V_{A}\right)\right)$ in the dynamic line as well. The proof works by evaluating the cost incurred under various possibilities for the local neighborhood structure around a node in the network. For any honest server node $v \in V_{H}$, let $\mathcal{E}_{v}(i, j)$ denote the event that (i) $i$ nodes preceding $v$ are honest nodes, the $(i+1)$-th node preceding $v$ is adversarial and (ii) $j$ nodes succeeding $v$ are honest nodes and the $(j+1)$-th node following $v$ is adversarial. Also for ease of notation let $\mathcal{I}_{v}$ denote the event $\cup_{i>0, j>0} \mathcal{E}_{v}(i, j)$. Then the following lemmas hold true.

Lemma D.2. On a line-graph, for any $i, j>0$, we have

$$
\begin{align*}
& \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right) \mid \mathcal{E}_{v}(i, 0)\right] \leq \frac{1}{i+1} \\
& \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(0, j)\right) \mid \mathcal{E}_{v}(0, j)\right] \leq \frac{1}{j+1} \\
& \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(0,0)\right) \mid \mathcal{E}_{v}(0,0)\right] \leq 1 \\
& \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{I}_{v}\right) \mid \mathcal{I}_{v}\right] \leq \frac{1}{n(1-3 p)} . \tag{D.29}
\end{align*}
$$

Proof. Consider a realization $G$ of the network topology such that our desired event $\mathcal{E}_{v}(i, 0)$ happens. In such a graph $G$, the node succeeding $v$ is an adversarial node and the $i$ nodes preceding $v$ are honest. Let us denote this set of $i+1$ nodes - comprising of the $i$ nodes preceding $v$ and $v$ itself - as $W_{v}$ (i.e., the ward of $v$ ). Now, if the messages assigned to the nodes outside of $W_{v}$ is denoted by $X\left(V_{H} \backslash W_{v}\right)$, then for any $x \in S_{v}$ we have $\mathbb{P}\left(X_{v}=\right.$

$$
\begin{align*}
x \mid G, \mathbf{S}, \Gamma\left(V_{A}\right) & , \mathcal{E}_{v}(i, 0), X\left(V_{H} \backslash W_{v}\right) \\
& =\frac{\mathbb{P}\left(X_{v}=x, \mathbf{S}, X\left(V_{H} \backslash W_{v}\right) \mid G, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right)}{\sum_{x \in S_{v}} \mathbb{P}\left(X_{v}=x, \mathbf{S}, X\left(V_{H} \backslash W_{v}\right) \mid G, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right)} \\
= & \frac{\mathbb{P}\left(X_{v}=x, X\left(V_{H} \backslash W_{v}\right) \mid G, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right)}{\sum_{x \in S_{v}} \mathbb{P}\left(X_{v}=x, X\left(V_{H} \backslash W_{v}\right) \mid G, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right)} \\
= & \frac{1}{i+1}, \tag{D.30}
\end{align*}
$$

by using the fact that the allocation of messages $\mathbf{X}$ is independent of the graph structure $\left(G, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right)$ and

$$
\mathbb{P}\left(\mathbf{S} \mid X_{v}=x, X\left(V_{H} \backslash W_{v}\right), G, \Gamma\left(V_{A}\right) \mathcal{E}_{v}(i, 0)\right)=1
$$

on a line-graph. Now, taking expectation on both sides of Equation (D.30) we get

$$
\begin{align*}
\mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right) & =\frac{1}{i+1} \forall x \in S_{v} \\
\Rightarrow \max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right) & =\frac{1}{i+1} \text { or } \\
\mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right) \mid \mathcal{E}_{v}(i, 0)\right] & =\frac{1}{i+1} . \tag{D.31}
\end{align*}
$$

By a similar argument as above, we can also show that

$$
\begin{align*}
& \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(0, j)\right) \mid \mathcal{E}_{v}(0, j)\right]=\frac{1}{j+1}, \\
& \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(0,0)\right) \mid \mathcal{E}_{v}(0,0)\right]=1 \tag{D.32}
\end{align*}
$$

Finally let us consider the case where $v$ is an interior node, i.e., event $\mathcal{I}_{v}$ happens. As before, for a head-node $u$ (an honest node whose successor is an adversarial node) let $W_{u}$ denote the ward containing $u$. Notice that under observations $\mathbf{S}, \Gamma\left(V_{A}\right)$ the adversaries know (i) the head and tail nodes of each ward (from $\left.\Gamma\left(V_{A}\right)\right)$ and (ii) the size of each ward $\left(\left|W_{u}\right|=\left|S_{u}\right|\right)$. Therefore if a message $x$ is such that $x \in S_{u}$ for some $u$, then

$$
\begin{aligned}
& \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{I}_{v}\right)=\mathbb{P}\left(X_{v}=x, v \in W_{u} \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{I}_{v}\right) \\
& \quad=\mathbb{P}\left(v \in W_{u} \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{I}_{v}\right) \mathbb{P}\left(X_{v}=x \mid v \in W_{u}, \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{I}_{v}\right)
\end{aligned}
$$

$$
\begin{equation*}
=\frac{\left|W_{u}\right|-2}{|I|} \frac{1}{\left|W_{u}\right|} \leq \frac{1}{|I|} \leq \frac{1}{n(1-3 p)}, \tag{D.33}
\end{equation*}
$$

where $I$ denotes the set of all interior nodes and $|I| \geq n(1-3 p)$ since each adversary is a neighbor to at most two honest server nodes. Hence we have

$$
\begin{equation*}
\mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{I}_{v}\right) \mid \mathcal{I}_{v}\right] \leq \frac{1}{n(1-3 p)} \tag{D.34}
\end{equation*}
$$

concluding the proof.
Lemma D.3. On a line-graph, for $i, j>0$ we have

$$
\begin{align*}
\mathbb{P}\left(\mathcal{E}_{v}(i, 0)\right) & \leq\left(p+\frac{1}{n}\right)^{2}\left(1-p+\frac{2}{n}\right)^{i}  \tag{D.35}\\
\mathbb{P}\left(\mathcal{E}_{v}(0, j)\right) & \leq\left(p+\frac{1}{n}\right)^{2}\left(1-p+\frac{2}{n}\right)^{j}  \tag{D.36}\\
\mathbb{P}\left(\mathcal{E}_{v}(0,0)\right) & \leq(p+1 / n)^{2}  \tag{D.37}\\
\mathbb{P}\left(\mathcal{I}_{v}\right) & \leq(1-p)^{2} . \tag{D.38}
\end{align*}
$$

Proof. First let us consider the event $\mathcal{E}_{v}(i, 0)$ in which node $v$ has an adversarial successor, $i$ honest predecessor nodes and an adversarial $i+1$-th predecessor. Let $Y_{v}$ denote the position of node $v$ in the line graph. Then

$$
\begin{equation*}
\mathbb{P}\left(\mathcal{E}_{v}(i, 0)\right)=\sum_{j=i+1}^{n} \mathbb{P}\left(Y_{v}=j\right) \mathbb{P}\left(\mathcal{E}_{v}(i, 0) \mid Y_{v}=j\right) \tag{D.39}
\end{equation*}
$$

since $v$ needs to be at a position on the line graph where at least $i+1$ predecessors are feasible. Now, for $i+1 \leq j \leq n$, by a simple counting argument we have

$$
\begin{aligned}
\mathbb{P}\left(\mathcal{E}_{v}(i, 0) \mid Y_{v}=j\right) & =\left(\frac{n p}{n-1}\right)\left(\frac{n p-1}{n-2}\right)\left(\frac{\tilde{n}-1}{n-3}\right)\left(\frac{\tilde{n}-2}{n-4}\right) \ldots\left(\frac{\tilde{n}-i}{n-i-2}\right) \\
& \leq\left(p+\frac{1}{n}\right)^{2}\left(1-p+\frac{2}{n}\right)^{i}
\end{aligned}
$$

Combining the above inequality with Equation (D.39) we conclude that

$$
\begin{equation*}
\mathbb{P}\left(\mathcal{E}_{v}(i, 0)\right) \leq\left(p+\frac{1}{n}\right)^{2}\left(1-p+\frac{2}{n}\right)^{i} \tag{D.40}
\end{equation*}
$$

for $i>0$. By essentially a similar counting as above, we can also obtain the remaining Equations (D.36), (D.37) and (D.38) from the lemma.

Lemma D.4. If $\mathcal{E}_{1}, \mathcal{E}_{2}, \ldots, \mathcal{E}_{k}$ is a set of mutually exclusive and exhaustive events, and $v \in V_{H}$ is any honest server node, then

$$
\begin{equation*}
\mathbf{D}_{0 \mathrm{PT}}(v) \leq \sum_{i=1}^{k} \mathbb{P}\left(\mathcal{E}_{i}\right) \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{i}\right) \mid \mathcal{E}_{i}\right] \tag{D.41}
\end{equation*}
$$

Proof. The proof is straightforward and follows from Corollary 5.1. From Equation (5.8) we have

$$
\begin{align*}
\mathbb{E} & {\left[D_{\mathrm{opT}}(v) \mid \mathbf{S}, \Gamma\left(V_{A}\right)\right] \leq \max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right)\right) } \\
& =\max _{x \in \mathcal{X}} \sum_{i=1}^{k} \mathbb{P}\left(\mathcal{E}_{i} \mid \mathbf{S}, \Gamma\left(V_{A}\right)\right) \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{i}\right) \\
& \leq \sum_{i=1}^{k} \mathbb{P}\left(\mathcal{E}_{i} \mid \mathbf{S}, \Gamma\left(V_{A}\right)\right) \max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{i}\right) . \tag{D.42}
\end{align*}
$$

Taking expectation on both sides of the above equation, we get

$$
\begin{aligned}
\mathbf{D}_{\mathrm{OPT}}(v) & \leq \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{i}\right)\right] \\
\Rightarrow & \mathbf{D}_{\mathrm{OPT}}(v) \leq \sum_{i=1}^{k} \mathbb{P}\left(\mathcal{E}_{i}\right) \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{i}\right) \mid \mathcal{E}_{i}\right],
\end{aligned}
$$

and thus proving the lemma.
To complete the proof of the Theorem, let use Lemma D. 4 with $\mathcal{E}_{v}(i, 0)$, $\mathcal{E}_{v}(0, j), \mathcal{E}_{v}(0,0)$ and $\mathcal{E}_{v}$ for $i, j>0$ as the set of mutually exclusive and exhaustive events. Then the expected payoff at $v$ can be bounded as

$$
\begin{array}{r}
\mathbf{D}_{\mathrm{oPT}}(v) \leq \sum_{i>0} \mathbb{P}\left(\mathcal{E}_{v}(i, 0)\right) \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(i, 0)\right) \mid \mathcal{E}_{v}(i, 0)\right] \\
+\sum_{j>0} \mathbb{P}\left(\mathcal{E}_{v}(0, j)\right) \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(0, j)\right) \mid \mathcal{E}_{v}(0, j)\right] \\
+\mathbb{P}\left(\mathcal{E}_{v}(0,0)\right) \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{E}_{v}(0,0)\right) \mid \mathcal{E}_{v}(0,0)\right] \\
+\mathbb{P}\left(\mathcal{I}_{v}\right) \mathbb{E}\left[\max _{x \in \mathcal{X}} \mathbb{P}\left(X_{v}=x \mid \mathbf{S}, \Gamma\left(V_{A}\right), \mathcal{I}_{v}\right) \mid \mathcal{I}_{v}\right] \tag{D.43}
\end{array}
$$

where the values of the individual expectation and probability terms in the
above Equation (D.43) have been computed in Lemmas D. 2 and D. 3 respectively. Using those bounds, we get

$$
\begin{align*}
\mathbf{D}_{\mathrm{OPT}}(v) \leq & \sum_{i>0}\left(p+\frac{1}{n}\right)^{2}\left(1-p+\frac{2}{n}\right)^{i} \frac{1}{i+1} \\
& +\sum_{j>0}\left(p+\frac{1}{n}\right)^{2}\left(1-p+\frac{2}{n}\right)^{j} \frac{1}{i+1}+\left(p+\frac{1}{n}\right)^{2}+\frac{(1-p)^{2}}{n(1-3 p)} \\
\leq & \frac{2\left(p+\frac{1}{n}\right)^{2}}{\left(1-p+\frac{2}{n}\right)} \log \left(\frac{1}{p-\frac{2}{n}}\right)+\frac{(1-p)^{2}}{n(1-3 p)} \\
\leq & \frac{2 p^{2}}{1-p} \log \left(\frac{2}{p}\right)+O\left(\frac{1}{n}\right) . \tag{D.44}
\end{align*}
$$

Finally averaging the expected payoff $\mathbf{D}_{\mathrm{OPT}}(v)$ over each of the $\tilde{n}$ honest server nodes $v \in V_{H}$, we get the desired result.

## D. 4 Section 5.5: Systems Issues

## D.4.1 Proof of Proposition 5.6

We map the problem of constructing a line (i.e., a 2-regular digraph) to one of assigning balls to bins. Suppose each ball represents an outgoing connection, and each bin represents a server who may accept that outgoing connection. There are $n$ balls and $n$ bins; for the sake of simplicity, we assume that a server can establish a connection to itself, so all $n$ bins are available to each ball. Then the maximum degree $d_{m}$ of the degree distribution is linearly related to the maximum number of balls in any bin $h_{m}$. That is, $d_{m}=1+h_{m}$. When $k=1$, each ball is assigned to a bin uniformly. The quantity $h_{m}$ has been studied extensively in this case, and the result for $k=1$ in Proposition 5.6 is well-known [211]. When $k>1$, Algorithm 9 exploits the 'power of two choices' paradigm. Power of two choices states that by picking the minimum-degree node among two choices, the maximum in-degree $h_{m}=$ $\frac{\log \log n}{\log 2}(1+o(1))+\Theta(1)$ with high probability [212]. This is an exponential reduction in maximum degree compared to when $k=1$. More generally, for arbitrary $k>1$, the maximum degree is $\frac{\log \log n}{\log k}(1+o(1))+\Theta(1)$. This result is due to Azar et al. [212] and is well-studied in subsequent literature [213].

## REFERENCES

[1] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in SIGCOMM, 2010.
[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, pp. 63-74, 2008.
[3] D. Perino and M. Varvello, "A reality check for content centric networking," in Proceedings of the ACM SIGCOMM Workshop on InformationCentric Networking, 2011, pp. 44-49.
[4] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber, "Pushing CDN-ISP collaboration to the limit," ACM SIGCOMM Computer Communication Review, vol. 43, no. 3, pp. 34-44, 2013.
[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[6] R. Steinmetz and K. Wehrle, Peer-to-Peer Systems and Applications. Springer-Verlag Berlin Heidelberg, 2005, vol. 3485.
[7] "Data center efficiency assessment," Natural Resources Defense Council, Tech. Rep., 082014.
[8] A. Sharma and D. FitzGerald, "ABC's Oscars streaming outage shows web limitations for TV networks," Wall Street J, 2014.
[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the $A C M$, vol. 51, no. 1, pp. 107-113, 2008.
[10] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, pp. 379-423, 1948.
[11] T. M. Cover and J. A. Thomas, Elements of Information Theory. Wiley-Interscience, 2006.
[12] A. C.-C. Yao, "Some complexity questions related to distributive computing (preliminary report)," in Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing. ACM, 1979, pp. 209-213.
[13] E. Kushilevitz and N. Nisan, Communication Complexity. New York, NY, USA: Cambridge University Press, 1997.
[14] M. Braverman, A. Garg, D. Pankratov, and O. Weinstein, "From information to exact communication," in Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing. ACM, 2013, pp. 151-160.
[15] M. Braverman and A. Rao, "Information equals amortized communication," in FOCS, 2011, pp. 748-757.
[16] B. Barak, M. Braverman, X. Chen, and A. Rao, "How to compress interactive communication," SIAM Journal on Computing, vol. 42, no. 3, pp. 1327-1363, 2013.
[17] M. Braverman, "Interactive information complexity," in Proc. ACM Symposium on Theory of Computing Conference (STOC), 2012, pp. 505-524.
[18] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: A platform for high-performance internet applications," ACM SIGOPS Operating Systems Review, vol. 44, no. 3, pp. 2-19, 2010.
[19] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in Bitcoin P2P network," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014, pp. 15-29.
[20] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," SIGCOMM, 2013.
[21] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering Bitcoin's public topology and influential nodes," 2015. [Online]. Available: https://cs.umd.edu/projects/coinscope/coinscope.pdf
[22] H. Tyagi and S. Watanabe, "A bound for multiparty secret key agreement and implications for a problem of secure computing," in EUROCRYPT, 2014, pp. 369-386.
[23] H. Tyagi and S. Watanabe, "Converses for secret key agreement and secure computing," IEEE Transactions on Information Theory, vol. 61, no. 9, pp. 4809-4827, 2015.
[24] A. C. Yao, "Some complexity questions related to distributive computing," Proc. Annual Symposium on Theory of Computing, pp. 209-213, 1979.
[25] M. Karchmer and A. Wigderson, "Monotone circuits for connectivity require super-logarithmic depth," in Proc. Symposium on Theory of Computing (STOC), 1988, pp. 539-550.
[26] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in Proc. ACM Symposium on Theory of Computing (STOC), 1996, pp. 20-29.
[27] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," IEEE Transactions on Information Theory, vol. 19, no. 4, pp. 471-480, 1973.
[28] I. Csiszár and J. Körner, Information Theory: Coding Theorems for Discrete Memoryless Channels. Cambridge University Press, 2011.
[29] B. Barak, M. Braverman, X. Chen, and A. Rao, "How to compress interactive communication," in Proc. ACM Symposium on Theory of Computing (STOC), 2010, pp. 67-76.
[30] M. H. Yassaee, A. Gohari, and M. R. Aref, "Channel simulation via interactive communications," in Proc. IEEE Symposium on Information Theory (ISIT), 2012, pp. 1049-1053.
[31] N. Ma and P. Ishwar, "Some results on distributed source coding for interactive function computation," IEEE Transactions on Information Theory, vol. 57, no. 9, pp. 6180-6195, 2011.
[32] T. S. Han, Information-Spectrum Methods in Information Theory [English Translation]. Series: Stochastic Modelling and Applied Probability, Vol. 50, Springer, 2003.
[33] H. Tyagi, S. Venkatakrishnan, P. Viswanath, and S. Watanabe, "Information complexity density and simulation of protocols," in Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. ACM, 2016, pp. 381-391.
[34] H. Tyagi, S. B. Venkatakrishnan, P. Viswanath, and S. Watanabe, "Information complexity density and simulation of protocols," CoRR, vol. abs/1504.05666, 2015. [Online]. Available: http://arxiv.org/abs/1504.05666
[35] A. Ganor, G. Kol, and R. Raz, "Exponential separation of information and communication for Boolean functions," Electronic Colloquium on Computational Complexity (ECCC), vol. 21, p. 113, 2014.
[36] A. Ganor, G. Kol, and R. Raz, "Exponential separation of information and communication," in 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 1821, 2014, 2014, pp. 176-185.
[37] I. Csiszár and P. Narayan, "Secrecy capacities for multiterminal channel models," IEEE Transactions on Information Theory, vol. 54, no. 6, pp. 2437-2452, 2008.
[38] M. Madiman and P. Tetali, "Information inequalities for joint distributions, with interpretations and applications," IEEE Transactions on Information Theory, vol. 56, no. 6, pp. 2699-2713, 2010.
[39] U. M. Maurer, "Secret key agreement by public discussion from common information," IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 733-742, 1993.
[40] R. Ahlswede and I. Csiszár, "Common randomness in information theory and cryptography. I. Secret sharing," IEEE Transactions on Information Theory, vol. 39, no. 4, pp. 1121-1132, 1993.
[41] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer, "Generalized privacy amplification," IEEE Transactions on Information Theory, vol. 41, no. 6, pp. 1915-1923, 1995.
[42] R. Renner and S. Wolf, "Simple and tight bounds for information reconciliation and privacy amplification," in Proc. ASIACRYPT, 2005, pp. 199-216.
[43] J. M. Renes and R. Renner, "Noisy channel coding via privacy amplification and information reconciliation," IEEE Transactions on Information Theory, vol. 57, no. 11, pp. 7377-7385, 2011.
[44] J. Muramatsu, "Channel coding and lossy source coding using a generator of constrained random numbers," IEEE Transactions on Information Theory, vol. 60, no. 5, pp. 2667-2686, 2014.
[45] M. H. Yassaee, M. R. Aref, and A. Gohari, "Achievability proof via output statistics of random binning," IEEE Transactions on Information Theory, vol. 60, no. 11, pp. 6760-6786, 2014.
[46] T. S. Han and S. Verdú, "Approximation theory of output statistics," IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 752-772, 1993.
[47] P. Narayan, H. Tyagi, and S. Watanabe, "Common randomness for secure computing," in Information Theory (ISIT), 2015 IEEE International Symposium on. IEEE, 2015, pp. 949-953.
[48] D. Beaver, Perfect Privacy for Two-Party Protocols. Harvard University, Center for Research in Computing Technology, Aiken Computation Laboratory, 1989.
[49] E. Kushilevitz, "Privacy and communication complexity," SIAM Journal on Math, vol. 5, no. 2, pp. 273-284, 1992.
[50] S. Miyake and F. Kanaya, "Coding theorems on correlated general sources," IIEICE Trans. Fundamental, vol. E78-A, no. 9, pp. 10631070, September 1995.
[51] H. Tyagi, P. Viswanath, and S. Watanabe, "Interactive communication for data exchange," in Information Theory (ISIT), 2015 IEEE International Symposium on. IEEE, 2015, pp. 1806-1810.
[52] M. Feder and N. Shulman, "Source broadcasting with unknown amount of receiver side information," in ITW, Oct 2002, pp. 127-130.
[53] A. Orlitsky, "Worst-case interactive communication. I. Two messages are almost optimal," IEEE Transactions on Information Theory, vol. 36, no. 5, pp. 1111-1126, 1990.
[54] E.-H. Yang and D.-K. He, "Interactive encoding and decoding for one way learning: Near lossless recovery with side information at the decoder," IEEE Transactions on Information Theory, vol. 56, no. 4, pp. 1808-1824, 2010.
[55] M. Hayashi, H. Tyagi, and S. Watanabe, "Secret key agreement: General capacity and second-order asymptotics," arXiv:1411.0735, 2014.
[56] V. Strassen, "Asymptotic estimates in Shannon's information theory," in Proc. 3rd Trans. Prague Conf. Inf. Theory, 2009, pp. 689-723.
[57] M. Hayashi, "Information spectrum approach to second-order coding rate in channel coding," IEEE Transactions on Information Theory, vol. 55, no. 11, pp. 4947-4966, 2009.
[58] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," IEEE Transactions on Information Theory, vol. 56, no. 5, pp. 2307-2359, 2010.
[59] S. Arimoto, "On the converse to the coding theorem for discrete memoryless channels (corresp.)," IEEE Transactions on Information Theory, vol. 19, no. 3, pp. 357-359, 1973.
[60] G. Dueck and J. Korner, "Reliability function of a discrete memoryless channel at rates above capacity (corresp.)," IEEE Transactions on Information Theory, vol. 25, no. 1, pp. 82-85, 1979.
[61] Y. Polyanskiy and S. Verdú, "Arimoto channel coding converse and Rényi divergence," in Communication, Control, and Computing (Allerton), 201048 th Annual Allerton Conference on. IEEE, 2010, pp. 1327-1333.
[62] M. Braverman, A. Rao, O. Weinstein, and A. Yehudayoff, "Direct products in communication complexity," in FOCS, 2013, pp. 746-755.
[63] M. Braverman and O. Weinstein, "An interactive information odometer with applications," ECCC, 2014.
[64] R. Jain, A. Pereszlenyi, and P. Yao, "A direct product theorem for the two-party bounded-round public-coin communication complexity," in FOCS, 2012, pp. 167-176.
[65] A. Singla, A. Singh, and Y. Chen, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," in NSDI, 2012.
[66] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," SIGCOMM, 2011.
[67] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," SIGCOMM, 2011.
[68] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," 2009. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2009/08/master.pdf
[69] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror mirror on the ceiling: Flexible wireless links for data centers," SIGCOMM, 2012.
[70] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: A reconfigurable wireless data center fabric using free-space optics," in SIGCOMM, 2014.
[71] N. Farrington, Optics in Data Center Network Architecture. University of California, San Diego, 2012.
[72] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving $100 \%$ throughput in an input-queued switch," Communications, IEEE Transactions on, vol. 47, no. 8, pp. 1260-1267, 1999.
[73] I. Keslassy, C.-S. Chang, N. McKeown, and D.-S. Lee, "Optimal loadbalancing," in INFOCOM, 2005.
[74] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," Networking, IEEE/ACM Transactions on, vol. 7, no. 2, pp. 188-201, 1999.
[75] Z. Cao, M. Kodialam, and T. Lakshman, "Joint static and dynamic traffic scheduling in data center networks," in INFOCOM, 2014.
[76] X. Li and M. Hamdi, "On scheduling optical packet switches with reconfiguration delay," Selected Areas in Communications, IEEE Journal on, vol. 21, no. 7, pp. 1156-1164, 2003.
[77] V. Mirrokni, R. P. Leme, A. Vladu, and S. C.-w. Wong, "Tight bounds for approximate Carathéodory and beyond," arXiv preprint arXiv:1512.08602, 2015.
[78] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky et al., "Scheduling techniques for hybrid circuit/packet networks," in Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies. ACM, 2015, p. 41.
[79] S. Bojja Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate Carathéodory theorems," in Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science. ACM, 2016, pp. 75-88.
[80] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in INFOCOM, 2000.
[81] Y. Azar and I. Gamzu, "Efficient submodular function maximization under linear packing constraints," in Automata, Languages, and Programming. Springer, 2012, pp. 38-50.
[82] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," Journal of the ACM (JACM), vol. 36, no. 2, pp. 335-348, 1989.
[83] L. G. Valiant, "A bridging model for parallel computation," Communications of the ACM, vol. 33, no. 8, pp. 103-111, 1990.
[84] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a next generation data center architecture: Scalability and commoditization," in Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow. ACM, 2008, pp. 57-62.
[85] B. Prabhakar and N. McKeown, "On the speedup required for combined input-and output-queued switching," Automatica, vol. 35, no. 12, pp. 1909-1920, 1999.
[86] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, "Circuit switching under the radar with reactor," in NSDI, 2014.
[87] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," SIGCOMM, 2011.
[88] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in SIGCOMM, 2015.
[89] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in NSDI, vol. 10, 2010, pp. 19-19.
[90] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, "Load balanced Birkhoff-von Neumann switches," in High Performance Switching and Routing, 2001 IEEE Workshop on. IEEE, 2001, pp. 276-280.
[91] C.-H. Wang, T. Javidi, and G. Porter, "End-to-end scheduling for all-optical data centers," in Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015, pp. 406-414.
[92] Y. Li, S. Panwar, and H. J. Chao, "Frame-based matching algorithms for optical switches," in High Performance Switching and Routing, 2003, HPSR. Workshop on. IEEE, 2003, pp. 97-102.
[93] C.-H. Wang and T. Javidi, "Adaptive policies for scheduling with reconfiguration delay: An end-to-end solution for all-optical data centers," arXiv preprint arXiv:1511.03417, 2015.
[94] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," Communications, IEEE Transactions on, vol. 27, no. 10, pp. 14491455, 1979.
[95] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," Networking, IEEE/ACM Transactions on, vol. 11, no. 5, pp. 835-847, 2003.
[96] B. Wu and K. L. Yeung, "Nxg05-6: Minimum delay scheduling in scalable hybrid electronic/optical packet switches," in GLOBECOM, 2006.
[97] I. S. Gopal and C. K. Wong, "Minimizing the number of switchings in an SS/TDMA system," Communications, IEEE Transactions on, vol. 33, no. 6, pp. 497-501, 1985.
[98] S. Fu, B. Wu, X. Jiang, A. Pattavina, L. Zhang, and S. Xu, "Cost and delay tradeoff in three-stage switch architecture for data center networks," in HPSR, 2013.
[99] B. Wu, K. L. Yeung, and X. Wang, "Nxg06-4: Improving scheduling efficiency for high-speed routers with optical switch fabrics," in GLOBECOM, 2006.
[100] A. Dasylva and R. Srikant, "Optimal WDM schedules for optical star networks," IEEE/ACM Transactions on Networking (TON), vol. 7, no. 3, pp. 446-456, 1999.
[101] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," Selected Areas in Communications, IEEE Journal on, vol. 21, no. 4, pp. 546-559, 2003.
[102] A. Schrijver, Combinatorial Optimization - Polyhedra and Efficiency. Springer, 2003.
[103] S. Barman, "Approximating Nash equilibria and dense bipartite subgraphs via an approximate version of Caratheodory's theorem," in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing. ACM, 2015, pp. 361-369.
[104] A. Shieh, S. Kandula, A. G. Greenberg, and C. Kim, "Seawall: Performance isolation for cloud datacenter networks." in HotCloud, 2010.
[105] R. Duan and H.-H. Su, "A scaling algorithm for maximum weight matching in bipartite graphs," in SODA, 2012.
[106] R. Duan and S. Pettie, "Linear-time approximation for maximum weight matching," J. ACM, vol. 61, no. 1, pp. 1:1-1:23, Jan. 2014. [Online]. Available: http://doi.acm.org/10.1145/2529989
[107] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve $100 \%$ throughput in input-queued switches," in INFOCOM, 1998.
[108] S. Pettie and P. Sanders, "A simpler linear time 2/3- $\varepsilon$ approximation for maximum weight matching," Information Processing Letters, vol. 91, no. 6, pp. 271-276, 2004.
[109] P. F. Felzenszwalb and R. Zabih, "Dynamic programming and graph algorithms in computer vision," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 33, no. 4, pp. 721-740, 2011.
[110] R. Srikant and L. Ying, Communication Networks: An Optimization, Control and Stochastic Networks Perspective. New York, NY, USA: Cambridge University Press, 2014.
[111] R. Ahuja, T. Magnanti, and J. Orlin, Network flows: Theory, Algorithms, and Applications. Prentice Hall, 1993. [Online]. Available: https://books.google.com/books?id=WnZRAAAAMAAJ
[112] M. Grötschel, L. Lovász, and A. Schrijver, Geometric Algorithms and Combinatorial Optimization, ser. Algorithms and combinatorics. Springer-Verlag, 1993. [Online]. Available: https://books.google.com/books?id=agLvAAAAMAAJ
[113] C. Barnhart and Y. Sheffi, "A network-based primal-dual heuristic for the solution of multicommodity network flow problems," Transportation Science, vol. 27, no. 2, pp. 102-117, 1993.
[114] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas, "Fast approximation algorithms for multicommodity flow problems," Journal of Computer and System Sciences, vol. 50, no. 2, pp. 228-243, 1995.
[115] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," SIAM Journal on Computing, vol. 37, no. 2, pp. 630-652, 2007.
[116] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: A meta-algorithm and applications." Theory of Computing, vol. 8, no. 1, pp. 121-164, 2012.
[117] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," Journal of the ACM (JACM), vol. 34, no. 3, pp. 596-615, 1987.
[118] I. Keslassy, M. Kodialam, T. Lakshman, and D. Stiliadis, "On guaranteed smooth scheduling for input-queued switches," in INFOCOM, 2003.
[119] I. Keslassy, R. Zhang-Shen, and N. McKeown, "Maximum size matching is unstable for any packet switch," IEEE Communications Letters, vol. 7, no. 10, pp. 496-498, 2003.
[120] V. N. Padmanabhan and K. Sripanidkulchai, "The case for cooperative networking," in Revised Papers from the First International Workshop on Peer-to-Peer Systems IPTPS, 2001.
[121] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth content distribution in cooperative environments," in Peer-to-Peer Systems II. Springer, 2003, pp. 292-303.
[122] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-topeer streaming," in Network Protocols, 2003. Proceedings. 11th IEEE International Conference on. IEEE, 2003, pp. 16-27.
[123] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-topeer scheme for media streaming," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 2. IEEE, 2003, pp. 1283-1292.
[124] W. Zhang, Q. Zheng, H. Li, and F. Tian, "An overlay multicast protocol for live streaming and delay-guaranteed interactive media," Journal of Network and Computer Applications, vol. 35, no. 1, 2012.
[125] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in ACM SIGMETRICS Performance Evaluation Review. ACM, 2008.
[126] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou, "P2p streaming capacity under node degree bound," in Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on. IEEE, 2010, pp. 587-598.
[127] J. Zhu and B. Hajek, "Tree dynamics for peer-to-peer streaming," arXiv:1308.1971, 2013.
[128] J. Kim and R. Srikant, "Real-time peer-to-peer streaming over multiple random Hamiltonian cycles," Information Theory, IEEE Transactions on, vol. 59, no. 9, pp. 5763-5778, 2013.
[129] B. Doerr, T. Friedrich, and T. Sauerwald, "Quasirandom rumor spreading," in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, ser. SODA '08, 2008, pp. 773-781.
[130] G. Giakkoupis, "Tight bounds for rumor spreading in graphs of a given conductance," in 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011), vol. 9, 2011, pp. 57-68.
[131] K. Censor-Hillel, B. Haeupler, J. Kelner, and P. Maymounkov, "Global computation in a poorly connected world: Fast rumor spreading with no dependence on conductance," in Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, ser. STOC '12. ACM, 2012.
[132] B. Haeupler, "Simple, fast and deterministic gossip and rumor spreading." in SODA. SIAM, 2013.
[133] S. Bojja Venkatakrishnan and P. Viswanath, "Deterministic nearoptimal p2p streaming," in Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. ACM, 2015, pp. 451-452.
[134] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ser. SIGCOMM '01, 2001, pp. 149-160.
[135] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, A Scalable Content-Addressable Network. ACM, 2001, vol. 31, no. 4.
[136] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in Middleware 2001. Springer, 2001, pp. 329-350.
[137] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in Proceedings of IEEE INFOCOM, 2007.
[138] S. Sanghavi, B. Hajek, and L. Massoulie, "Gossiping with multiple messages," IEEE Transactions on Information Theory, 2007.
[139] T. Bonald, L. Massoulie, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: Optimal performance trade-offs," in Proceedings of ACM SIGMETRICS, Annapolis, MD, June 2008.
[140] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven meshbased streaming," IEEE/ACM Transactions on Networking (TON), vol. 17, no. 4, 2009.
[141] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in ACM SIGOPS Operating Systems Review, vol. 37, no. 5. ACM, 2003, pp. 282-297.
[142] B. Awerbuch and C. Scheideler, "The hyperring: A low-congestion deterministic data structure for distributed environments," in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2004.
[143] A. Bhargava, K. Kothapalli, C. Riley, C. Scheideler, and M. Thober, "Pagoda: A dynamic overlay network for routing, data management, and multicasting," in Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures. ACM, 2004, pp. 170-179.
[144] R. Jacob, A. Richa, C. Scheideler, S. Schmid, and H. Täubig, "A distributed polylogarithmic time algorithm for self-stabilizing skip graphs," in Proceedings of the 28th ACM Symposium on Principles of Distributed Computing. ACM, 2009, pp. 131-140.
[145] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," Journal of Scheduling, vol. 11, no. 2, pp. 105-120, 2008.
[146] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for p2p streaming systems," in INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE. IEEE, 2007, pp. 919-927.
[147] C. Law and K.-Y. Siu, "Distributed construction of random expander networks," in Proc. IEEE INFOCOM, 2003.
[148] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI'12, 2012.
[149] P. Babarczi, J. Tapolcai, L. Rónyai, and M. Médard, "Resilient flow decomposition of unicast connections with network coding," CoRR, vol. abs/1401.6670, 2014.
[150] "Cryptocurrency market capitalizations," 2016. [Online]. Available: https://coinmarketcap.com/
[151] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: Characterizing payments among men with no names," in Proceedings of the 2013 Conference on Internet Measurement. ACM, 2013, pp. 127-140.
[152] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in Bitcoin," in International Conference on Financial Cryptography and Data Security. Springer, 2013, pp. 34-51.
[153] S. B. Venkatakrishnan, G. Fanti, and P. Viswanath, "Dandelion: Redesigning the Bitcoin network for anonymity," arXiv:1701.04439, 2017.
[154] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical decentralized coin mixing for Bitcoin," in European Symposium on Research in Computer Security. Springer, 2014, pp. 345-364.
[155] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for Bitcoin with accountable mixes," in International Conference on Financial Cryptography and Data Security. Springer, 2014, pp. 486-504.
[156] P. Koshy, "Coinseer: A telescope into Bitcoin," Ph.D. dissertation, The Pennsylvania State University, 2013.
[157] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in International Conference on Financial Cryptography and Data Security. Springer, 2014, pp. 469-485.
[158] A. Biryukov and I. Pustogarov, "Bitcoin over Tor isn't a good idea," in 2015 IEEE Symposium on Security and Privacy. IEEE, 2015, pp. 122-134.
[159] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in IEEE P2P 2013 Proceedings. IEEE, 2013, pp. 1-10.
[160] M. Bartlett, "Deterministic and stochastic models for recurrent epidemics," in Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, vol. 4, no. 81, 1956, p. 109.
[161] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted Bitcoin-compatible anonymous payment hub," Cryptology ePrint Archive, Report 2016/575, Tech. Rep., 2016.
[162] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from Bitcoin," in Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013, pp. 397-411.
[163] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in 2014 IEEE Symposium on Security and Privacy. IEEE, 2014, pp. 459-474.
[164] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," Computer Networks, vol. 57, no. 2, pp. 378-403, 2013.
[165] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking Bitcoin: Large-scale network attacks on cryptocurrencies," arXiv preprint arXiv:1605.07524, 2016.
[166] D. Plohmann and E. Gerhards-Padilla, "Case study of the miner botnet," in 2012 4th International Conference on Cyber Conflict (CYCON 2012). IEEE, 2012, pp. 1-16.
[167] F. Sebastiani, "Machine learning in automated text categorization," ACM Computing Surveys (CSUR), vol. 34, no. 1, pp. 1-47, 2002.
[168] F. Reid and M. Harrigan, "An analysis of anonymity in the Bitcoin system," in Security and Privacy in Social Networks. Springer, 2013, pp. 197-223.
[169] M. Ober, S. Katzenbeisser, and K. Hamacher, "Structure and anonymity of the bitcoin transaction graph," Future Internet, vol. 5, no. 2, pp. 237-250, 2013.
[170] D. Ron and A. Shamir, "Quantitative analysis of the full Bitcoin transaction graph," in International Conference on Financial Cryptography and Data Security. Springer, 2013, pp. 6-24.
[171] "Bitcoin core integration/staging tree." [Online]. Available: https://github.com/bitcoin/bitcoin
[172] D. Shah and T. Zaman, "Rumors in a network: Who's the culprit?" Information Theory, IEEE Transactions on, vol. 57, pp. 5163-5181, Aug 2011.
[173] D. Shah and T. Zaman, "Rumor centrality: A universal source detector," in ACM SIGMETRICS Performance Evaluation Review, vol. 40. ACM, 2012, pp. 199-210.
[174] V. Fioriti and M. Chinnici, "Predicting the sources of an outbreak with a spectral technique," arXiv:1211.2333, 2012.
[175] Z. Wang, W. Dong, W. Zhang, and C. Tan, "Rumor source detection with multiple observations: Fundamental limits and algorithms," in ACM SIGMETRICS, 2014.
[176] A. Y. Lokhov, M. Mézard, H. Ohta, and L. Zdeborová, "Inferring the origin of an epidemic with dynamic message-passing algorithm," arXiv:1303.5315, 2013.
[177] B. A. Prakash, J. Vreeken, and C. Faloutsos, "Spotting culprits in epidemics: How many and which ones?" in $I C D M$, vol. 12, 2012, pp. 11-20.
[178] P. C. Pinto, P. Thiran, and M. Vetterli, "Locating the source of diffusion in large-scale networks," Physical Review Letters, vol. 109, no. 6, p. 068702, 2012.
[179] K. Zhu and L. Ying, "A robust information source estimator with sparse observations," arXiv:1309.4846, 2013.
[180] K. Zhu, Z. Chen, and L. Ying, "Locating contagion sources in networks with partial timestamps," arXiv:1412.4141, 2014.
[181] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," Journal of Cryptology, vol. 1, no. 1, 1988.
[182] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale." in OSDI, 2012, pp. 179182.
[183] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable anonymous group messaging," in CCS. ACM, 2010.
[184] P. Golle and A. Juels, "Dining cryptographers revisited," in Advances in Cryptology-Eurocrypt 2004, 2004.
[185] G. Fanti, P. Kairouz, S. Oh, and P. Viswanath, "Spy vs. spy: Rumor source obfuscation," in SIGMETRICS Perform. Eval. Rev., vol. 43, 2015, pp. 271-284.
[186] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The secondgeneration onion router," DTIC Document, Tech. Rep., 2004.
[187] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," ACM Transactions on Information and System Security (TISSEC), vol. 1, no. 1, pp. 66-92, 1998.
[188] "ZCash," https://z.cash/.
[189] T. Jedusor, "Mimblewimble," 2016. [Online]. Available: https://scalingbitcoin.org/papers/mimblewimble.txt
[190] G. Maxwell, "Coinjoin: Bitcoin privacy for the real world," in Post on Bitcoin Forum, 2013.
[191] M. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in Proc. CCS. ACM, 2002.
[192] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," 2011. [Online]. Available: https://csem.flinders.edu.au/research/techreps/SIE07001.pdf
[193] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," ACM Computing Surveys (CSUR), vol. 18, no. 1, pp. 23-38, 1986.
[194] M. Bayati, D. Shah, and M. Sharma, "Max-product for maximum weight matching: Convergence, correctness, and LP duality," IEEE Transactions on Information Theory, vol. 54, no. 3, pp. 1241-1251, 2008.
[195] M. R. Garey, D. S. Johnson, and R. E. Tarjan, "The planar Hamiltonian circuit problem is NP-complete," SIAM Journal on Computing, vol. 5, no. 4, pp. 704-714, 1976.
[196] D. Karger, R. Motwani, and G. Ramkumar, "On approximating the longest path in a graph," Algorithmica, vol. 18, no. 1, pp. 82-98, 1997.
[197] E. Levy, G. Louchard, and J. Petit, "A distributed algorithm to find hamiltonian cycles in $\mathrm{G}(\mathrm{n}, \mathrm{p})$ random graphs," in Workshop on Combinatorial and Algorithmic Aspects of Networking. Springer, 2004, pp. 63-74.
[198] "Confirmed transactions per day," 2016. [Online]. Available: https://blockchain.info/
[199] D. Danchev, "New underground service offers access to thousands of malware-infected hosts," 2013. [Online]. Available: https://www.webroot.com/blog/2013/02/12/new-underground-service-offers-access-to-thousands-of-malware-infected-hosts/
[200] Bitnodes, "Global bitcoin nodes distribution," 2016. [Online]. Available: https://bitnodes.21.co/
[201] C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," SIAM J. Comput., vol. 17, no. 2, pp. 210-229, 1988.
[202] R. Impagliazzo, L. A. Levin, and M. Luby, "Pseudo-random generation from one-way functions," in Proc. ACM Symposium on Theory of Computing (STOC), 1989, pp. 12-24.
[203] R. Renner, "Security of quantum key distribution," Ph.D. dissertation, ETH Zurich, 2005.
[204] A. Orlitsky and A. E. Gamal, "Communication with secrecy constraints," in Proc. ACM Symposium on Theory of Computing (STOC), 1984, pp. 217-224.
[205] I. Csiszár and P. Narayan, "Secrecy capacities for multiple terminals," IEEE Transactions on Information Theory, vol. 50, no. 12, pp. 30473061, 2004.
[206] H. Tyagi, "Common randomness principles of secrecy," Ph.D. dissertation, Univeristy of Maryland, College Park, 2013.
[207] V. Strassen, "The existence of probability measures with given marginals," Ann. Math. Statist., vol. 36, no. 2, pp. 423-439, 1965.
[208] S. Kuzuoka, "On the redundancy of variable-rate Slepian-Wolf coding," Proc. International Symposium on Information Theory and its Applications (ISITA), pp. 155-159, 2012.
[209] W. Feller, An Introduction to Probability Theory and Its Applications, Volume II. John Wiley \& Sons Inc., UK, 1971.
[210] E. Makover and J. McGowan, "Regular trees in random regular graphs," arXiv preprint math/0610858, 2006.
[211] N. L. Johnson and S. Kotz, Urn Models and Their Application; An Approach to Modern Discrete Probability Theory. New York, NY (USA) Wiley, 1977.
[212] Y. Azar, A. Broder, A. Karlin, and E. Upfal, "Balanced allocations," SIAM Journal of Computing, vol. 29, no. 1, pp. 180-200, 1999.
[213] M. Mitzenmacher, "The power of two choices in randomized load balancing," IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 10, pp. 1094-1104, 2001.


[^0]:    ${ }^{1}$ Or indeed even across intersections of multiple disciplines.

[^1]:    ${ }^{1}$ For brevity, we do not display the dependence of $\operatorname{IC}(\pi)$ on the (fixed) distribution $\mathrm{P}_{X Y}$.

[^2]:    ${ }^{2}$ Braverman and Rao [15] actually used their general simulation protocol as a tool for deriving the amortized distributional communication complexity of function computation. This result was obtained independently by Ma and Ishwar in [31] using standard information theoretic techniques.

[^3]:    ${ }^{3}$ Formally, our lower bound uses lower $\varepsilon$-tail $\sup \{\lambda: \operatorname{Pr}(\mathrm{ic}(\Pi ; X, Y)>\lambda)>\varepsilon\}$ and the upper bound uses upper $\varepsilon$-tail $\inf \{\lambda: \operatorname{Pr}(i c(\Pi ; X, Y)>\lambda)<\varepsilon\}$. For many interesting cases, the two coincide.

[^4]:    ${ }^{4}$ The proof in [15] uses the inequality $\operatorname{IC}(\pi) \leq\|\pi\|$, a multiparty extension of which is available in [37, 38].

[^5]:    ${ }^{5}$ The spectrum slicing technique was introduced in [32] to derive the error exponents of various problems for general sources and a rate-distortion function for general sources.
    ${ }^{6}$ The random variables $U, U_{\mathcal{X}}, U_{\mathcal{Y}}$ are mutually independent and independent jointly of $(X, Y)$.

[^6]:    ${ }^{7}$ We allow $\Pi_{i}$ to be constant and allow it to depend only on the local observation (and not on the previous communication $\Pi_{1}, \ldots, \Pi_{i-1}$ ). This description of an interactive protocol is very general and is equivalent to the usual protocol-tree based description ( $c f$. [29, 15]).
    ${ }^{8}$ Since we are not interested in minimizing the amount of randomness used in a simulation, and private randomness can always be sampled from public randomness, we can restrict ourselves to public protocols for simulating.

[^7]:    ${ }^{9}$ The lower bound in [15] gives only the weak converse which holds only when $\varepsilon=\varepsilon_{n} \rightarrow$ 0 as $n \rightarrow \infty$.

[^8]:    ${ }^{10}$ The result in $[62,63]$ shows a direct product theorem when we communicate less than $n \operatorname{IC}(f) / \operatorname{poly}(\log n)$.

[^9]:    ${ }^{1}$ Servers with 10 Gbps network interfaces are common today and 40/100 Gbps servers are being deployed.

[^10]:    ${ }^{2}$ Designs based on point-to-point wireless links have also been proposed [68, 69]. Our abstract model is general.

[^11]:    ${ }^{3}$ Our work is orthogonal to how the controller obtains the traffic demand estimate. For example, the nodes could simply report their backlogs before each scheduling window, or a more sophisticated prediction algorithm could be used.

[^12]:    ${ }^{4}$ Indirect routing in a distributed setting but without consideration of switch reconfiguration delay was studied in a recent work [75].

[^13]:    ${ }^{1}$ For ease of notation, we will drop the argument $t$ from $G_{i}(t), V(t), E_{i}(t)$ and denote them simply by $G_{i}, V_{i}, E_{i}$ respectively, with the time aspect implicitly understood.

[^14]:    ${ }^{2}$ The exact topology of the cluster is irrelevant to the functioning of our algorithm, as long as the peers are able to learn the state of the other peers sufficiently fast.

[^15]:    ${ }^{3}$ We contrast this with our algorithm in Section 4.3 which operates at every time-slot.

[^16]:    ${ }^{1}$ In particular, we assume adversaries do not repeatedly exit and re-join the network in order to achieve a favorable graph configuration.

[^17]:    ${ }^{2}$ Following convention we define $D_{\mathrm{M}}(v)=0$ if both the numerator and denominator are 0 in Equation (5.2).

[^18]:    ${ }^{3}$ We use lowercase "dandelion" to denote the spreading protocol, and uppercase "DANDELION" to denote the overall network-level algorithm.

[^19]:    ${ }^{1}$ There is no agreement over the definition conditional min-entropy; the form adopted here is convenient for our use.

[^20]:    ${ }^{2}$ The $N$-bit ACK-NACK feedback used in the protocol can be determined from the length of the transcript.

[^21]:    ${ }^{3}$ Csiszár and Narayan considered a multiterminal version of the data exchange problem in [205] and connected the minimum (amortized) rate of communication needed to the maximum (amortized) secret key rate.

[^22]:    ${ }^{4}$ For clarity, we display the dependence of each information density on the underlying distribution in the remainder of this section.

[^23]:    ${ }^{5}$ When $h_{\mathrm{Q}_{X \mid Y}}(X \mid Y)<\lambda_{\mathrm{Q}_{X \mid Y}}^{\min }$, Protocol 11 may transmit more than $\left(h_{\mathrm{Q}_{X \mid Y}}(X \mid Y)+\right.$ $\left.\Delta_{\mathrm{Q}_{X \mid Y}}+\gamma+N_{\mathrm{Q}_{X \mid Y}}\right)$ bits.

[^24]:    ${ }^{6}$ When the protocol terminate before $N_{\mathrm{Q}_{\Pi_{1} \mid Y}}$ th round, a part of $\left(f\left(\Pi_{1}\right), f^{\prime}\left(\Pi_{1}\right)\right)$ may not be sent.

[^25]:    ${ }^{7}$ We introduce $Z$ as a placeholder for $X$ or $Y$ for brevity.
    ${ }^{8}$ Although the constant depends on random variables appearing in each round, since the number of rounds is bounded, we take the maximum constant so that Equation (A.27) holds for every $t$.

[^26]:    ${ }^{1}$ If no such $d_{l}$ exists, then $d_{l}=0$.

[^27]:    ${ }^{1}$ In case of ties, we repeat the above process to each of the servers until the maximum server degree reduces by one.

