Applications of Coding in Network Communications

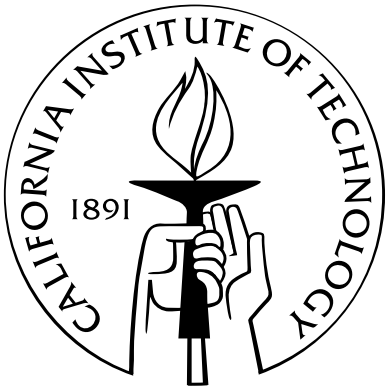
Thesis by

Christopher SungWook Chang

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2012

(Defended May 29th, 2012)

To My Mother, Father, and Grandparents

The Truth is realized in an instant; the Act is practiced step by step.

— Zen Master Seung Sahn

Acknowledgments

My Korean name “Sung-Wook Chang” means by characters “accomplishment”, “shine”, and “benevolence”, respectively. Thanks to many people around me, my doctoral thesis could be completed with shiny memories and learning. It is my pleasure to express my sincere gratitude to everyone and now it is time to contribute in the world!

First of all, it is my honor and luck to have studied with my two advisers, Professor Tracey Ho and Professor Michelle Effros. I can never appreciate them enough. As academic advisers, mentors, and friends, they guided, supported, and encouraged me throughout the journey for my doctorate at Caltech. I owed an inestimable debt to them for my professional and personal development. Professor Ho shared her great intellectual assets with me and led me to solve seemingly impossible problems. Professor Effros provided the big picture and deep insight on the problem directions. By virtue of their well-balanced roles, I could achieve pleasant accomplishments without falling.

I also thank my thesis committee, Professor Robert J. McEliece, Professor Babak Hassibi, Professor Katrina Ligett, Professor Jehoshua Bruck, and Professor Steven Low for their support and help during my graduate study.

I gratefully appreciate Professor YongHwan Lee, Professor NamKyu Park, Professor WonChan Kim, Professor HyuckJae Lee, Professor SungHyun Choi, and Professor MinKoo Han. From their inspiring guidance and teachings, I was first enchanted by the field of electrical engineering. During my undergraduate and even during my graduate studies, they gave continuous encouragement to help me realize my full potential.

I thank my collaborators (Professor Muriel Médard, Dr. Matthew Klimesh, Dr. Jon Hamkins, Dr. Xinzhou Wu, Dr. Sundar Subramanian, Professor Ben Leong, and Dr. Brian DeCleene) and lab mates (Dr. WeiHsin Gu, Dr. Mayank Bakshi, Derek Leong, Dr. Tao Cui, Theodore Dikaliotis, Dr. Svitlana Vyetrenko, Dr. HongYi Yao, and MingFai Wong). It was wonderful experience to share considerable intellect with them. I appreciate Shirley

Slattery, Tanya Owen, and Linda Dozsa for their perfect administrative support. I express my joyful gratitude to Ernie. I am sure I had lunch from his kitchen at least half of my Caltech life. “Muchas gracias, Ernesto!”

My very special thanks go to the most precious earning in my life: my friends, the essential ingredient of my life. First, I express tremendous gratitude and love to my SSHS friends, HyungIl Chae, Dr. BongSu Kyung, SeungHyuck Hong, HyoungJun Ahn, MyungHwan Kim, JaeHoo Lee, DongHoon Yi, TaeJoon Suk, GyuWeon Hwang, JaeWon Shim, HakYun Lee (above a.k.a. Oshiba family), Dr. SungWhan Oh, DongWoo Kang, Mo Lee, Lin Han, and JaeKyung Suh. I thank my soul brothers who always inspire me and boost my spirits up, Dr. SangHyun Chang, Dr. KunHo Roh, Professor DonHee Ham, Dr. SungWon Lee, Dr. JuYong Kim, Dr. JinSung Heo, Dr. WonTae Joe, Professor JongSub Lee, SeungHyun Son, DooHyun Kim, Dr. Joohnyup Kim, Dr. JeeHwan Kim, Dr. JaeSun Seo, and Dr. HoJun Kim. My work and life at Caltech has been joyfully filled with great buddies, Dr. ChangHo Sohn, Dr. SukWon Kim, Dr. OhHoon Kwon, Dr. JinHwan Lee, JiHoon Kim, KunWoo Kim, ShinChul Yeom, SangHee Park, KiYeol Yang, JungWoo Kim, and SeungAh Lee. I can’t miss my beloved DGM crews out, JaeMin Park, JinSoo Park, and KiWon Kim, who adorned my glorious life in the city of angels. Regarding my spiritual perfection, Dahl Mah Sah temple has played the best role. Thank you, BumRak Sohn, DongWoo Sohn, and all Shin Do Nims.

I gratefully acknowledge financial support from Caltech Lee Center of Advanced Networking, NSF, DARPA, BAE systems, Samsung Scholarship Foundation, Korea Foundation for Advanced Studies, NASA-JPL, and KUSCO-KSEA.

Last but most important acknowledgment goes to my mother YounSoon Yang, my father YoungHoon Chang, my grandmother SangYong Hwang, my late grandmother SoonJong Lee, my late grandfather BumJin Chang, and my late grandfather DongWun Yang. Without these greatest minds, I would not exist in this blessed world. I dedicate this thesis to their unconditional love, endless encouragement, warmest guidance, firmest support, and strongest belief with my deepest gratitude. I love you so much.

Abstract

This thesis uses the tool of network coding to investigate fast peer-to-peer file distribution, anonymous communication, robust network construction under uncertainty, and prioritized transmission.

In a peer-to-peer file distribution system, we use a linear optimization approach to show that the network coding framework significantly simplifies analysis even in scenarios where the optimal solution does not require coding. We also study the effect of requiring reciprocity and the impact of dynamically changing network scenarios.

Second, we investigate anonymous routing in peer-to-peer networks. The goal is to design and analyze a peer-to-peer system that hides the identities of source and sink pairs against adversarial nodes. We first propose a protocol for subgraph construction signaling. The protocol uses path diversity rather than cryptographic keys. We prove information theoretic security of the proposed protocol. We investigate a variety of deterministic and randomized subgraph designs. We also give a reverse path construction mechanism, with which a sink can reply to the source without knowing the source identity. We next investigate anonymous data transmission using network coding. Again, path diversity (with network coding) is used to hide the identities of source and sink pairs. We investigate the effect of subgraph shape on anonymity and congestion arising from traffic shaping constraints, demonstrating the tradeoff between the two through simulations.

Third, we study the problem of network construction under uncertainty about link-loss rates. We prove that both maximizing throughput and minimizing cost are coNP-hard problems. We find polynomial time-solvable solutions that outperform other deterministic approaches.

Lastly, we investigate strategies for communication under a system that prioritizes data based on the worth of data and the probability of successful transmission. Only the highest priority data is transmitted when communication is very limited.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis studies network communications from a network coding and information theory viewpoint. The consideration of coding in these network problems provides a variety of advantages in terms of performance in unreliable networks, reduced complexity of analysis, and reduced complexity of implementation.

Two of the four problems we study in this thesis concern peer-to-peer networks. The peer-to-peer architecture is an alternative to the traditional server-client architecture [71]. In the server-client model, multiple clients are connected to the server (service provider) that stores and distributes the data. Accordingly, the server is required to have large capacity, large storage, and high processing speed, all of which are costly. In contrast, the peer-to-peer system distributes networking load over multiple peer computers on a network. Instead of a single high-performance server, multiple lower-performance peers cooperate to distribute content among themselves. This model provides good scalability in terms of network resources, but its distributed nature gives rise to new challenges in optimization, control, and analysis. We study both the limits of download times in peer-to-peer content distribution networks and techniques to establish sender- and receiver-anonymity in communication over peer-to-peer networks.

The other two problems we investigate concern unreliable networks, where coding is useful for providing robustness. In many wireless networks, factors such as mobility of users, finite battery life, and multipath fading lead to variability in, and therefore uncertainty about, the network. This variability and uncertainty together pose a variety of challenges for establishing robust communications and for supporting prioritization over competing communication interests when it is not possible to simultaneously provide the highest level of service to all. We consider robust construction of network coding subgraphs under un-

certainty about loss rates on links. We also study coding for prioritized transmission over lossy links.

1.1 Contributions and thesis outline

In Chapter 2, we consider a linear optimization approach for studying download finish times in peer-to-peer networks that allow but do not require coding. We characterize the structure of the optimal solution in terms of the number of required phases (time intervals during which the rate of flow between each pair of peers remains unchanged). Using the coding framework, we provide a counterexample disproving the conjecture in Ezovski *et al.* [27] that in the absence of coding, the sequential minimization of file download times minimizes the average finish time over all users. Our results demonstrate that using the network coding framework simplifies analysis even in scenarios where the optimal solution does not require coding. We also use this framework to study the effect of reciprocity constraints, which are commonly used in incentive-compatible peer-to-peer protocols. Lastly, we show that for a dynamically changing network scenario, coding can provide a robust and optimal solution that outperforms routing.

Chapter 3 investigates anonymous routing in a peer-to-peer network. Each source constructs a subgraph for anonymous communication to a sink node using a randomly chosen subset of available nodes, some of which may be adversarial and collude to try to identify the origin and destination. In the first part, we investigate the subgraph construction phase. We consider an entropy metric for anonymity, and suggest a protocol with a formal information theoretic security[1] characterization, which relies on path diversity rather than a public key infrastructure (PKI). We present performance analysis and optimization of subgraph shape and parameters with respect to this metric, and suggest a computationally efficient heuristic for determining subgraph shape. Furthermore, to improve the anonymity of the system, we investigate randomization of subgraph parameters. We also consider a reverse path construction for a feedback message and show that the reverse path does not reveal any information of source and sink identities.

In the second part of Chapter 3, we study the anonymous data transmission phase using network coding over a subgraph. Each node performs conventional linear network coding

[1]Information theoretic security means that the security does not depend on computational hardness assumptions.

operations across packets, as well as linear coding operations within packets making it difficult for the adversary to correlate the contents of different data packets in the subgraph. This has lower complexity than hop-by-hop cryptographic transformation used in existing anonymity schemes to prevent content correlation. By parameterizing the subgraph shape as well as the network connectivity, we investigate their effect on anonymity and networking performance.

Chapter 4 considers the problem of network coding subgraph construction under uncertainty about link loss rates. For a given set of scenarios specified by an uncertainty set of link loss rates, we provide a robust optimization-based formulation to construct a single subgraph that would work relatively well across all scenarios. We show the general problem is coNP-hard for the min-cost and max-throughput objectives. To make the problem tractable, we modify the problem by adding path constraints, which results in a polynomial time-solvable solution in terms of the problem size. Simulation results for simple network examples showing that this solution is better and more stable than the non-robust optimization solution in terms of worst-case performance.

Chapter 5 considers prioritized communication in a network where there are multiple paths from a sender to a destination and the availability of individual links is uncertain. Scenarios are considered in which the goal is to maximize a payoff that assigns weight based on the worth of data and the probability of successful transmission. Ideally, the choice of what information to send over the various links will provide protection of high value data when many links are unavailable, yet result in communication of significant additional data when most links are available. Here the focus is on the simple network of multiple parallel links, where the links have known capacities and outage probabilities. Given a set of simple inter-link codes, we propose a linear programming approach to find the optimal timesharing strategy among these codes. In the case of unit capacities and unit message size case, some observations are made about the problem of determining all potentially useful codes, and techniques (e.g., using matroid theory) to assist in such determination are presented.

Chapter 6 summarizes our investigations and concludes the thesis with discussion.

1.2 Background on Network Coding

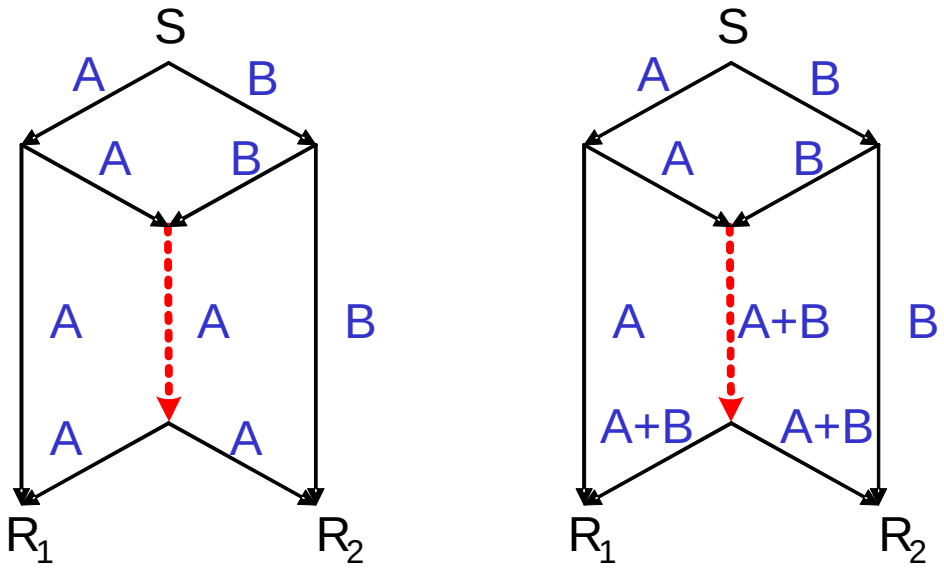
In this section, we provide some background on network coding. briefly describing the basic principle and some important benefits.

Network coding has recently been shown to improve performance over both wired and wireless communication networks [1, 46], see [39, 54, 81] and references therein. Network coding provides benefits in terms of throughput, robustness, security, etc. In addition, as described in Chapter 2, it can provide a computationally tractable solution of a problem that is computationally challenging (e.g., NP-hard) for the case of using conventional routing. In this section, we consider some benefits of network coding.

In multicast[2] communications, previous researchers [48, 50] have shown that linear network coding can achieve the maximum throughput of a given network that is equal to the maximum flow between the source and each destination, using either deterministic [41] or randomized [40] code construction algorithms. Ho *et al.* showed that random linear network coding—linear combination of packets with random coefficients from some field $\mathbb{F}_q$—can achieve the network capacity with probability exponentially approaching 1 if the field size is sufficiently large [40]. Fig 1.1 illustrates a throughput gain of network coding in a wired network. Since each link has unit capacity, the maximum flow between the source and each destination is 2. As shown in Fig 1.1b, network coding can achieve multicast capacity 2 by the simple XOR operation on the bottleneck link. In contrast, routing achieves strictly less capacity due to the uncoded packet on the bottleneck link (Fig 1.1a). In contrast to the directed networks, Li *et al.* bounded the throughput gain of network coding for unicast, broadcast (no gain for both), and multicast (factor of 2) scenarios in undirected networks, where each communication link is bidirectional [51].

Wireless networks possess some properties that make them different from wired networks. On one hand, their inherent nature (e.g., broadcast transmission, medium-sharing, and time-varying environments) provides more opportunities to benefit from network coding. On the other hand, it is more challenging to characterize and achieve the optimal strategy for wireless networks. In [46], it is shown that network coding is beneficial for a few topologies of wireless networks, and more interestingly, the interaction between the

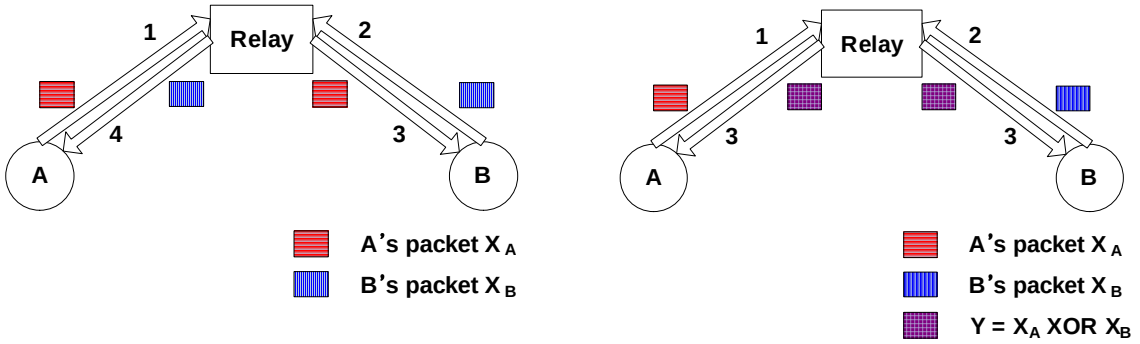
[2]Multicast means the delivery of information from one source node to a specified set of destination nodes, all of which require the same information. In the unicast scenario, there is one destination node. In the broadcast scenario, all participating nodes in the network are destination nodes requiring the same information.



(a) Butterfly network with routing

(b) Butterfly network with coding

Figure 1.1: The seminal butterfly network. Source node S sends 2 unit data A and B to the two destinations R_1 and R_2. Each link has unit capacity. The dashed links in red are bottleneck links. (a) For the routing case, the bottleneck link can transmit either A or B. In either case, one destination receives 2 units of message but the other receives 1 unit of message. (b) For the coding case, the bottleneck link can transmit coded packet A XOR B (denoted by $A + B$), and the two destination nodes can decode both A and B. Therefore, the coding gain is $\frac{2}{1.5} = \frac{4}{3}$.



(a) Wireless-Routing (b) Wireless-Coding

Figure 1.2: A canonical example of how network coding increases the throughput in a wireless network: node A and node B want to exchange messages. (a) Without network coding, 4 transmissions are needed. (b) With network coding, only 3 transmissions are needed: once the relay node receives packets from A and B, it broadcasts $Y = (X_A \text{ XOR } X_B)$ to all. Then, each node can decode both X_A and X_B by XORing the received packet Y with the packet it already has.

physical and MAC layer produces extra benefits. Let's consider a basic scenario of data exchange through a two-way relay channel (Fig. 1.2). We observe that the coding gain in terms of the number of transmissions is 4/3, and furthermore, the network coding gain with interaction between the physical and MAC layer (called CODING+MAC gain) is 2. In Fig. 1.2, the CODING+MAC gain can be understood as follows: the MAC divides the bandwidth equally among the 3 nodes to be fair. However, the relay nodes needs to transmit twice as many packets as node A or node B without network coding, which results in a bottleneck at the relay. In contrast, network coding allows the relay to XOR pairs of packets and drain the packets twice as fast (i.e., there is no bottleneck, and the throughput is doubled).

In addition to network throughput benefits, network coding can provide robustness to uncertain communication link status and uncertain network topology changes [13,14,30]. If we cannot predict the link condition (unknown link loss probability) or the topology changes (in dynamical scenario) a priori, the performance of the routing scheme with non-coded packets significantly depends on the realization of the uncertain parameters. In contrast, with random linear network coding, each transmitted packet is a mixture of packets. Therefore, even after the realization of uncertain parameters, the previously transmitted packets can be still useful.

Lastly, network coding can be a useful tool to find a solution that achieves the op-

timal system performance. For some problems where it is known to be computationally challenging to find an optimal routing solution, we can find an optimal coding solution in polynomial time. For example, in a peer-to-peer file distribution system, finding an optimal strategy for routing is demanding due to the large system size that makes it challenging to track data identities of all packets. In contrast, if we use network coding, random linear combinations can mitigate this difficulty since tracking data identity is not necessary; only the data amount matters. Examples in Chapter 2 illustrate this benefit. Another example is that routing in a BitTorrent-like peer-to-peer network usually suffers from the "missing the last packet" problem, whereas random linear network coding is free from this issue [34, 35, 42, 65].

Chapter 2

Peer-to-Peer File Distribution System

In this chapter we consider a linear optimization approach for studying download finish times in peer-to-peer networks that allow but do not require coding. We demonstrate that using the network coding framework simplifies analysis even in scenarios where the optimal solution does not require coding. We also use this framework to study the effect of requiring reciprocity, a typical feature of incentive-compatible protocols. Lastly, we show that for a dynamically changing network scenario, coding can provide a robust and optimal solution that outperforms routing.

2.1 Introduction

Peer-to-peer (P2P) file distribution algorithms are an active field of research in both academic [65] and industrial [19] settings. P2P algorithms are desirable for their scalability—allowing efficient and inexpensive file distribution from a single content provider (server) to thousands of users (Fig. 2.1). P2P systems achieve these benefits by exploiting the bandwidth of the peers.

While network coding has been applied to P2P systems to improve robustness and maximize throughput [35], the performance gain of network coding over routing in a P2P system remains a topic for further research. In [22], Deb *et al.* consider the dissemination of multiple messages using a gossip-based protocol, showing that in an n-node network, network coding speeds message dissemination from time $\Theta(n \log(n))$ for uncoded schemes

The work in this chapter was presented at the 6th IEEE International Symposium on Network Coding (NetCod) [14].

to time $O(n)$ for random linear coding. In [58], Mundinger and Weber introduce an uplink sharing model that assumes a fully connected network where each peer is constrained only in its upload capacity. In [18], Chiu *et al.* show that network coding does not increase multicast throughput in this scenario. In [55], Mehyar *et al.* investigate a few small networks of this type, studying optimal strategies for minimizing (a) the finish time of the last peer; (b) the average finish time over all users; and (c) the Min-Min finish time, which sequentially minimizes the finish time of the remaining peer with the highest upload capacity until all peers finish their downloads. In [27], Ezovski *et al.* present an optimal routing solution for minimizing the Min-Min finish time in the uplink sharing model. They further claim that following the Min-Min strategy minimizes the average finish time over all routing strategies.

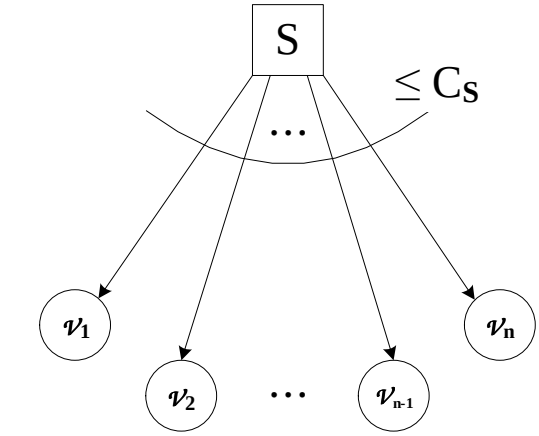
In Section 2.4, we use a counterexample to disprove Ezovski *et al.*'s claim that the Min-Min strategy minimizes the average finish time. Like [27, 55], we assume that all peers stay in the system after completing their own downloads. We derive our counterexample using the linear programming approach of Wu *et al.* from [80]. We also extend the LP to study the effect of a reciprocity constraint. Finally, we show that coding can improve robustness to unexpected network changes.

Our investigations underscore two benefits of the network coding framework. First, the framework makes the problem of finding optimal solutions tractable; even when routing suffices to obtain the optimal performance, finding the optimal routing solution is often an intractable problem. Second, coding can provide robustness in dynamically changing network scenarios.

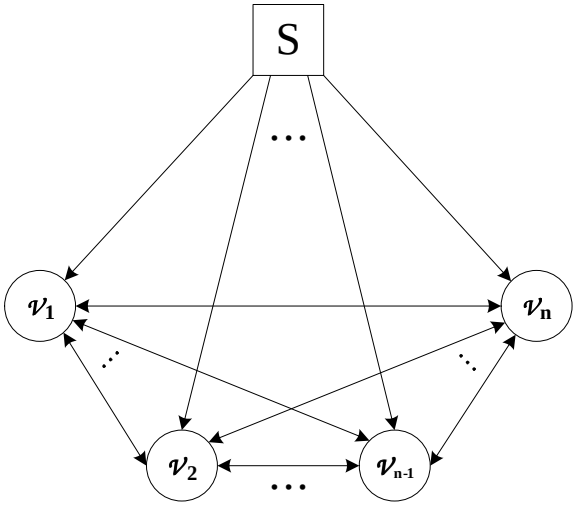
2.2 Preliminaries

We use the uplink sharing model of [58]. The network is fully connected, and the upload capacity of each node (including all peers and the server) is initially the only constraint. We discuss the system performance in terms of download finish times when there is a single server with a finite file to distribute to multiple peers. We consider the following performance metrics:

1. *Min-Min Finish Time:* The Min-Min finish time strategy sequentially minimizes the finish time of the remaining node with the highest capacity. Precisely, let T_i be the finish time for the i^{th} node when nodes are ordered from largest to smallest upload



(a) Server-Client Model



(b) Peer-to-Peer Model

Figure 2.1: (a) In the server-client model, server overhead can be significant for large number of peers. Downloads is slow due to the server's upload capacity constraint. (b) In the peer-to-peer model, each node uses its own upload capacity to aid file distribution. Therefore, it improves the server overhead problem and download speed.

capacity. Then the Min-Min strategy is the strategy that achieves

$$\begin{aligned} T_1^* &\triangleq \min T_1 = \frac{\text{(File Size)}}{\text{(Server Upload Capacity)}} \\ T_i^* &\triangleq \min\{T_i | T_j = T_j^*, \forall j < i\} \end{aligned} \tag{2.1}$$

The optimization is over all possible routing or coding schedules that satisfy the nodes' uplink capacity constraints.

2. *Min-Avg Finish Time:* The Min-Avg finish time strategy minimizes the average finish time over all users. If peers stay in the network until all downloads are completed, then peers finish in the order of highest to lowest upload capacity as in the Min-Min strategy.

3. *Min-Max Finish Time:* Another performance metric of interest is Min-Max. The goal of Min-Max finish time is to minimize the last (slowest) peer's finish time.

Both metrics can be applied either with or without reciprocity constraints.

2.3 Linear Programming Formulation

2.3.1 Problem Formulation

Let v_0 and $v_1, \ldots, v_m$ denote the server and peers, respectively, in a single-server, m-peer P2P network. Node $v \in \{v_0, \ldots, v_m\}$ has uplink capacity $c(v)$. All peers remain in the network after finishing their downloads. It is therefore always optimal for higher-capacity peers to finish earlier than lower-capacity peers since their greater upload capacity makes them more useful for serving other peers. We therefore order the peers from highest to lowest upload capacity giving $c(v_1) \geq c(v_2) \geq \cdots \geq c(v_m)$. We describe each solution for distributing a file of size F from the server to the peers by describing a sequence of phases (Fig. 2.2). Each phase is a period in which the upload strategies of all nodes are fixed—that is, in phase τ each node $v \in \{v_0, \ldots, v_m\}$ allocates its upload capacity according to some fixed flow vector describing the proportion of node v's upload capacity used to upload data to each of the users in $\{v_1, \ldots, v_m\} \setminus \{v\}$. The duration of phase τ equals the maximum over nodes $v \in \{v_0, \ldots, v_m\}$ of the total flow from node v in phase τ divided by the uplink capacity of node v. To make this precise, we represent a full solution with I phases by

the following time-expanded graph. Let $\mathcal{V} = \{v_0^{(\tau)}, v_1^{(\tau)}, \ldots, v_m^{(\tau)}\}_{\tau=1}^{I}$, where $v_i^{(\tau)}$ represents node v_i in phase $\tau \in \mathcal{I} \triangleq \{1, \ldots, I\}$. Let $\mathcal{E}$ denote the set of edges in the time-expanded graph. Set $\mathcal{E}$ contains two types of edges:

- Transmission edge $e = (v_i^{(\tau)}, v_{i'}^{(\tau)})$ corresponds to the transmission from v_i to $v_{i'}$ within the τ^{th} phase.
- Memory edge $e = (v_i^{(\tau)}, v_i^{(\tau+1)})$ corresponds to the accumulation of received information from previous time steps. Memory edges have infinite capacities.

Each transmission edge exists within a single phase. Each memory edge crosses from one phase to the next. As in [27, 55], we assume continuous data flow and allow each node to forward data immediately upon receipt.[1] Further, each node can transmit data to multiple nodes simultaneously.

Since there always exists an optimal strategy with $I \leq m$ (Theorem 2.1 in Section 2.3.3), we set $I = m$ and treat $v_\tau^{(\tau)}$, $\tau \in \{1, \ldots, m\}$ as the sink nodes of the time-expanded graph. Let $\mathbf{t} = (t_1, \ldots, t_m)$ denote the vector of phase durations. Peer j finishes its download in the j^{th} phase, so its finish time is $T_j = \sum_{k=1}^{j} t_k$. Both the Min-Avg and Min-Min objective functions can be described as linear functions of vector $\mathbf{t}$. For Min-Avg, let $\mathbf{d} = \left[\frac{m}{m}, \frac{m-1}{m}, \ldots, \frac{1}{m}\right]$. Then the Min-Avg objective function is

$$\mathbf{d}^T\mathbf{t} = \frac{1}{m}\sum_{j=1}^{m}\left[\sum_{k=1}^{j} t_k\right].$$

To represent Min-Min as a linear function of $\mathbf{t}$, we introduce a weighting vector ω such that $\omega_1 \gg \omega_2 \gg \ldots \gg \omega_m$ and set $\mathbf{d} = \left[\frac{\omega_1+\cdots+\omega_m}{m}, \frac{\omega_2+\cdots+\omega_m}{m}, \ldots, \frac{\omega_m}{m}\right]$. The Min-Min objective function is

$$\mathbf{d}^T\mathbf{t} = \frac{1}{m}\sum_{j=1}^{m}\omega_j\left[\sum_{k=1}^{j} t_k\right].$$

The objective function for Min-Max is simply the finish time of the last (slowest) peer.

$$\mathbf{d}^T\mathbf{t} = \sum_{i=1}^{m} t_i$$

[1] This simplifying assumption is not realistic in practice since nodes typically cannot send out data until they receive at least a block of a certain size. As a result, optimal download times achieved using this model give lower bounds on the download times that can be achieved in practice.

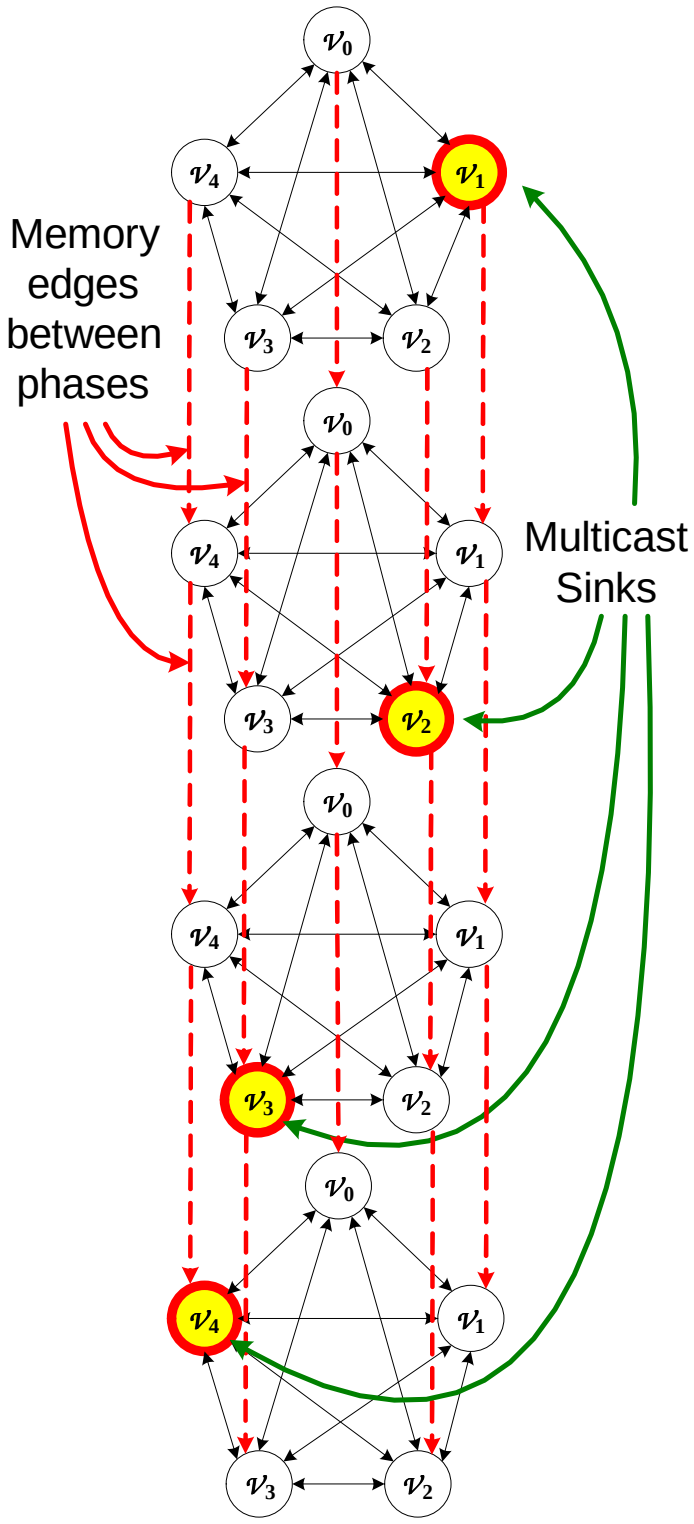


Figure 2.2: A 5-node network with one source and four sinks. The time-expanded graph has 4 copies (phases) of the network, each of which is connected via memory edges (red dashed lines) to other phases. The broadcast problem in the original network becomes a multicast problem with multiple sinks that are located in each phase.

Given this framework, the search for the most efficient download strategy becomes a linear programming (LP) problem, as described in [80] and restated below. In this LP, the optimization variables $f_i(e)$ and $x(e)$ represent the virtual flow to node i through edge $e \in \mathcal{E}$ and the total flow through edge $e \in \mathcal{E}$, respectively. The virtual flow $f_i(e)$ is the flow over edge e that is useful to node i. The LP is then given by

$$\begin{aligned}
\min_{\mathbf{f},\mathbf{x},\mathbf{t}} \quad & \mathbf{d}^T\mathbf{t} \\
s.t. \quad & x(e) \geq f_j(e),\ \forall j \in \{1,\ldots,m\},\ \forall e \in \mathcal{E} \\
& \sum_{v_j^{(\tau)}} \frac{x((v_i^{(\tau)}, v_j^{(\tau)}))}{c(v_i)} \leq t_\tau,\ \forall v_i^{(\tau)} \in \mathcal{V},\ \forall \tau \leq I \\
& f_j(e) \geq 0, \quad \forall j \in \{1,\ldots,m\},\ \forall e \in \mathcal{E} \\
& \sum_{v_k^{(\tau)}} f_j((v_k^{(\tau)}, v_i^{(\tau)})) - \sum_{v_k^{(\tau)}} f_j((v_i^{(\tau)}, v_k^{(\tau)})) \\
& = \left\{ \begin{array}{ll} F & \text{if } \tau = i \\ -F & \text{if } i = 0,\ \tau = 1 \\ 0 & \text{otherwise} \end{array} \right\}, \forall j \in \{1,\ldots,m\}
\end{aligned} \tag{2.2}$$

The first constraint sets the total flow on each edge to the maximum among all virtual flows over the edge; this value suffices for multicast network coding by [1]. The second constraint requires that the duration of each phase be the maximum, over all nodes, of the time required for node v to deliver its flow for the given phase; we calculate this value as the total flow out of node v divided by the upload capacity of v. The third constraint requires that all flows be non-negative. The last constraint guarantees the conservation of flow at all nodes in the network.

In [80], the LP is stated without an explicit proof. We provide a proof in the following section.

2.3.2 Verification of Correctness of the LP Formulation

In this section, we provide an explicit proof of correctness of the LP formulation. To verify the correctness of the LP, we will check if the flow at any time instant t is feasible. There are two different cases: (a) t is at the phase boundary, and (b) t is within the time slot.

We will check the capacity constraint and causal flow constraint (a counterpart of the flow conservation constraint) for these cases.

Now that the constraints in (2.2) are considered at the phase boundaries, the proof for the case where t is at the phase boundary is trivial. Therefore, we focus on proving the correctness of (2.2) for the case where t is within the time slot. For the proof, we first consider necessary and sufficient conditions for a feasible flow at any time instant t. Then, we show that the flow solution obtained from (2.2) satisfies these conditions.

Before starting the proof, we further define some notation.

- Let T_i and t_i denote the start time of the i^{th} time slot and the duration of the time slot, respectively.

- Let $F_i(n^{(i)})$ denote the amount of file at node $n^{(i)}$ at the beginning of the i^{th} time slot, T_i (where $F_i(n^{(i)}) \geq 0,\ \forall i,\ \forall n^{(i)} \in \mathcal{N}^{(i)}$). In other words, $F_i(n^{(i)})$ is the file size that is transferred from the replica of the node in the previous time slot ($n^{(i-1)}$) over the memory edge, and $F_{i+1}(n^{(i+1)})$ is the file size that is transferred to the replica of the node in the next time slot ($n^{(i+1)}$) over the memory edge.

- Let $h(l)$ and $h_p(l)$ be the total amount of actual flow and the multicast flow for destination p over edge $l \in \mathcal{E}^{(i)}_{tran}$ in the entire time slot, respectively.

2.3.2.1 Definition of a Feasible Flow

To define a "feasible flow" under our network model, we consider two constraints: (a) the upload capacity constraint of a node and (b) the causal flow constraint of a node. Since there are memory flows in our problem, we consider the causal flow constraint instead of the flow conservation constraint that is used for the static case where there are no memory flows.

Capacity Constraint:

Since we consider a flow problem in the time-expanded graph, the given upload capacity of a node is in terms of a transmission rate, rather than the amount of flow. Therefore, for a capacity constraint, the total instantaneous flow rate out of a node should be no greater than the upload capacity at any time instant. Note that within a time slot, the flow allocation is fixed. We also assume that the flow rate is constant within the time slot.

Then, the instantaneous flow rate over link $l \in \mathcal{E}_{tran}^{(i)}$ is $h(l)/t_i$. Now, the upload capacity constraint of a node at any time instant $t \in [T_i, T_i + t_i)$ is as follows:

$$\overbrace{\sum_{\substack{l:\ \text{tail}(l) = n^{(i)} \\ l \in \mathcal{E}_{tran}^{(i)}}} h(l)/t_i}^{\text{Instantaneous flow rate}} \leq \overbrace{cap(n^{(i)})}^{\text{Rate}},\ \forall n^{(i)} \in \mathcal{N}^{(i)} \tag{2.3}$$

Causal Flow Constraint:

In a network without memory, a flow solution is feasible if and only if it satisfies capacity constraints and flow conservation. In contrast, in a network with memory, there are additionally nonnegative memory flows over time, corresponding to received information that is stored at a node. In this case, the flow conservation constraint becomes the causal flow constraint: the total incoming flow up to time instant t is no less than the total outgoing flow up to t (called causality—the difference corresponds to the outgoing memory flow), and each sink node receives entire file by its finish time. Note that the total incoming flow includes the memory flow from the previous phase. Also, recall that for the causal flow constraint as well as the flow conservation constraint, we need to check these constraints for each multicast flow rather than the actual flow.

Now, the causal flow constraint of a node at any time instant $t \in [T_i, T_i + t_i)$ is as follows: for any peer p,

$$\begin{aligned} & \overbrace{\sum_{\substack{l:\ \text{head}(l) = n^{(i)} \\ l \in \mathcal{E}_{tran}^{(i)}}} h_p(l)\frac{(t - T_i)}{t_i}}^{\text{Total incoming flow in } [T_i,t)} + \overbrace{F_i(n^{(i)})}^{\substack{\text{Memory flow from} \\ \text{the previous phase}}} \\ \geq & \underbrace{\sum_{\substack{l:\ \text{tail}(l) = n^{(i)} \\ l \in \mathcal{E}_{tran}^{(i)}}} h_p(l)\frac{(t - T_i)}{t_i}}_{\text{Total outgoing flow in } [T_i,t)}, \quad \forall n^{(i)} \in \mathcal{N}^{(i)},\ \forall p \end{aligned} \tag{2.4}$$

In addition to the causal flow constraint within the time slot, we need the flow conser-

vation constraint at the phase boundary ($t = T_i + t_i$) as follows: for any peer p,

$$\begin{aligned} & \sum_{\substack{l: \ \mathrm{head}(l) = n^{(i)} \\ l \in \mathcal{E}^{(i)}_{tran}}} h_p(l) + F_i(n^{(i)}) - \sum_{\substack{l: \ \mathrm{tail}(l) = n^{(i)} \\ l \in \mathcal{E}^{(i)}_{tran}}} h_p(l) \\ = & \underbrace{F_{i+1}(n^{(i+1)})}_{\text{Memory flow to the following phase}} , \ \forall n^{(i)} \in \mathcal{N}^{(i)}, \ \forall p \end{aligned} \tag{2.5}$$

where for the source node $s^{(1)}$ in the first time slot, $F_1(s^{(1)}) = F$, and for the sink node $p_i^{(i)}$, $F_{i+1}(p_i^{(i+1)}) = F$.

2.3.2.2 Verification of the LP

We show that the flow solution from (2.2) satisfies constraints (2.3)-(2.5) at any time instant t within the time slot ($t \in (T_i, T_i + t_i)$).

Capacity Constraint: The flow solution $x(l)$ from (2.2) satisfies the capacity constraint as follows:

$$\sum_{\substack{l \, : \, \mathrm{tail}(l) = n \\ l \in \mathcal{E}^{(i)}_{tran}}} x(l)/cap(n) \leq t_i, \ \forall n \in \mathcal{N}, \ \forall i \tag{2.6}$$

Here, t_i is the maximum over all nodes belonging to the i^{th} time slot. By replacing t_i and $x(l)$ with t_i and $h(l)$, respectively, we show that (2.6) is equivalent to (2.3), since $t_i \geq 0$, and $cap(n) \geq 0$. Therefore, the LP solution is feasible in terms of the capacity constraint at any time instant.

Causal Flow Constraint: The flow conservation constraint in (2.2) is as follows:

$$\begin{aligned} & \sum_{l: \mathrm{head}(l) = n} f_{p_k^{(k)}}(l) - \sum_{l: \mathrm{tail}(l) = n} f_{p_k^{(k)}}(l) \\ = & \begin{cases} F & \text{if } n = p_k^{(k)} \\ -F & \text{if } n = s^{(1)} \\ 0 & \text{otherwise} \end{cases} , \forall k \in \{1, 2, \ldots, m\} \end{aligned} \tag{2.7}$$

Since each summation of incoming and outgoing flows includes the memory flows in

(2.7), we can separate them for clarification as follows:

$$\begin{aligned} & \sum_{\substack{l: \ \mathrm{head}(l) = n^{(i)} \\ l \in \mathcal{E}^{(i)}_{tran}}} f_{p_k^{(k)}}(l) + F_i(n^{(i)}) \\ = & \sum_{\substack{l: \ \mathrm{tail}(l) = n^{(i)} \\ l \in \mathcal{E}^{(i)}_{tran}}} f_{p_k^{(k)}}(l) + F_{i+1}(n^{(i+1)}) \\ & \text{where} \begin{cases} F_{i+1}(n^{(i+1)}) = F & \text{if } n^{(i)} = p_k^{(k)} \\ F_i(n^{(i)}) = F & \text{if } n^{(i)} = s^{(1)} \end{cases} \end{aligned} \tag{2.8}$$

Then, (2.8) is equivalent to (2.5) at the phase boundary. Now, let's consider the causal flow constraint within a time slot (i.e., $t \in [T_i, T_i + t_i)$). The net incoming flows to a node n up to time t is non-negative as shown in (2.9).

$$\begin{aligned} & \sum_{\substack{l: \ \mathrm{head}(l) = n \\ l \in \mathcal{E}^{(i)}_{tran}}} f_{p_k^{(k)}}(l) \frac{t - T_i}{t_i} + F_i(n) - \sum_{\substack{l: \ \mathrm{tail}(l) = n \\ l \in \mathcal{E}^{(i)}_{tran}}} f_{p_k^{(k)}}(l) \frac{t - T_i}{t_i} \\ \overset{(a)}{\geq} & \left(\sum_{\substack{l: \ \mathrm{head}(l) = n \\ l \in \mathcal{E}^{(i)}_{tran}}} f_{p_k^{(k)}}(l) + F_i(n) - \sum_{\substack{l: \ \mathrm{tail}(l) = n \\ l \in \mathcal{E}^{(i)}_{tran}}} f_{p_k^{(k)}}(l) \right) \frac{t - T_i}{t_i} \\ \overset{(b)}{=} & \ F_{i+1}(n) \frac{t - T_i}{t_i} \quad \overset{(c)}{\geq} \quad 0 \end{aligned} \tag{2.9}$$

In (2.9), (a) is from $F_i(n) \geq 0$ and $0 \leq \frac{t-T_i}{t_i} \leq 1$, (b) is from (2.8), and (c) is from non-negativity of $F_{i+1}(n)$ and $\frac{t-T_i}{t_i}$. Therefore, the flow solution from (2.2) also satisfies (2.4) at any time instant t within a time slot. Therefore, the LP solution is feasible in terms of the causal flow constraint at any time instant.

This completes the verification of feasibility.

2.3.3 Number of Phases for Minimum Finish Times

In [80], the authors implicitly assumed that there is one phase per peer. In this section, we explicitly prove that each peer indeed needs only one phase to achieve the minimum finish time.

Theorem 2.1. *In this network model, one phase for each peer is enough to achieve the minimum finish times.*

Recall that if we assume that there is no transmission delay within a time slot, each peer can transmit a packet while it is receiving the packet from the others. In addition, due to network coding, it is unnecessary to trace each packet (i.e., no identification is necessary) in scheduling.

To prove Theorem 2.1, it is sufficient to show the following Lemma:

Lemma 2.1. *Suppose that a given arbitrary schedule where some peer uses 2 phases achieves the minimum finish time. Then, there exists another schedule in which these 2 phases are combined into one phase such that the finish time does not increase and flows in all the other phases are unchanged.*

Proof. Suppose that we are given a schedule where a peer requires 2 phases (say, i^{th} and $(i+1)^{\text{th}}$ time slots), which achieves the minimum finish time for the peer. Let $x^{(i)}(l)$, $f_p^{(i)}(l)$, and t_i be the optimal flow over link l, the optimal multicast flow over link l for destination p, and time slot duration for the i^{th} time slot, respectively. Since $x^{(i)}(l)$, $f_p^{(i)}(l)$, $x^{(i+1)}(l)$, and $f_p^{(i+1)}(l)$ are optimal flows, all of them satisfy the constraints in (2.2).

Let $g_p^{(i,i+1)}(l)$ $(l \in \mathcal{E}_{tran})$ be the superposition of $f_p^{(i)}(l)$ and $f_p^{(i+1)}(l)$ for $l \in \mathcal{E}_{tran}$, and $y^{(i,i+1)}(l)$ be the maximum of $g_p^{(i,i+1)}(l)$ over all multicast flow destinations. i.e.,

$$\begin{aligned} g_p^{(i,i+1)}(l) &= f_p^{(i)}(l) + f_p^{(i+1)}(l),\ \forall l \in \mathcal{E}_{tran},\ \forall p \\ y^{(i,i+1)}(l) &= \max_p g_p^{(i,i+1)}(l) \end{aligned} \tag{2.10}$$

Note that since the maximum of the sum is always no greater than the sum of the maximum, $y^{(i,i+1)}(l)$ is no greater than the superposition of the actual flows, as follows:

$$\begin{aligned} & y^{(i,i+1)}(l) \\ = \ & \max_p g_p^{(i,i+1)}(l) \\ = \ & \max_p \{f_p^{(i)}(l) + f_p^{(i+1)}(l)\} \\ \leq \ & \underbrace{\max_p f_p^{(i)}(l)}_{x^{(i)}(l)} + \underbrace{\max_p f_p^{(i+1)}(l)}_{x^{(i+1)}(l)},\ \forall l \in \mathcal{E}_{tran} \\ \therefore \ & y^{(i,i+1)}(l) \leq x^{(i)}(l) + x^{(i+1)}(l),\ \forall l \in \mathcal{E}_{tran} \end{aligned} \tag{2.11}$$

Now, let's consider the capacity constraint. From (2.11), we have the following:

$$\begin{aligned} & \sum_{\substack{l\,:\,\mathrm{tail}(l)=n \\ l\in\mathcal{E}_{tran}^{(i)}}} \frac{y^{(i,i+1)}(l)}{cap(n)} \\ \leq \quad & \sum_{\substack{l\,:\,\mathrm{tail}(l)=n \\ l\in\mathcal{E}_{tran}^{(i)}}} \frac{x^{(i)}(l)+x^{(i+1)}(l)}{cap(n)} \\ \leq \quad & t_i+t_{i+1}, \quad \forall n\in\mathcal{E} \end{aligned} \tag{2.12}$$

where the last inequality also comes from the argument that the maximum of the sum is always no greater than the sum of the maximum. Let t' denote the maximum over all nodes of the first line of (2.12). Then, since (2.12) holds for all nodes, t' is no greater than t_i+t_{i+1}, and this can be achieved by superposition of flows over transmission edges.

Next, we will show that the flow conservation constraint (or equivalently, causal flow constraint: the last constraint in (2.2)) is satisfied by the superposed flows. Recall that only the flows over the transmission edges are superposed. As in Section 2.3.2, we use the same notation for $F_i(n^{(i)})$, and we can rewrite the flow conservation constraint of the i^{th} phase by separating memory edge flows and transmission edge flows as in (2.8). If we combine the flow conservation equalities of the i^{th} phase and the $(i+1)^{th}$ phase, then we have a flow conservation equation for $g_p^{(i,i+1)}(l)$ as in (2.13).

$$\begin{aligned} & \begin{cases} \displaystyle\sum_{l:\ \substack{\mathrm{head}(l)=n^{(i)} \\ l\in\mathcal{E}_{tran}}} f_p^{(i)}(l) - \sum_{l:\ \substack{\mathrm{tail}(l)=n^{(i)} \\ l\in\mathcal{E}_{tran}}} f_p^{(i)}(l) = -F_i(n^{(i)})+F_{i+1}(n^{(i+1)}) \\ \displaystyle\sum_{l:\ \substack{\mathrm{head}(l)=n^{(i+1)} \\ l\in\mathcal{E}_{tran}}} f_p^{(i+1)}(l) - \sum_{l:\ \substack{\mathrm{tail}(l)=n^{(i+1)} \\ l\in\mathcal{E}_{tran}}} f_p^{(i+1)}(l) = -F_i(n^{(i+1)})+F_{i+1}(n^{(i+2)}) \end{cases} \\ \Rightarrow \quad & \sum_{l:\ \substack{\mathrm{head}(l)=n^{(i)} \\ l\in\mathcal{E}_{tran}}} \left[f_p^{(i)}(l)+f_p^{(i+1)}(l)\right] - \sum_{l:\ \substack{\mathrm{tail}(l)=n^{(i)} \\ l\in\mathcal{E}_{tran}}} \left[f_p^{(i)}(l)+f_p^{(i+1)}(l)\right] \\ & \quad = -F_i(n^{(i)})+F_{i+1}(n^{(i+2)}) \\ \Rightarrow \quad & \sum_{l:\ \substack{\mathrm{head}(l)=n^{(i)} \\ l\in\mathcal{E}_{tran}}} g_p^{(i,i+1)}(l) - \sum_{l:\ \substack{\mathrm{tail}(l)=n^{(i)} \\ l\in\mathcal{E}_{tran}}} g_p^{(i,i+1)}(l) = -F_i(n^{(i)})+F_{i+1}(n^{(i+2)}) \end{aligned} \tag{2.13}$$

When we combine two phases, the incoming memory edge flows of the combined phase should be unchanged as $F_i(n^{(i)})$. Then, by (2.13), the new outgoing memory edge flows of

the combined phase are still $F_{i+1}(n^{(i+2)})$, which remains the same as the ones in the original graph before superposition. Therefore, we can combine the two phases in a way such that the transmission edge flows are superposed, and the memory edge flows at the boundaries are unchanged. Then, with the setting[2] used in (2.8), i.e.,

$$\begin{cases} F_{j+1}(n^{(j+1)}) = F & \text{if } n^{(j)} = p \\ F_j(n^{(j)}) = F & \text{if } n^{(j)} = s^{(1)} \end{cases}, \ j = i \text{ and } (i+1)$$

we can show that (2.13) is equivalent to the flow conservation constraint in (2.2). Therefore, the flow conservation constraints are also satisfied by the superposed flows.

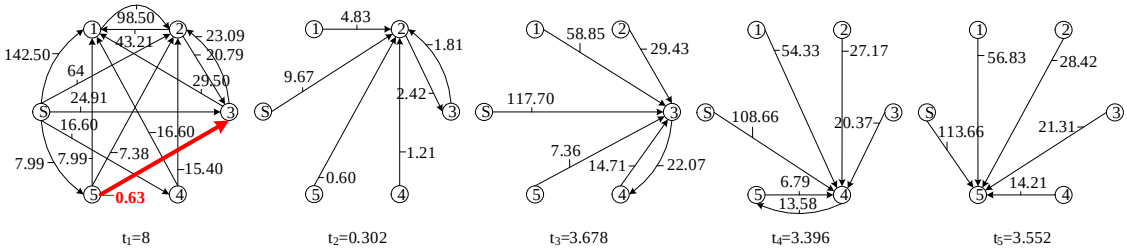
Note that all of the flows before and after the combined phase are unchanged, and the duration of combined phase is no greater than the sum of two original phase durations. Therefore, if we combine two phases by superposing the transmission edge flows and using the same memory edge flows, then all finish times are unchanged. This completes the proof. □

2.4 Min-Min versus Min-Avg Finish Times

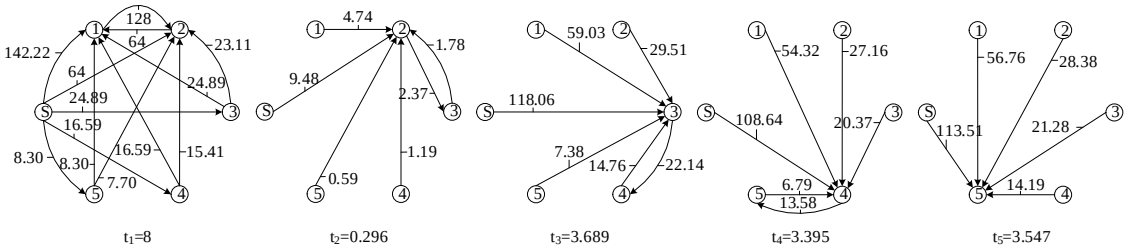
In this section, we show by example that routing algorithms achieving Min-Min finish times do not necessarily minimize the average finish time. This contradicts Claim 1 in [27].

A counterexample with five peers is shown in Fig. 2.3. The source has upload capacity 32, and each peer has upload capacity 8. We use the LP (2.2) to find optimal flow solutions for Min-Avg and Min-Min. We then show that routing is sufficient to achieve the optimal solutions in both cases. We prove the existence of an optimal routing solution by explicitly labeling the identities of the flows (see Fig. 2.4). The labeling procedure is simplified by noting that by the i^{th} time step, the first i peers each have all of the data. As a result, for the $(i+1)^{th}$ phase, we only need to check if peers $i+2, i+3, \ldots, m$ can send distinct data to peer $i+1$. In this example, the average finish time for Min-Avg is less than that of the Min-Min solution, which contradicts Claim 1 in [27]. Let M be the maximal number of users that can finish in the "*bottleneck time*", $F/c(0)$. In [27] the authors tried to prove Claim 1 by first showing that it is necessary to minimize $\sum_{i=1}^{M+1} T_i$ in order to minimize the

[2] In the proof, we assume that a peer requires two phases to minimize the finish time. Therefore, no other peer completes download between the i^{th} phase and the $(i+1)^{th}$ phase, which means that $n^{(i)} \neq p$ and consequently, $F_i(n^{(i+1)}) \neq F$ for all peers.



(a) Min-Avg finish time solution : Finish times $\{8, 8.302, 11.980, 15.376, 18.928\}$ and the average finish time is 12.5173.



(b) Min-Min finish time solution : Finish times $\{8, 8.296, 11.986, 15.381, 18.928\}$ and the average finish time is 12.5181.

Figure 2.3: Optimal flow solutions for a P2P network with 5 peers and upload capacity constraints $c(0) = 32$ and $c(i) = 8$, $i \in \{1, \ldots, 5\}$. The file size is 256. Each graph shows a single phase. Edges are labeled with the total amount of flow along the edge in that phase. Recall that t_i denotes the duration of the i^{th} phase. Min-Avg scheduling (a) has smaller average finish time than Min-Min scheduling (b). The main difference comes from the bold link in red.

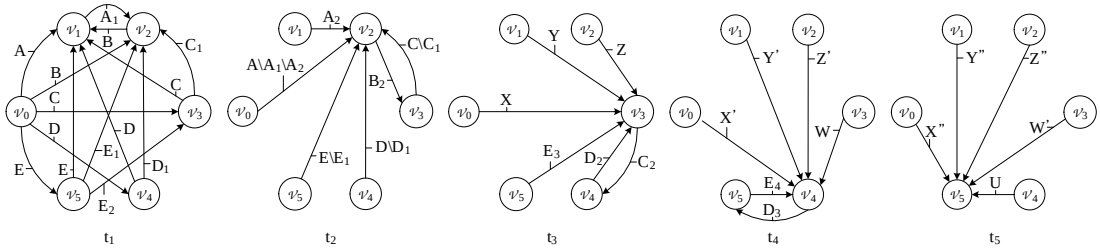


Figure 2.4: A routing realization of the optimal network coding solution from Fig. 2.3. Each edge carries the same amount of information as the corresponding edge from Fig. 2.3. Edges are labeled with the identity of the information being transmitted. The correctness of the first two phases is easy to verify. In the later phases, $C_2 \subseteq C$, $D_2 \subseteq D$, $D_3 \subseteq D$, $E_3 \subseteq E \backslash E_2$, $E_4 \subseteq E$ can be any subsets satisfying causality. The other data flows X, Y, Z, U, W are from nodes that have the entire file.

average finish time $\sum_{i=1}^{m} T_i$. However, the counterexample in Fig. 2.3 shows that minimizing $\sum_{i=1}^{M+1} T_i$ is *not* necessary for minimizing $\sum_{i=1}^{m} T_i$.

In the example of Fig. 2.3, the main difference between the two strategies occurs in the first phase. For Min-Min scheduling, peer 5 sends data to peers 1 and 2 only. For Min-Avg scheduling, peer 5 sends data to peers 1, 2, and 3 (the bold link in red in Fig. 2.3(a)). Sending data to peer 3 delays peer 2's finish time in the second phase but significantly reduces the duration of the third phase.

We observe empirically that the Min-Avg and Min-Min strategies can differ for networks with more than 4 peers. For most randomly generated capacity values, the difference between the finish times resulting from the two strategies is nonzero but small. When the peers all have the same upload capacities, the gap increases with the number of peers. The difference between the finish times of Min-Avg and Min-Min is 0.032% for the 5 peer example in (Fig. 2.3), and 0.171% for an example with 10 peers, all with capacity constraint 8.

This example illustrates the power of the network coding framework for routing problems. Finding an optimal routing solution directly is often extremely difficult. Using the given LP, we can find an optimal coding solution in polynomial time. In some cases, applying our labeling strategy to the resulting coding solution allows us to demonstrate that the optimal solution is also achievable with routing alone.

2.5 Reciprocity Constraints

Reciprocity is a concept used in incentive-compatible protocols to encourage users to operate in a manner that benefits the entire network. In this section, we show how the LP approach can be used to study the effect of reciprocity. The goal of reciprocity constraints is to encourage peers joining the network to help in distributing information to other users. A number of simple models for reciprocity are possible. For example, one model of reciprocity sets a reciprocity constant $\rho \in [0, 1]$ and imposes the constraint that v_i should send to v_j approximately the same amount of information as v_j sends to v_i in each phase (Fig. 2.5)—more precisely, the two flows should differ by at most a factor ρ. The reciprocity constraint

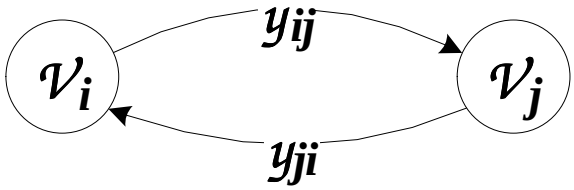


Figure 2.5: The graphical view for the reciprocity constraint: v_i should send to v_j approximately the same amount of information as v_j sends to v_i in each phase, $y_{ij} \approx y_{ji}$.

can be applied to virtual flows as

$$\rho f_i(v_j^{(\tau)}, v_i^{(\tau)}) \leq f_j(v_i^{(\tau)}, v_j^{(\tau)}) \leq f_i(v_j^{(\tau)}, v_i^{(\tau)})$$

or to actual flows as

$$\rho x(v_j^{(\tau)}, v_i^{(\tau)}) \leq x(v_i^{(\tau)}, v_j^{(\tau)}) \leq x(v_j^{(\tau)}, v_i^{(\tau)}).$$

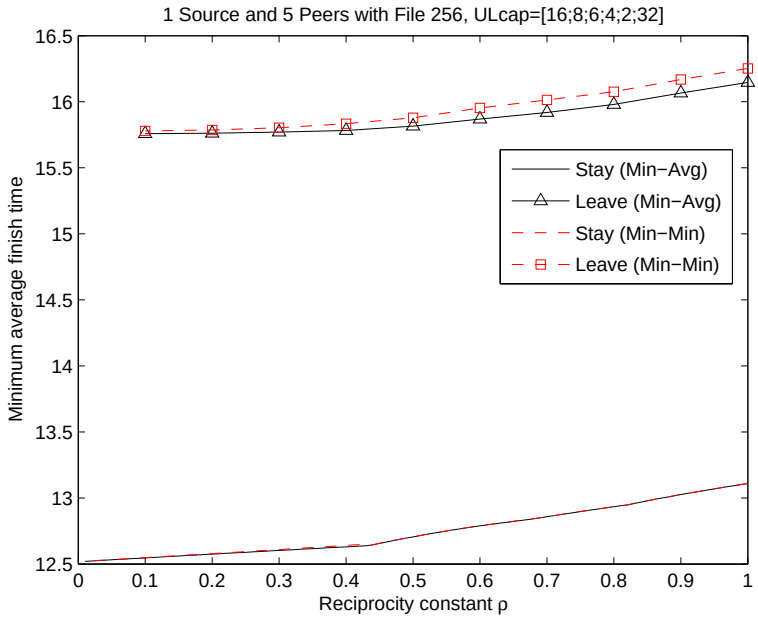
Lemma 2.2 shows that the two definitions are equivalent since $f_j(i,j) = x(i,j)$ for all i, j. A variety of other definitions of reciprocity are possible. One alternative way to model reciprocity is to set a limit on the absolute difference between the cumulative amount sent in each direction. Since both of these constraints are linear, either one can be added to the LP. The examples that follow use the first model.

Lemma 2.2. *For any P2P file distribution network there exists an optimal solution in which*

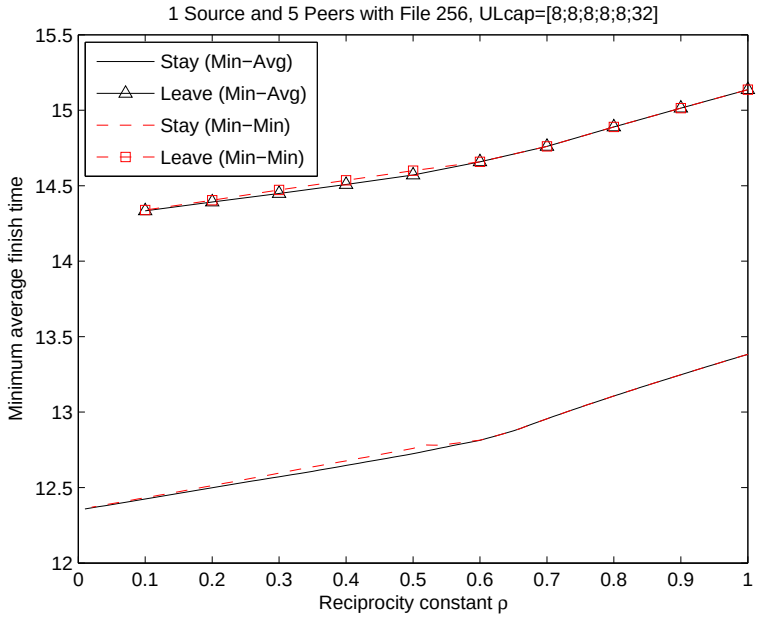
$$f_j(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)}) \quad \textit{for all } (i,j),\ \tau$$

By Lemma 2.2, definitions of reciprocity in terms of $f_j(i,j)$ and those in terms of $x(i,j)$ are always equivalent.

Proof. For a given flow solution, we consider two cases: 1) $f_j(v_i^{(\tau)}, v_j^{(\tau)}) < x(v_i^{(\tau)}, v_j^{(\tau)})$ and 2) $f_j(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)})$. For the first case, we partition the total flow $x(v_i^{(\tau)}, v_j^{(\tau)})$ into the portion that contains the virtual flow $f_j(v_i^{(\tau)}, v_j^{(\tau)})$ and the part that does not. Note that the second part contains information that is linearly dependent on information already received at peer v_j. As a result, node v_j can serve any node v_k that relies on this information without sending this part of the transmission. The following argument makes

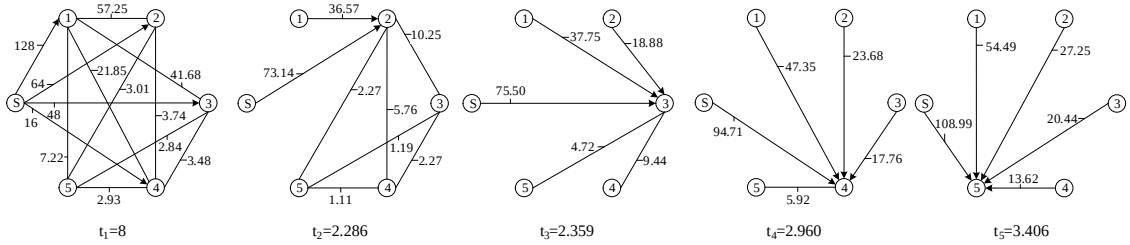


(a) Heterogeneous Capacities

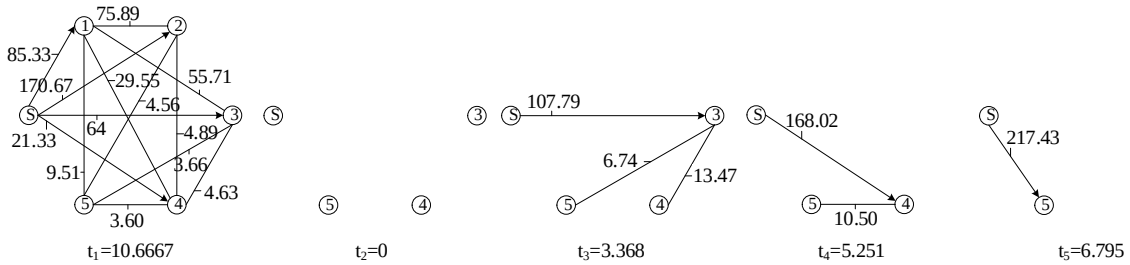


(b) Homogeneous Capacities

Figure 2.6: The minimum average finish time versus reciprocity constant ρ. Recall that $\rho = 0$ means no reciprocity, and $\rho = 1$ means a strict reciprocity. The graphs plot average finish times of Min-Avg and Min-Min scheduling for (a) heterogeneous (Upload capacities: server 32 and peers $\{16, 8, 6, 4, 2\}$) and (b) homogeneous (Upload capacities: server 32 and peers 8) upload capacities.



(a) Peers staying: Finish times $\{8, 10.2857, 12.6452, 15.6048, 19.0108\}$ and the average finish time is 13.1093.



(b) Peers leaving: Finish times $\{10.667, 10.667, 14.035, 19.286, 26.081\}$ and the average finish time is 16.1470.

Figure 2.7: Optimal Min-Avg flow solutions for a network with heterogeneous capacities case and a strict reciprocity constraint. File size is 256, server's upload capacity is 32, and peers' upload capacities are $\{16, 8, 6, 4, 2\}$. We consider both (a) the case where peers stay after completing download and (b) the case where peers leave the network after completing download. A link without an arrow denotes a bidirectional link that has the same amount of flow in each direction. Without the reciprocity constraint where peers stay and leave after completing download, the average finish times are 12.517 and 15.756, respectively. Note that peer 1 and 2 finish at the same time in the first phase for (b).

this precise.

We wish to show that there exists an optimal solution such that $f_j(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)})$ for all i, j. Suppose that an optimal solution has an edge $(v_i^{(\tau)}, v_j^{(\tau)})$ for which $f_j(v_i^{(\tau)}, v_j^{(\tau)}) < x(v_i^{(\tau)}, v_j^{(\tau)})$. Since $x(v_i^{(\tau)}, v_j^{(\tau)}) = \max_k f_k(v_i^{(\tau)}, v_j^{(\tau)})$, there exists some $k \neq j$ for which $f_k(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)})$. Since $x(v_i^{(\tau)}, v_j^{(\tau)}) > f_j(v_i^{(\tau)}, v_j^{(\tau)})$, the given solution sends $x(v_i^{(\tau)}, v_j^{(\tau)}) - f_j(v_i^{(\tau)}, v_j^{(\tau)})$ bits from peer v_i to peer v_j for use by peer v_k, but all of these bits are linearly dependent on bits already known to peer v_j. As a result, we can remove these redundant $x(v_i^{(\tau)}, v_j^{(\tau)}) - f_j(v_i^{(\tau)}, v_j^{(\tau)})$ linearly dependent bits from $v_i^{(\tau)}$ to $v_j^{(\tau)}$, leaving the rest of the solution unchanged. □

When considering incentive-compatible mechanisms such as reciprocity where non-altruistic peers leave upon completion, there is an inherent design tension between optimizing individual and average finish times. The issue here is that average finish time can be improved by delaying the completion time for fast peers so that they continue to contribute upload capacity. We simply note that for any given objective function and ordering of peers finishing, we can use the LP formulation (2.2) with the following additional linear constraint—requiring all outgoing flows from a peer that completed its download to be zero after its finish time:

$$\sum_{v_j^{(\tau)}} x((v_i^{(\tau)}, v_j^{(\tau)})) \leq c(v_i^{(\tau)}) = 0, \, \forall v_i^{(\tau)} \in \mathcal{V}, \; i < \tau \leq I \tag{2.14}$$

Figure 2.6 shows Min-Avg and Min-Min finish times, for example, heterogeneous and homogeneous P2P networks. In both examples the upload capacity of the server is 32. The upload capacities for the peers are $(16, 8, 6, 4, 2)$ in the heterogeneous network and $(8, 8, 8, 8, 8)$ in the homogeneous network. Results for reciprocity coefficients varying from 0 to 1 are included, where $\rho = 0$ means no reciprocity, and $\rho = 1$ means strict reciprocity (i.e., $x(e) = x(e')$ for all pairs). Reciprocity constraints are applied to all nodes except for the server.

As expected, the minimum average finish time increases as ρ increases. We note, however, that in these examples, increasing reciprocity from 0 to 1 increases the minimal average finish times for Min-Min and Min-Avg strategies by less than 10%. In Fig. 2.7, we show an optimal Min-Avg flow solution for a sample case with a strict reciprocity ($\rho = 1$) and

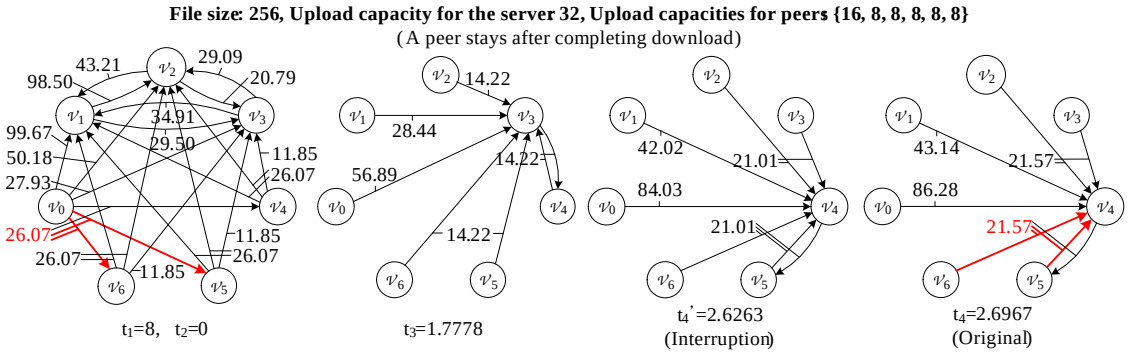


Figure 2.8: Optimal flow graph for the example of Section 2.6 for both coding and routing. Note that in this case, v_1 and v_2 finish at the same time in the first phase (the second phase has zero duration). In the static network, the duration of the fourth phase is $2.6967s$. Note that neither v_5 nor v_6 can send the whole data that were received from the server in the first phase (highlighted in red). Instead, the server (including seeds) should compensate by sending the repeated data. Before the end of the fourth phase, at $T = 12.4038s$, an interruption occurs. Until then, v_4 receives the repeated data, the amount of 4.5037.

heterogeneous upload capacities and peers (a) stay or (b) leave the network after completing their downloads.

2.6 Robustness Benefit of Coding

In this section, we show that network coding can improve the P2P networks robustness against unforeseen events such as changes in upload capacity, changes in connectivity, or nodes joining or leaving the network unexpectedly.

For instance, consider the following scenario with a file of size 256, a server of capacity 32, and 6 peers whose capacities are $\{16, 8, 8, 8, 8, 8\}$. For the static case without reciprocity, we obtain an optimal flow solution from (2.2) and find a corresponding routing solution, as described in Section 2.4 (see Fig. 2.8). Now suppose that the network follows an optimal solution up to the fourth phase, when an unexpected event occurs. By then, the fastest three peers have finished their downloads. As a result, these peers have the complete file, and we treat them as part of an augmented server. This effectively increases the server's capacity from 32 to 64. In the optimal routing solution, $v_5^{(4)}$ and $v_6^{(4)}$ use phase 4 to send to $v_4^{(4)}$ some of the data that v_5 and v_6 received from the server in previous phases. (The relevant flows are highlighted in red in Fig. 2.8.) The rest of the data is sent to $v_4^{(4)}$ by the server. Therefore, $v_4^{(4)}$ receives directly from the source some of the same information that

Table 2.1: Finish times for each case (server capacity is decreased to 0.1)

	Coding	Routing (half)	Routing (worst)
Finish Time of v_4	18.00	40.52	63.04
Finish Time of v_5	38.19	40.52	63.04
Avg. Finish Time*	16.27	21.25	30.25

* Finish times for the first 3 peers are same for all cases.

$v_5^{(4)}$ and $v_6^{(4)}$ have previously received from the server. Before the fourth phase, both v_5 and v_6 receive data only from the server and only in the first phase. In a routing solution, the server can only time share between sending information known to v_5 and information known to v_6 and information known to both. In contrast, in a network coding solution, the server can send linear combinations of these subsets of the data.

Now suppose that at time $\tau \in (T_3, T_4)$, the connectivity between the augmented server and the remaining peers decreases greatly, and, probabilistically, either $v_5^{(4)}$ or $v_6^{(4)}$ leaves the system. Without prior knowledge of which peer will leave the system, any particular choice from the family of time sharing routing solutions will have a worse expected average finish time than the coding solution.

To give a specific numerical example, suppose that the network disruption occurs at 12.4038 seconds (2.6263 seconds after the beginning of the fourth phase). Note that in the fourth phase, the server sends an amount 163.56 of innovative data in the first 2.5559 seconds, and an amount 4.5037 of repeated data in the remaining 0.0704 seconds before the disruption. At the time of the disruption, the capacity of the augmented server decreases to 0.1, and either v_5 or v_6 leaves the network. In the case where v_6 leaves the system, Table 2.1 shows the finish times when coding is used, when routing is used and the server sends equal amounts of data known to v_5 and v_6, and when routing is used and the server sends only v_5's data.

Note that the performance gap between coding and routing can be made arbitrarily large by reducing the capacity between the augmented server and the remaining peers after the disruption.

2.7 Conclusions

In this chapter, we apply a linear programming approach based on network coding to analyze download finish times in a P2P network. We focus on an uplink sharing model with one source and multiple peers in a complete graph, but the optimization framework extends readily to more general cases. We rigorously prove the feasibility of the flow solution obtained from the LP, and we show that one phase (a period of invariant flow allocation) for each peer is sufficient to achieve the minimum finish time. We disprove the claim in [27] that Min-Min scheduling achieves the minimum average finish time for routing. We also investigate the effect of reciprocity using the LP. Lastly, we show that coding can provide a robust optimal solution, outperforming routing in dynamically changing network scenarios. Ultimately, we expect that the LP can be used to gain insights into how to design practical P2P algorithms, and to predict how different factors will affect the strategies that can be used in practice and the resulting performance.

Chapter 3

Peer-to-Peer Anonymous Networking

The goal of anonymous networking is to communicate information over a network without revealing the identities of communicating nodes. Applications of anonymous networking include electronic voting, military communications, and communications of a sensitive commercial or political nature.

In this chapter, we consider design and analysis of coding-based anonymous routing systems in peer-to-peer (P2P) overlay networks. An unknown subset of participating nodes is adversarial, and can collude to try to identify the communicating nodes. The adversarial nodes contribute to the system by relaying packets, while attempting to use their observations to identify source and sink nodes.

The first part of this work, in Section 3.3, considers subgraph setup in the absence of a reliable public key infrastructure (PKI). We propose a coding-based scheme that allows a source node to anonymously set up a subgraph involving multiple relay nodes, and to anonymously send a small secret message (e.g., a cryptographic key) to the sink. We analyze the anonymity and security properties of this scheme in an information theoretic framework.

The second part of this work, in Section 3.4, focuses on the data transmission phase, assuming availability of a subgraph setup scheme (either PKI-based or coding-based) and end-to-end encryption (either by public or symmetric keys shared by source and sink). In particular, we employ end-to-end encryption for data security, but we use network coding at intermediate nodes to improve networking performance and reduce complexity (by replacing expensive cryptographic operations at each hop with simpler linear algebra operations).

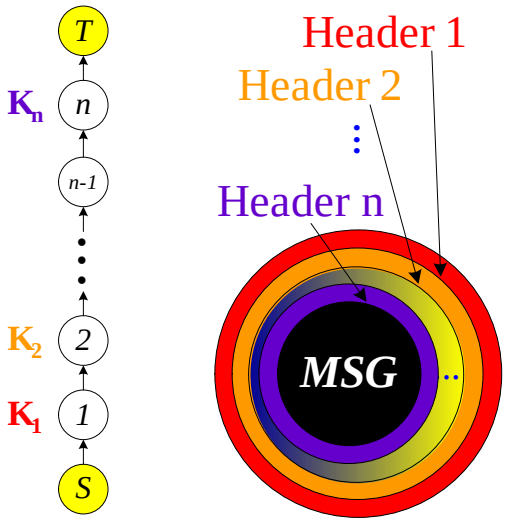


Figure 3.1: In onion routing [24, 67], to obtain next hop information, each user decrypts and "peels off" the header by using its private key, K_i.

We first provide a description of background and related work in Section 3.1 and the general network and adversary model in Section 3.2.

3.1 Background and Related Work

Many anonymous networking systems that rely on a public key infrastructure (PKI) have been proposed, starting from the seminal work of Chaum [16] on mix networks, to the "onion routing" approach of Reed et al. [67] and the Tor protocol [24] (illustrated in Fig. 3.1) which is the most widely used anonymous networking system currently. The public keys of intermediate relay nodes are used to recursively encrypt information at the source, and each relay node decrypts a layer of information using its private key. A number of anonymous networking strategies have also focused on P2P overlay networks, as the decentralized nature of P2P systems and their potential to scale to a large number of participating nodes are attractive in many scenarios. Such proposed schemes include Tarzan [31] (a P2P networking system based on onion routing), MorphMix [69] (similar to Tarzan but where the routes are determined by intermediaries), and Salsa [59] and Torsk [53] (structured approaches to build scalable P2P anonymous networks), which require a reliable public key infrastructure.

Other P2P anonymous networking schemes such as Crowds [68], AP3 [56], and the "slicing the onion" scheme of Katti *et al.* [45] have less reliance on a PKI, and are use-

ful in situations where a PKI is not available/reliable or may potentially be compromised. Crowds and AP3 use randomized forwarding, and protect the sender but not the receiver identity. The "slicing the onion" scheme considers both sender and receiver anonymity. It splits routing information across multiple relay nodes which are arranged in a rectangular subgraph consisting of l layers of d nodes each, as illustrated in Fig. 3.3. Information intended for each node (which includes information about its next hop nodes) is split into d slices and multiplied with an invertible $d \times d$ matrix. Thus, each node is able to decode its intended information using the packets received from all nodes in the previous layer, while being unable to decode information intended for other nodes. The type of security provided is information theoretic in nature, relying on path diversity and not on computational assumptions. However, the scheme is not strongly secure in the information theoretic sense (as measured by mutual information) and no formal analytical characterization of the security of the scheme is given, e.g., it is not clear how much information is leaked to an adversary who controls a subset of relay nodes. In Section 3.3 we propose a different coding scheme over the subgraph with a formal information theoretic security characterization, and further consider optimization and randomization of the subgraph parameters.

Several recent works [28,33,79] have investigated the use of network coding in anonymous networking assuming availability of a separate scheme such as those discussed above, for setting up a subgraph anonymously. These works propose modifications to conventional network coding to protect the coded packets against traffic content correlation. In practical network coding, the source information is divided into multiple generations of packets. Network nodes carry out random linear coding among packets of each generation, and the coding operations are captured by global encoding vectors (GEVs) that undergo the same linear coding operations as the data. Such coding is not compatible with the layered encryption schemes employed in non-network coded anonymity schemes to cryptographically transform packet contents at each hop. To address this issue, Fan *et al.* [28] proposed a scheme in which GEVs are encrypted using homomorphic encryption so that only the sink node with the appropriate decryption key can decode the GEVs and hence the message. While traffic content correlation is made more difficult for the adversary, an adversary who controls multiple participating nodes can still check if a packet is in the span of another set of h packets with $O(h^3+hn)$ complexity, where n is the length of each packet. In Section 3.4 we propose an alternative approach using algebraic coding over layered subgraphs, where

the complexity of such content correlation attacks is substantially higher. Wang *et al.* [79] proposed a lower overhead network coding scheme where only routing information, flow and generation numbers are encrypted while GEVs and message contents are not encrypted. The scheme hides the correlation of upstream and downstream GEVs of flows by designing the GEVs to be linearly dependent with those from other flows, but is only secure against external observers and not internal participating nodes. Gasti *et al.* [33] considered the problem of checking data integrity in anonymous network coded peer-to-peer file sharing networks in the presence of active adversaries that may corrupt coded packets. Unlike PKI-based integrity checking schemes used in the non-anonymous case, the authors proposed a hash-based approach for integrity checking of packets.

3.2 Model

3.2.1 Network and Adversary Model

The P2P overlay network consists of N participating nodes, each of which is adversarial independently with probability p. There are multiple concurrent unicast sessions, each with one source and one sink. Each source chooses a random subset of nodes from the network to construct an overlay subgraph, which it uses to communicate anonymously with an intended sink node. By choosing the nodes randomly, we avoid potential attacks where the adversary can try to bias the choice towards adversarial nodes by advertising favorable characteristics.[1] A relaying node can serve in multiple communication sessions (i.e., different subgraphs) simultaneously. We assume that the underlying physical network is generally well-connected so that there is path diversity between source and sink nodes.[2]

As in [56], techniques from structured P2P overlay networks, e.g., [12], can be used to provide an efficient means of choosing a random subset of nodes from a large network, in conjunction with techniques for defending against Sybil attacks either with or without a PKI, e.g., [11]. This is further discussed in Section 3.3.9.

We consider passive attacks from adversarial participating nodes, who collude to try to determine the source and sink identities from their observed transmissions and connectivity

[1] In cases where information such as the geographic location of nodes provides an indicator of their probability of being adversarial, such information can be taken into account in the choice of relay nodes and subgraph design. This is a topic of future work.

[2] As will be evident from our study below, if all overlay paths between a source-sink pair pass through a very small number of physical nodes, then the anonymity of the coding-based schemes will be reduced.

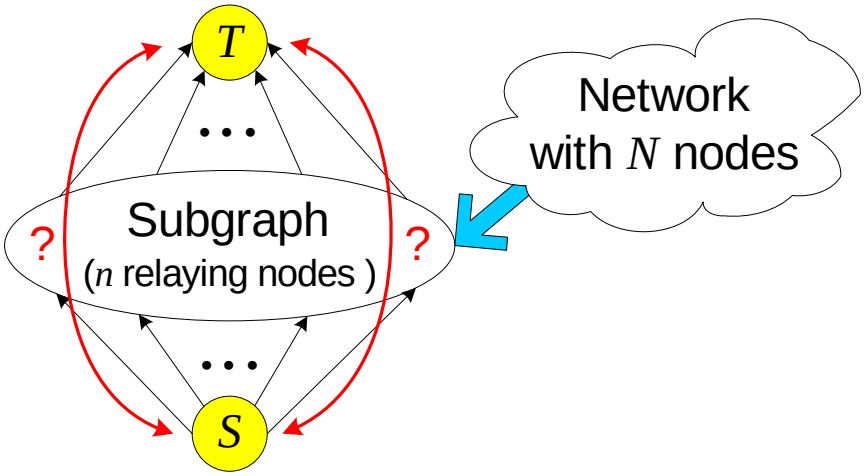


Figure 3.2: There are multiple unicast sessions in the network. Each source node randomly selects n nodes out of N nodes to construct a subgraph.

information. Each adversarial node is assumed to know and follow the protocol. We do not consider active attacks such as corruption or dropping of packets by relay nodes. The combination of this work with erasure and error correction coding to combat such active attacks is a topic for future work.

3.2.2 Anonymity Metric

We use a conditional entropy to measure the anonymity performance, as proposed in [23,72]. Specifically, we consider the entropy $H(S,T|A)$ which quantifies the amount of remaining uncertainty about the source-sink pair (S,T) given the adversary's observations A, or the amount of additional information required to identify pair (S,T). As shown in Section 3.3.3, there can be tension between optimizing anonymity of the source versus anonymity of the sink. Thus, optimization of the joint entropy $H(S,T)$ provides a solution that balances between protecting the source and sink identities.

3.3 Subgraph Construction Phase

In this section, we focus on the subgraph construction phase. We propose a scheme that uses coding, rather than a PKI, to enable a source node to anonymously set up a subgraph and send a small secret message (e.g., a cryptographic key) to the sink.

As in the "slicing the onion" scheme [45] described in Section 3.1, we consider a rect-

angular layered subgraph consisting of l layers of d nodes each. The overlay links between nodes in two consecutive layers form a complete bipartite graph, and the coding scheme allows each node to decode its next hop routing information only if it receives messages from all of its neighbors in the previous layer. There are no overlay links between nodes that are not in successive layers. To prevent nodes from deducing information about their position in the subgraph from their in-degree or out-degree, the source sends from d distinct IP addresses.

Our coding scheme differs from that of [45], and we provide a formal information theoretic security characterization. In particular, we show that as long as the adversary does not control a complete cut between the source and sink, the adversary gains no information about the connections in the subgraph other than the one-hop connectivity information provided to each node to specify its operation. Additionally, we consider optimization of the subgraph parameters and show that randomization of these parameters can improve the anonymity of the system and the resource utilization at the same time. We also address the problem of constructing reverse paths anonymously.

3.3.1 Coding Scheme

In this subsection we describe the coding scheme and characterize its information theoretic security properties.[3]

Let the layers of the subgraph be indexed in increasing topological order starting from the source layer. The packet contents are constructed recursively for each consecutive layer, starting from the last layer. Consider nodes $\{u_1, \ldots, u_d\}$, $\{v_1, \ldots, v_d\}$, and $\{w_1, \ldots, w_d\}$ in three successive layers $k-1$, k, and $k+1$, respectively. A node v_j $(j = 1, \ldots, d)$ has upstream neighbor nodes u_i $(i = 1, \ldots, d)$ and downstream neighbor nodes w_i $(i = 1, \ldots, d)$. Let the packet going from node x to y be represented by a vector g_x^y of symbols from a finite field $\mathbb{F}_q$.

The message intended for x consists of, in order, a last-hop flag ψ_x, a sink-flag ϕ_x, a secret θ_x for x (e.g., cryptographic key) if it is a sink, and packets to be forwarded further. The last-hop flag indicates whether the node is located at the end of the subgraph. The

[3]The results of this subsection generalize straightforwardly to any (not necessarily rectangular) layered subgraph where links between nodes in two consecutive layers form a bipartite graph, though, as noted above, the uniformity of node in-degrees and out-degrees is useful to prevent nodes from deducing information about their position in the subgraph from their in-degree or out-degree.

sink-flag indicates whether a node is a sink and has a secret θ_x intended to it; if it is not not a sink, θ_x consists of random symbols and contains no valid information. To simplify notation, let h_x denote the private information for node x, that is, (last-hop flag, sink-flag, secret) (i.e., $h_x \triangleq (\psi_x, \phi_x, \theta_x)$). The protocol ensures that each node v_j can decode its message $(h_{v_j}, g_{v_j}^{w_1}, \ldots, g_{v_j}^{w_d})$ by summing together the contents of all its received packets $(g_{u_1}^{v_j}, \ldots, g_{u_d}^{v_j})$.

The packet contents are defined recursively as follows. If k is the last layer ($k = l$), then $\psi_j = 1$ and v_j does not have any outgoing packets. Therefore, the message for v_j consists only of the private information for v_j. The contents of the packets transmitted from layer $k-1$ to k are defined as

$$\begin{aligned} g_{u_i}^{v_j} &= \left[v_j,\ K_{u_i}^{v_j}\right],\ i = 1, \ldots, d-1 \\ g_{u_d}^{v_j} &= \left[v_j,\ h_{v_j} - \textstyle\sum_{i=1}^{d-1} K_{u_i}^{v_j}\right] \end{aligned} \tag{3.1}$$

where each $K_{u_i}^{v_j}$ is an independent and uniformly distributed random vector of symbols from $\mathbb{F}_q$ of length equal to the message h_{v_j}. If k is not the last layer ($k \neq l$), $\psi_j = 0$ and we define the packet contents recursively based on the previous layer:

$$\begin{aligned} g_{u_i}^{v_j} &= \left[v_j,\ K_{u_i}^{v_j}\right],\ i = 1, \ldots, d-1 \\ g_{u_d}^{v_j} &= \left[v_j,\ (h_{v_j}, g_{v_j}^{w_1}, \ldots, g_{v_j}^{w_d}) - \textstyle\sum_{i=1}^{d-1} K_{u_i}^{v_j}\right] \end{aligned} \tag{3.2}$$

where each $K_{u_i}^{v_j}$ is an independent and uniformly distributed random vector of symbols from $\mathbb{F}_q$ of length equal to the message $(h_{v_j}, g_{v_j}^{w_1}, \ldots, g_{v_j}^{w_d})$. Fig. 3.3 is an example subgraph illustrating this construction.

Each node v_j in the network strips off its ID from each received packet $g_{u_i}^{v_j}$ and sums over the packets' contents to decode its message: $\sum_{i=1}^{d-1} K_{u_i}^{v_j} + (h_{v_j}, g_{v_j}^{w_1}, \ldots, g_{v_j}^{w_d}) - \sum_{i=1}^{d-1} K_{u_i}^{v_j} = (h_{v_j}, g_{v_j}^{w_1}, \ldots, g_{v_j}^{w_d})$.

Note that the size of the packet contents decreases with distance from the source. To prevent adversaries from deducing their location within the subgraph based on packet size, we maintain a constant packet size by padding with random symbols. In the following description of the random padding procedure, we will refer to the block of symbols in a packet received by node v that corresponds to its ID and private information as the block

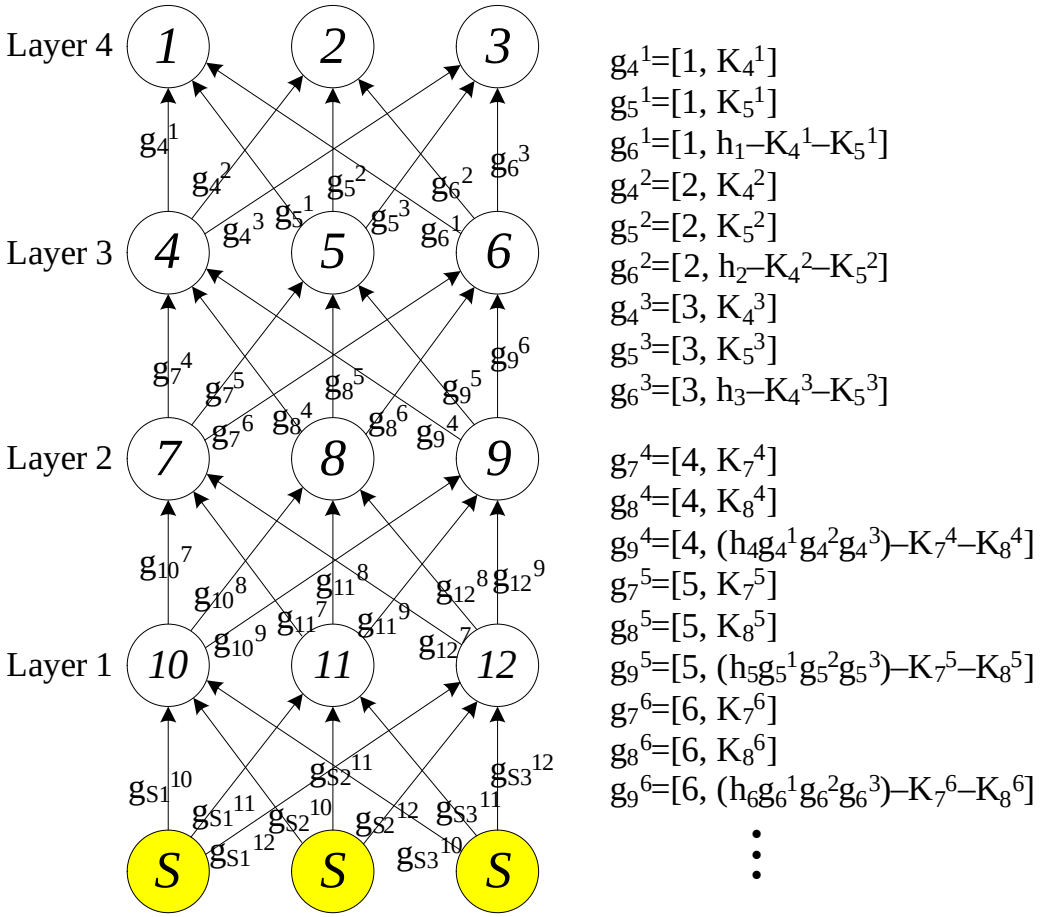


Figure 3.3: Example of a rectangular subgraph with length of 4 and width of 3. g_i^j represents the packet from node i to j. K_i^j denotes an independent random vector of symbols from $\mathbb{F}$. The message consists of (sink-flag ϕ_j, last-hop flag ψ_j, the secret θ_j, and the packets to be forwarded), where $h_j = (\phi_j, \psi_j, \theta_j)$. Note that $\psi_i = 1$ if $i = 1, 2, 3$ and $\psi_i = 0$ otherwise.

corresponding to node v.[4]

Since each message is split into d outgoing packets at each layer, the total number of blocks contained in each message from the source is

$$N_0 = \sum_{i=0}^{l-1} d^i = \frac{d^l - 1}{d - 1}. \tag{3.3}$$

The packet size at each layer is maintained constant at N_0 blocks by padding with random symbols. The actual number of blocks (excluding random padding inserted by non-source nodes) in each packet transmitted from layer k is denoted by N_k.

Let v_i^k denote the i^{th} node in layer k. After a node v_i^k decodes its message from layer $k-1$, it strips off its block and partitions the remaining blocks uniformly into d contiguous segments denoted by $\tilde{g}_{v_i^k}^{(1)}, \ldots, \tilde{g}_{v_i^k}^{(d)}$ which form the contents of packets to be transmitted to the d nodes in layer $k+1$. Accordingly, N_k can be calculated as:

$$\begin{aligned} &\begin{cases} N_0 = \frac{d^l - 1}{d-1} \\ N_k = \frac{N_{k-1} - 1}{d}, \quad 1 \le k \le l \end{cases} \\ &\Rightarrow N_k = \frac{d^{l-k} - 1}{d - 1}, \ 0 \le k \le l \end{aligned} \tag{3.4}$$

Consider the message received by a node v_i^1 in the first layer, and let the blocks in that message be indexed from 1 to N_0; block 1 corresponds to v_i^1. To maintain a constant packet size, node v_i^1 inserts $N_0 - N_1 = d^{l-1}$ blocks of random symbols in each outgoing packet. To ensure that nodes cannot gain information about their location in the subgraph, the scheme is designed such that each node in every layer performs the same operations; nodes cannot distinguish between blocks originating at the source (called original blocks) and random blocks inserted by intermediate nodes. Based on this requirement, the ordering of the blocks and random symbols are determined recursively as described below and illustrated in Fig. 3.4. We describe how to construct from segment $\tilde{g}_{v_i^1}^{(1)}$ of node v_i^1's message the packet $g_{v_i^1}^{v_1^2}$ that is sent to node v_1^2. The packet from node v_i^1 to each node v_j^2 in the second layer is constructed in the same way from segment $\tilde{g}_{v_i^1}^{(j)}$ respectively.

First, block 2, which corresponds to the recipient of the packet v_1^2, is placed at the front of the packet. The remaining blocks of $\tilde{g}_{v_i^1}^{(1)}$ are divided into d contiguous segments of length

[4]Recall that v strips of its ID from each received packet and sums over the packets' contents to decode its message, of which the initial block of symbols corresponds to its private information.

Packet from the source	1	2	3	...	l-2	l-1	l	$n_{l-1,2}$	…	$n_{l-2,2}$	…	$n_{3,2}$	…	$n_{2,2}$	…	$n_{1,2}$	$n_{1,2}$ +1	…	$n_{1,2}$ +l-4	$n_{1,2}$ +l-3	$n_{1,2}$ +l-2	$n_{1,2}$ +l-1	…
Packet from the 1st layer	2	3	4	…	l-1	l			…	$n_{l-1,2}$	…	$n_{4,2}$	…	$n_{3,2}$	…	$n_{2,2}$	$n_{2,2}$ +1	…	$n_{2,2}$ +l-4	$n_{2,2}$ +l-3			…
Packet from the 2nd layer	3	4	5	…	l				…		…	$n_{5,2}$	…	$n_{4,2}$	…	$n_{3,2}$	$n_{3,2}$ +1	…	$n_{3,2}$ +l-4				…
Packet from the 3rd layer	4	5	6	…					…		…	$n_{6,2}$	…	$n_{5,2}$	…	$n_{4,2}$	$n_{4,2}$ +1	…					…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Packet from layer l-3	l-2	l-1	l	…					…		…		…	$n_{l-1,2}$	…	$n_{l-2,2}$	$n_{l-2,2}$ +1	…					…
Packet from layer l-2	l-1	l		…					…		…		…		…	$n_{l-1,2}$		…					…
Packet from layer l-1	l			…					…		…		…		…			…					…

Figure 3.4: The table shows how to relocate each block and where to pad the random symbols. Random symbols are denoted by the yellow boxes. The first block index of the j^{th} part in layer k is $n_{k,j} = k + (j-1)N_k + 1$ for $1 \leq k < l$ and $2 \leq j \leq d$. Note that $n_{l-1,2} = l + 1$ and $n_{l-2,2} = l + d$. The scheme requires to place $n_{i,j}$ right below $n_{i-1,j}$ for $2 \leq i < l$ and $2 \leq j \leq d$. The block relocation can be constructed in a recursive way.

$(N_1 - 1)/d$, each of which is padded with blocks of random symbols in the same positions to form a segment of length $(N_0 - 1)/d$. Thus, we will just describe how to determine the positions of the random blocks in the first segment. Blocks $3, \ldots, l$, corresponding to nodes v_1^k in layers $3 \leq k \leq l$ respectively, are placed in order at the front of the segment. In the message received by v_1^k, the first block index is k and the number of original blocks in each segment is N_k. Therefore, the first block index of the j^{th} segment that is forwarded to v_j^{k+1} is $k + (j-1)N_k + 1$, denoted by $n_{k,j}$. The positions where each block is moved to can be found in a recursive way: the block $n_{k,j}$ is placed in the original position of the block $n_{k-1,j}$ for $3 \leq k < l$ and $2 \leq j \leq d$. Once all blocks are repositioned, d^{l-1} random symbols are inserted in the empty positions. Fig. 3.5 illustrates this random symbols padding scheme for a rectangular-shaped subgraph of dimensions $(l, d) = (4, 2)$.

Now, we characterize the information theoretic security properties of the signaling scheme against adversarial overlay nodes as well as adversarial overlay links, i.e. paths between overlay nodes that contain an adversarial physical node. We show that if there is a non-adversarial path (or equivalently, no adversarial cut) between the source and the last layer, the signaling is information theoretically secure in that colluding adversarial nodes obtain no information about the subgraph other than their own local connectivity.

Lemma 3.1. *For a uniformly distributed random vector $X \in \mathbb{F}_q^l$ independent from (Y, Z) where Y is an arbitrary vector from $\mathbb{F}_q^l$ and Z is a random vector, $X+Y$ is also a uniformly distributed random vector independent from (Y, Z).*

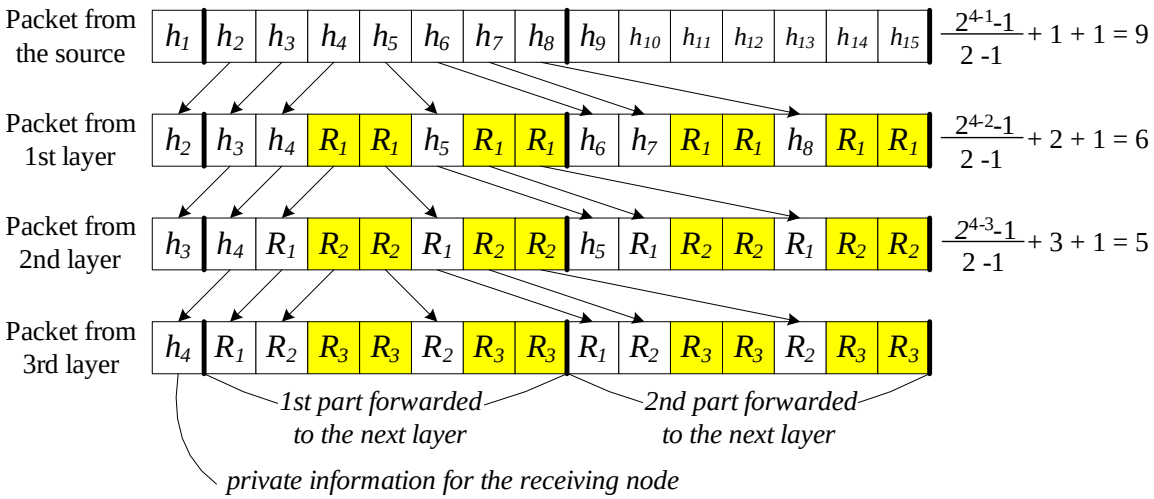


Figure 3.5: Example of random symbols padding scheme for $(l, d) = (4, 2)$. The packet size is proportional to $\frac{d^l-1}{d-1}$ $(= 15)$, and the packet sent from the source includes 15 blocks. To maintain the packet size constant, each relay node pads d^{l-1} $(= 8)$ random symbols (yellow boxes) before forwarding packets. After stripping off the first block, the node performs random symbols padding operation for each of d forwarding parts.

Proof. For any $a, b \in \mathbb{F}_q^l$ and $c \in \mathbb{F}_q^{l'}$,

$$\begin{aligned} & Pr\left[X+Y=b,\ Y=a,\ Z=c\right] \\ =\ & Pr\left[X=b-a,\ Y=a,\ Z=c\right] \\ =\ & Pr\left[Y=a,\ Z=c\right] Pr\left[X=b-a\right] \quad (X \perp (Y,Z)) \\ =\ & Pr\left[Y=a,\ Z=c\right] \frac{1}{q^l} \quad (X \sim \text{Uniform Distr.}) \end{aligned} \tag{3.5}$$

Now, we will show that $X + Y$ is an uniform random vector from $\mathbb{F}_q^l$.

$$\begin{aligned} & Pr\left[X+Y=b\right] \\ =\ & \sum_{i \in \mathbb{F}_q} Pr\left[Y=i\right] Pr\left[X=b-i\right] \\ =\ & \sum_{i \in \mathbb{F}_q} Pr\left[Y=i\right] \frac{1}{q^l} \\ =\ & \frac{1}{q^l} \end{aligned} \tag{3.6}$$

Therefore, we have

$$Pr\left[X+Y=b, Y=a, Z=c\right] = Pr\left[X+Y=b\right] Pr\left[Y=a,\ Z=c\right].$$

This completes the proof. □

Theorem 3.1. *As long as the adversary does not control a complete cut between the source and the last layer, the combined set of packets observed by the adversarial nodes does not reveal any information about the subgraph beyond the information intended for each of the nodes, i.e. one-hop connectivity and private information.*

Proof. The packet g_x^y from x to y contains the next-hop ID y and a payload represented by a vector f_x^y (i.e., $g_x^y = [y, f_x^y]$). The next-hop ID in the packet g reveals only local (one-hop) connectivity that is directly connected to the node. Therefore, we focus on the payload f in this proof. Let the set of payloads of outgoing and incoming packets of a node v be denoted by O_v and I_v, respectively. Note that I_v consists of independent uniform random variables K_w^v and a linear combination of the random variables and the message for v. The message for v is in turn composed of h_v and O_v with next-hop ID's of forwarding packets, where $h_v = (\phi_v, \psi_v, \theta_v)$ is the local information for v. Therefore, I_v does not reveal any further information about connectivity beyond one-hop from the adversaries than O_v (i.e., $\mathbf{I}(\text{connectivity}; I_v) = \mathbf{I}(\text{connectivity}; O_v)$).

We assume that adversaries (both nodes and links) do not form an edge-cut. Suppose that there is a trusted node v receiving and decoding message f_v. We will show $H(f_v|M) = H(f_v)$, where M is a set of messages collected from adversarial nodes and adversarial links.

Let S_k denote the set of adversarial nodes in layer k. Let s_k denote the number of symbols in $O_{v_i^k}$ where v_i^k is the i^{th} node in layer k (s_k are constant for all nodes in a layer). The worst case in which no edge-cut exists is that there exists a single path from the source to the last layer composed of non-adversarial nodes and edges, and all the other nodes and edges are adversarial. To do so, each layer contains one trusted node and $d-1$ adversaries.

First, we consider the case where in each layer k, nodes $S_k = \{v_1^k, \ldots, v_{d-1}^k\}$ are adversarial. In this case, nodes $\{v_d^k : k \in [1,\ l]\}$ are not adversarial and form a trusted path between the source and the sink. It follows from the definition of the protocol that the vector $(O_{v_i^k} : v_i^k \in S_k, i \in [1,\ d], k \in [1,\ l])$ is independently and uniformly distributed over

$\mathbb{F}_q^{\sum_{k=1}^{l} s_k|S_k|}$ and independent from f_v (i.e., all the adversarial messages are uniform i.i.d.).

Now, we consider the general case where the adversarial set S_k can include the node v_d^k. We will show that substituting v_d^k for some node v_a^k $(a < d)$ in S_k does not change the distribution of $(O_{v_i^m} : v_i^m \in S_m, i \in [1,\ d], m \in [1,\ l])$ and its independence with f_v. The outgoing messages of a node v_d^k is $O_{v_d^k} = \{f_{v_j^{k+1}} - \sum_{i=1}^{d-1} K_{v_i^k}^{v_j^{k+1}} : j \in [1,\ d]\}$. We simplify the notation by dropping the layer indexes as $f_j - \sum_{i=1}^{d-1} K_i^j$ for $j \in [1,\ d]$. Applying Lemma 3.1 with $X = (\sum_{i=1}^{d-1} K_i^1, \ldots, \sum_{i=1}^{d-1} K_i^d)$, $Y = (f_1, \ldots, f_d)$, and $Z = (O_{v_i^m} : v_i^m \in S_m, i \in [1,\ d], m \in [1,\ l], (i,m) \neq (a,k))$, we have that $(f_1 - \sum_{i=1}^{d-1} K_i^1, \ldots, f_d - \sum_{i=1}^{d-1} K_i^d)$ is a uniform random vector, independent from $(f_1, \ldots, f_d)$ and $(O_{v_i^m} : v_i^m \in S_m, i \in [1,\ d], m \in [1,\ l], (i,m) \neq (a,k))$. We can proceed recursively in a similar manner to replace $v_d^{k'}$ for some node $v_{a'}^{k'}$ $(a' < d)$ in $S_{k'}$ for other layers k'. This completes the proof. □

3.3.2 Calculation of Conditional Entropy

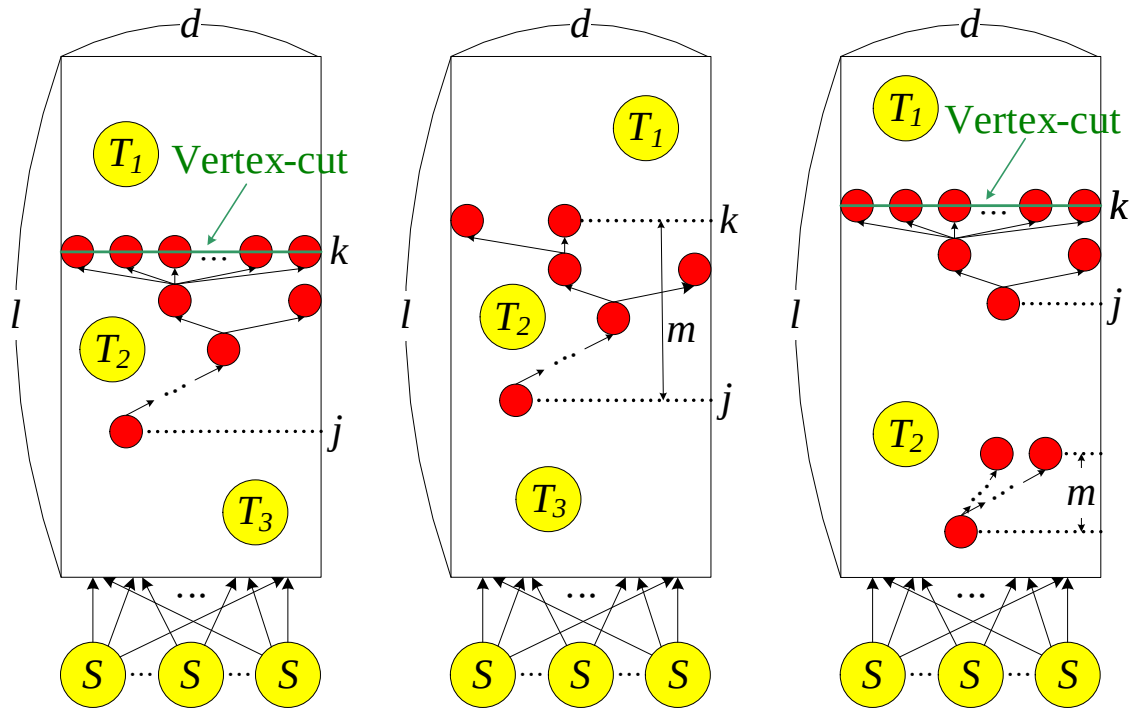
Recall from Section 3.2.1 that there are N nodes in the network, each of which is adversarial independently with probability p. Each source chooses n nodes among N nodes uniformly at random to construct a subgraph. The adversaries aim to identify the source and sink of the communication session that they participate in.

For simplicity, we focus on one source-sink pair (S, T). Let $\mathcal{A}$ be the adversary's observations (i.e. observed messages and local connectivity) corresponding to a realization of adversarial node locations in the subgraph. We are interested in the conditional entropy $(S, T|A)$:

$$H(S, T|\mathcal{A}) = \sum_a \mathcal{P}(a) H(S, T|\mathcal{A} = a) \tag{3.7}$$

where $\mathcal{P}(a)$ denotes the probability of a particular adversarial realization $\mathcal{A} = a$. In the previous section, we showed that the adversaries' observed messages do not reveal any information about the subgraph beyond their local connectivity. Therefore, in our entropy calculation we need only condition on the adversary's local connectivity.

We assume a sufficiently large number of communication sessions (by sending cover traffic if necessary) so that adversaries that are not directly connected to each other do not know if they are in the same subgraph. This means that only connected adversaries can effectively collude to identify a source-sink pair.



(a) Vertex-cut (b) Path of length m (c) Vertex-cut and path

Figure 3.6: A sink is randomly located within the network. (a) adversary-set contains a vertex-cut, which can identify sink T_1, but cannot identify sink T_2 nor T_3; (b) adversary-set does not contain a vertex-cut, but a connected path of length m $(= k - j + 1)$; (c) adversary-set contains a vertex-cut and a path of length m that are disconnected from each other.

3.3.2.1 Calculation of $H(S, T|\mathcal{A} = a)$

Recall that the subgraph forms a rectangular grid of d nodes in each of l layers, not counting the source layer. Two adjacent layers form a complete bipartite graph. The sink is located randomly within the subgraph. We classify three cases corresponding to different configurations of adversarial nodes, as shown in Fig. 3.6.

Adversaries contain a vertex-cut

If adversaries contain a vertex-cut of the network, they can identify all downstream nodes from the vertex-cut. If there are multiple vertex-cuts in the network, we focus on the one closest to the source node, since it can identify more nodes including all the other vertex-cuts. To calculate $H(S, T|\mathcal{A})$, we first calculate $H(T|\mathcal{A})$ and then, $H(S|T, \mathcal{A})$.

In Fig. 3.6 (a), a particular realization of adversaries form a vertex-cut at the k^{th} layer,

and have a connected path down to the j^{th} layer. Since the sink is located randomly within a subgraph, we have three cases (T_1, T_2, and T_3 in Fig. 3.6 (a)): the sink is located either in $[k+1,\ l]$ layers, in $[j,\ k-1]$ layers, or in $[1,\ j-1]$ layers. Let $E \in \{E_1, E_2, E_3\}$ be the event that the sink is located in the three regions, respectively. Due to the existence of a vertex-cut, the sink can be uniformly located among $(l-1)$ layers. Therefore, $\mathcal{P}(E_1|a) = \frac{l-k(a)}{l-1}$, $\mathcal{P}(E_2|a) = \frac{k(a)-j(a)}{l-1}$, and $\mathcal{P}(E_3|a) = \frac{j(a)-1}{l-1}$, where a is a given adversaries realization, and $(k(a), j(a))$ are determined by a.

If the sink node is located in $[k+1,\ l]$ layers (E_1), then adversaries can identify the sink with certainty (i.e., $H(T|\mathcal{A}, E_1) = 0$), since they know all information from the k^{th} to the l^{th} (last) layer.

If the sink is in $[j,\ k-1]$ layers (E_2), the candidate nodes (trusted nodes) in $[j,\ k-1]$ layers are equally likely to be a sink. In average,[5] the number of trusted nodes in $[j,\ k-1]$ is $(k-l)d(1-p)$. Therefore, $H(T|\mathcal{A}, E_2) = \sum_a \sum_{e\in E_2} \mathcal{P}(a)\mathcal{P}(e|a) \log_2 [(k(a) - j(a))d(1-p)]$. The calculation of $\mathcal{P}(a)$ is described in Section 3.3.2.2.

Lastly, if the sink is in $[1,\ j-1]$ layers (E_3), all the rest of trusted nodes can equally be candidates. The number of left trusted nodes is $(N-(l-j+1)d)(1-p)$. Therefore, we have $H(T|\mathcal{A}, E_3) = \sum_a \sum_{e\in E_3} \mathcal{P}(a)\mathcal{P}(e|a) \log_2 [(N-(l-j(a)+1)d)(1-p)]$. In total, the conditional entropy of sink is

$$\begin{aligned} & H(T|\mathcal{A}, E) \\ = & \sum_a \mathcal{P}(a) \frac{k(a)-j(a)}{l-1} \log_2 [(k(a)-j(a))d(1-p)] \\ & + \sum_a \mathcal{P}(a) \frac{j(a)-1}{l-1} \log_2 [(N-(l-j(a)+1)d)(1-p)] . \end{aligned} \tag{3.8}$$

[5]Note that throughout this chapter, for simplicity of calculation, we make use of "average" number of candidate nodes assuming the law of large number (i.e., large system size N in P2P). Due to the logarithmic function in entropy calculation, Jensen's inequality provides an upper bound of the true entropy.

Using $H(T|\mathcal{A}, E)$, we can calculate $H(T|\mathcal{A})$ as follows (chain-rule):

$$\begin{aligned} H(T|\mathcal{A}) &= H(T, E|\mathcal{A}) - \underbrace{H(E|T, \mathcal{A})}_{=0} \\ &= H(T|\mathcal{A}, E) + H(E|\mathcal{A}) \\ &= H(T|\mathcal{A}, E) \\ &\quad + \sum_a \mathcal{P}(a) H\left(\frac{l-k(a)}{l-1}, \frac{k(a)-j(a)}{l-1}, \frac{j(a)-1}{l-1}\right) \end{aligned} \tag{3.9}$$

Now, let's consider the source identification. Recall that all nodes know d and l a priori. If $j = 1$, then the adversaries know they know all layers of a subgraph and therefore, are able to identify the source node with certainty. In this case, $H(S|T, \mathcal{A}) = 0$. If $j \neq 1$, adversaries know that the source node is not located from the j^{th} to the l^{th} layers—since they know source is always located at the beginning of a subgraph—and rule out those nodes. It is the same calculation as the sink anonymity E_3. Therefore, $H(S|T, \mathcal{A}) = \sum_{T,a} \mathcal{P}(T, a) \log_2 [(N - (l - j(a) + 1)d)(1 - p) - 1]$ where -1 comes from the condition on knowledge of sink identity.

Now, we combine the results and calculate $H(S, T|\mathcal{A}) = H(S|T, \mathcal{A}) + H(T|\mathcal{A})$.

Adversaries do not contain a vertex-cut

Even if adversaries do not contain a vertex-cut, they still obtain some information about the source-sink pair from their realization. Since the sink is located randomly within a subgraph, we have two cases (T_2 and $T_1(= T_3)$ in Fig. 3.6 (b)): the sink is located either in $[j,\ k]$ layers where the path is located or outside. Let $E \in \{E_1, E_2\}$ be the event that the sink is in layer $[j,\ k]$ or not, respectively. Since the sink can be uniformly located among l layers, $\mathcal{P}(E_1|a) = \frac{m(a)}{l}$ and $\mathcal{P}(E_2|a) = \frac{l-m(a)}{l}$, where a is a given adversaries realization, and $m(a) = k(a) - j(a) + 1$ is the length of the connected path.

The conditional entropy calculation is analogous to the analysis in Section 3.3.2.1. If the sink is in $[j,\ k]$ layers (E_1), a sink can be equally located among candidates of $(k-j+1)d(1-p)$ trusted nodes in the layers. Accordingly, $H(T|\mathcal{A}, E_1) = \sum_a \mathcal{P}(a) \frac{m(a)}{l} \log_2 [md(1-p)]$. Otherwise ($E_2$), all the rest of trusted nodes can be equally candidate. Hence, $H(T|\mathcal{A}, E_2) =$

$\sum_a \mathcal{P}(a)\frac{l-m(a)}{l}\log_2[(N-m(a)d)(1-p)]$. Now, we have

$$\begin{aligned} & H(T|\mathcal{A},E) \\ =\ & \sum_a \mathcal{P}(a)\frac{m(a)}{l}\log_2[m(a)d(1-p)] + \sum_a \mathcal{P}(a)\frac{l-m(a)}{l}\log_2[(N-m(a)d)(1-p)]\,. \end{aligned} \tag{3.10}$$

Using $H(T|\mathcal{A},E)$ and (3.9), we can calculate $H(T|\mathcal{A})$ as follows:

$$H(T|\mathcal{A}) = H(T|\mathcal{A},E) + \sum_a \mathcal{P}(a) H\left(\frac{m(a)}{l}, \frac{l-m(a)}{l}\right)$$

Let's consider the source anonymity. Suppose that the path length is m $(= k-j+1)$ as shown in Fig. 3.6 (b). Unless $m = l$ or $m = l-1$, the adversaries do not know the location of themselves in the subgraph in contrast to the vertex-cut case. Due to the uncertainty, we calculate the source anonymity in two steps. Let $\mathcal{B}$ be the realization of adversaries who know their location. Recall that for $\mathcal{A}$ adversaries do not know their location except for the cases of $m = l,\ l-1$ or vertex-cut. First, we calculate the entropy conditioned on $\mathcal{B}$, and then convert it to the one conditioned on $\mathcal{A}$.

With probability of $\frac{1}{l-m+1}$, the path starts from the first layer. In this case, adversaries know the source layer and identify the source node, $H(S|T,\mathcal{B}) = 0$. Otherwise, with probability of $\frac{l-m}{l-m+1}$, the path does not start from the first layer. In this case, adversaries know that the source is not located from the j^{th} to the k^{th} layers: $H(S|T,\mathcal{B}) = \sum_{T,b}\mathcal{P}(T,b)\log_2[(N-m(b)d)(1-p)-1]$. The probabilities $\frac{1}{l-m+1}$ and $\frac{l-m}{l-m+1}$ are considered in $\mathcal{P}(b)$.

$$\begin{aligned} & H(S|T,\mathcal{B}) \\ =\ & \sum_{\substack{b \\ \text{starting from} \\ j=1}} \mathcal{P}(b)\log_2[1] + \sum_{\substack{b \\ \text{starting from} \\ j\neq 1}} \mathcal{P}(b)\log_2[(N-m(b)d)(1-p)-1]\,. \end{aligned} \tag{3.11}$$

Note that since a sink is randomly located, the knowledge of adversaries location is not helpful to reduce the uncertainty of the sink location, that is, $H(T|\mathcal{B}) = H(T|\mathcal{A})$. Now, we

derive $H(S,T|\mathcal{A})$ from $H(S,T|\mathcal{B})$, using the chain rule.

$$\begin{aligned} H(S,T|\mathcal{A}) &= H(S,T,\mathcal{B}|\mathcal{A}) - H(\mathcal{B}|S,T,\mathcal{A}) \\ &= H(S,T|\mathcal{A},\mathcal{B}) + H(\mathcal{B}|\mathcal{A}) - H(\mathcal{B}|S,T,\mathcal{A}) \\ &= H(S,T|\mathcal{B}) + H(\mathcal{B}|\mathcal{A}) - H(\mathcal{B}|S,T,\mathcal{A}) \end{aligned} \tag{3.12}$$

$H(S,T|\mathcal{B})$ can be calculated from (3.11). If adversaries at the end layer do not have downstream neighbors, they know they are at the end of the subgraph. However, if adversaries at both ends have neighbors, the path can be located among $l-m$ positions: $H(\mathcal{B}|\mathcal{A}) = \max\{0, \log_2(l-m)\}$.

Let's calculate $H(\mathcal{B}|S,T,\mathcal{A})$. If $m = l$ or $m = l-1$, $H(\mathcal{B}|\mathcal{A}) = H(\mathcal{B}|S,T,\mathcal{A}) = 0$. For the case of $m \leq l-2$, we have the following cases:

1. S is connected to the path: $H(\mathcal{B}|s,t,a) = 0$
2. S is not connected to the path:
   (a) T is not connected to the path:
      i. $m = l-2$: $H(\mathcal{B}|s,t,a) = 0$
      ii. $m < l-2$: $H(\mathcal{B}|s,t,a) = \log_2(l-m)$
   (b) T is connected to the path:
      i. T is connected one hop ahead (toward to the end): $H(\mathcal{B}|s,t,a) = \log_2(l-m-1)$
      ii. T is connected within length m: $H(\mathcal{B}|s,t,a) = \log_2(l-m)$
      iii. T is connected one hop behind (toward to the source side): $H(\mathcal{B}|s,t,a) = \log_2(l-m)$

Each probability is as follows:

1. S is connected to the path: $\mathbb{P}(m,l)\frac{1}{l-m+1}$
2. S is not connected to the path:
   (a) T is not connected to the path:
      i. $m = l-2$: $\mathbb{P}(m,l)\frac{1}{(l-m+1)l}$

ii. $m < l-2$: $\mathbb{P}(m,l)\frac{(l-m-1)^2}{(l-m+1)l}$

(b) T is connected to the path:

i. T is connected one hop ahead (toward to the end): $\mathbb{P}(m,l)\frac{l-m-1}{(l-m+1)l}$

ii. T is connected within length m: $\mathbb{P}(m,l)\frac{(l-m)m}{(l-m+1)l}$

iii. T is connected one hop behind (toward to the source side): $\mathbb{P}(m,l)\frac{(l-m)}{(l-m+1)l}$

where $\mathbb{P}(m,l)$ is defined in (3.14) (for its calculation, refer Section 3.3.2.2). Then, we can calculate $H(\mathcal{B}|S,T,\mathcal{A})$ and $H(S,T|\mathcal{A})$.

Adversaries contain a vertex-cut and other paths

As in Fig. 3.6 (c), adversaries can contain both vertex-cut(s) and path(s), each of which is not connected to the other. If some paths or other vertex-cuts exist in the downstream network of a vertex-cut, they can be ignored in the calculation since they are identified by the vertex-cut closest to the source layer. We take into account a path only if it locates in the upstream network of a vertex-cut if any exists. For each set, we calculate $H(S,T|\mathcal{A})$ and compare to pick the isolated adversaries set providing the smallest $H(S,T|\mathcal{A})$.

3.3.2.2 Calculation of $\mathcal{P}(a)$

To complete the calculation of $H(S,T|\mathcal{A})$ in (3.7), we also need to calculate $\mathcal{P}(a)$. In this section, we derive $\mathcal{P}(a)$ in each scenario.

Adversaries contain a vertex-cut

Consider $\mathcal{P}(a)$ for a vertex-cut and a connected path, as in Fig. 3.6 (a). As mentioned previously, if there are more than one vertex-cut, then we focus on the vertex-cut that is closest to the source node.

Each node can be adversarial with probability p. For the realization in Fig. 3.6 (a), one layer is the vertex-cut (all adversaries in the layer); from j^{th} to $(k-1)^{th}$, each layer has at least one adversary but not all; $(j-1)^{th}$ layer has no adversary; from 1^{st} to $(j-2)^{th}$, there

is no vertex-cut:

$$\mathcal{P}(a) = \begin{cases} p^d \times (1-p^d-(1-p)^d)^{k-j}, \ j=1 \\ p^d \times (1-p^d-(1-p)^d)^{k-j} \times (1-p)^d \\ \quad \times (1-p^d)^{j-2}, \ j \geq 2 \end{cases} \tag{3.13}$$

Adversaries do not contain a vertex-cut

In Fig. 3.6 (b), we want to calculate Prob [$\exists$ a path of length m in a rectangular network of length l and the adversarial set does not contain a vertex-cut] $\triangleq \mathbb{P}(m,l)$.

When we call a path, any path cannot include a vertex-cut. To form a path, each layer should contain at least one adversarial node but not all nodes in a layer are adversarial.

To calculate the probability, we first calculate

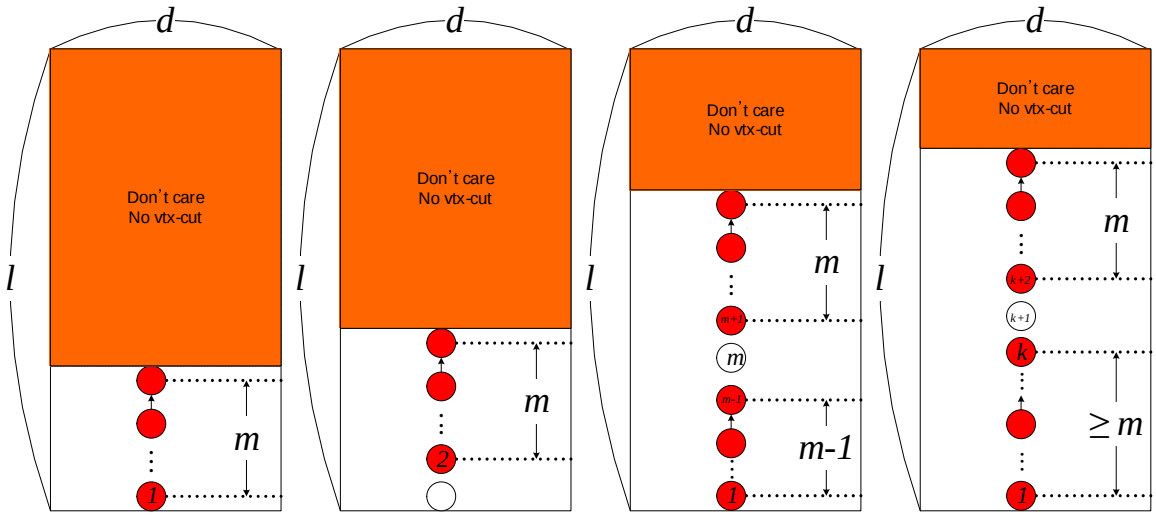
$$\begin{aligned} & P(m,l) \\ \triangleq\ & \text{Prob}\,[\exists \text{a path of length} \geq m \text{ in a rectangular network of length } l] \\ =\ & \text{Prob}\,[\exists \text{a path of length} \geq m \text{ starting in layer 1}] \\ & + \text{Prob}\,[\exists \text{a path of length} \geq m \text{ starting in layer 2} \\ & \quad \text{and no path of length} \geq m \text{ starting in layer 1}] \\ & + \text{Prob}\,[\exists \text{a path of length} \geq m \text{ starting in layer 3} \\ & \quad \text{and no path of length} \geq m \text{ starting in layers} < 3] \\ & + \cdots \\ & + \text{Prob}\,[\exists \text{a path of length} \geq m \text{ starting in layer } l-m+1 \\ & \quad \text{and no path of length} \geq m \text{ starting in layers} < l-m+1] \end{aligned} \tag{3.14}$$

We consider each term in (3.14) as depicted in Fig. 3.7.

(a) If a path of length $\geq m$ starts in layer 1, then the probability is

$$\mathcal{P} = (1-p^d-(1-p^d))^m \times (1-p^d)^{l-m}.$$

From 1^{st} to m^{th} layer, there is at least one but not all adversarial nodes in each layer, and in the rest of layers, no vertex-cut exists.



(a) Starting at $i = 1$ (b) Starting at $i = 2$ (c) Starting at $i \in (2,\ m+2)$ (d) Starting at $i \geq m+2$

Figure 3.7: To calculate each term in (3.14), we consider 3 difference cases depending on the location of a path length of m: (a) the path starts from the first layer; (b-c) the path starts from layer $i \in [2,\ m+1]$, where there is no path in layer $j \in [1,\ i-1]$ with path length no smaller than m; and (d) the path starts from layer $i \geq m+2$, where there can be a path in layer $j \in [1,\ i-1]$ with path length no smaller than m. In (d), $k \geq m$.

(b-c) If a path of length $\geq m$ starts in layer $2 \leq j \leq m+1$, then , then the probability is

$$\mathcal{P} = (1-p)^d \times (1 - p^d - (1-p^d))^m \times (1-p^d)^{l-m-1}.$$

From j^{th} to $(j+m-1)^{th}$ layer, there is at least one but not all adversarial nodes in each layer, and the $(j-1)^{th}$ layer should contain all trustworthy nodes. In the rest of layers, no vertex-cut exists.

(d) If a path of length $\geq m$ starts in layer $m+2 \leq j \leq l-m+1$, then , then the probability is

$$\mathcal{P} = ((1-p^d)^{j-2} - P(m, j-2)) \times (1-p)^d \times (1 - p^d - (1-p^d))^m \times (1-p^d)^{l-(j+m-1)}.$$

From j^{th} to $(j+m-1)^{th}$ layer, there is at least one but not all adversarial nodes in each layer, and the $(j-1)^{th}$ layer should contain all trustworthy nodes. From 1^{st} to $(j-2)^{th}$ layer, there is no path of length $\geq m$ nor a vertex-cut, whose probability is $(1-p^d)^{j-2} - P(m, j-2)$. In the rest of layers, no vertex-cut exists.

For a given l, we calculate $P(m,l)$ for each $m \leq l$, recursively, and construct a table of $P(m,l)$. Then, we can calculate the desired probability as

$$\begin{aligned} & \mathbb{P}(m,l) \\ & \triangleq \text{Prob}\left[\exists \text{a path of length } m \text{ in a rect. network of length } l\right] \\ & = P(m,l) - P(m+1,l) \end{aligned} \tag{3.15}$$

for $m \leq l-1$. For $m = l$, $\mathbb{P}(l,l) = P(l,l)$.

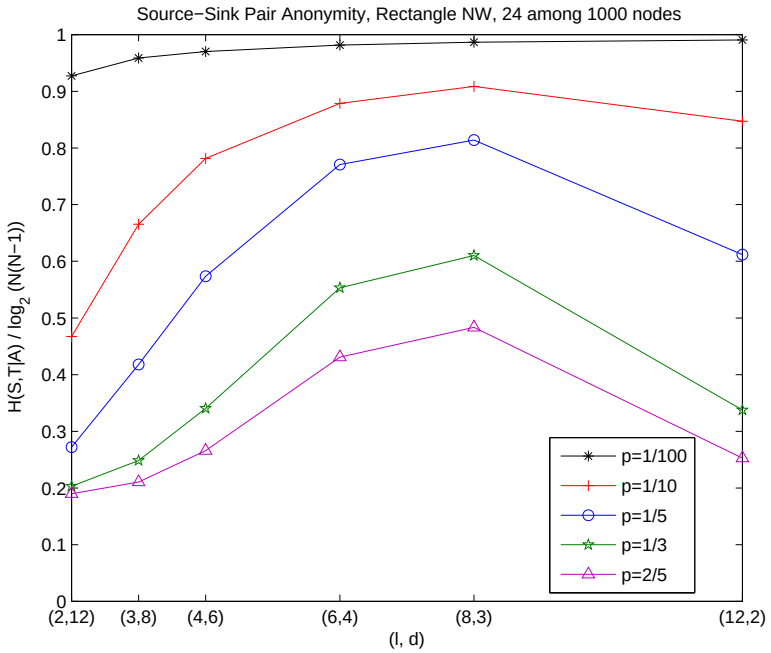
3.3.3 Anonymity of Deterministic Rectangular Networks

3.3.3.1 Experimental Results

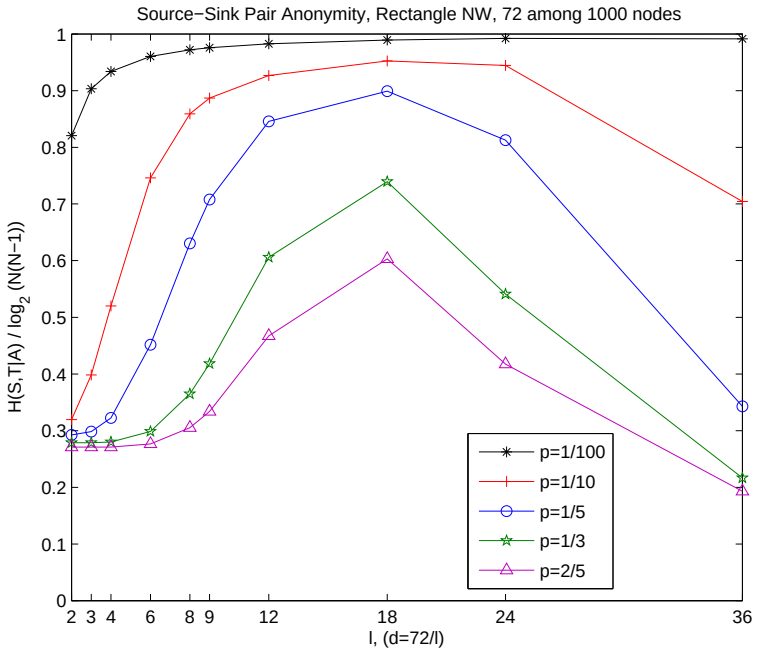
In this section, we present performance analysis of a few example networks with various network parameters (i.e., different (l,d) for fixed $n = l \times d$ and various colluding probabilities p). We consider joint entropy of source-sink pair as well as the separate analyses of source anonymity and sink anonymity.

In Fig. 3.8, we plot normalized conditional entropies of source-sink pair in the subgraphs constrained on $l \times d = 24$ and $l \times d = 72$. Each joint entropy conditioned on adversaries realization is normalized by maximum uncertainty of source-sink pair, that is $\log_2\left[N(N-1)\right]$ when no adversary appears in the subgraph. We observe that the conditional entropy varies depending on the subgraph shape (l,d) for not-too-small colluding probabilities ($p > 0.01$). The anonymity dependence on subgraph parameters is discussed in Section 3.3.3.2. For any probability $p > 0.01$, the best subgraph shapes providing largest conditional entropy are $(l,d) = (8,3)$ and $(l,d) = (18,4)$ for $l \times d = 24$ and $l \times d = 72$, respectively. With extensive simulations, we observe that the optimal shape for a given subgraph size is invariant against colluding probability p and network size N. We can predict the optimal shape efficiently by using a mechanism introduced in Section 3.3.4. When it comes to comparison of different subgraph sizes, the larger subgraph "usually" outperforms the smaller subgraph, but not always. The details are described also in Section 3.3.4.

Now, we present individual analyses of source and sink anonymities separately. In Fig. 3.9, plots of source anonymity and sink anonymity illustrate that there exists a tension between optimizing source and sink anonymity. More precisely, there exists a tension



(a) $l \times d = 24$



(b) $l \times d = 72$

Figure 3.8: Source-sink pair entropy conditioned on adversaries realization. Sink is randomly located within the subgraph. The subgraph contains (a) 24 and (b) 72 nodes randomly selected out of 1000 nodes. Plot of $\frac{H(S,T|\mathcal{A})}{\log_2[N(N-1)]}$ for different (l, d) and p.

between vertex-cut and long path.[6]

For small l and large d, a long path easily appears so that the source anonymity is poor. Note that unless a path spans the entire subgraph, adversaries cannot identify the source correctly. However, they can still rule out all trustworthy nodes connected the path for guessing the source, since they know the source is located at the beginning of the subgraph. Therefore, the longer the adversarial path is, the less uncertainty on the source the adversaries have. The long path also decreases the sink anonymity, even though sink can be located randomly. If sink is connected to the adversarial path, adversaries guess sink candidates among the trustworthy nodes connected to the path. Otherwise, they guess sink candidates among the trustworthy nodes in the entire network.

For small d and large l, a vertex-cut appears in the subgraph frequently. Although sink is randomly located within the subgraph, the sink can be located in the downstream network of a vertex-cut with nontrivial chances, which makes the sink anonymity zero. Even if sink is located in the upstream network of a vertex-cut, adversaries still rule out many trustworthy nodes in the subgraph to guess the sink node, which is the case for the source as well. Therefore, a vertex-cut significantly reduces the sink anonymity and decrease the source anonymity with a fair amount. In Fig. 3.9, we observe the performance pattern for different choice of (l, d) as expected in the description above.

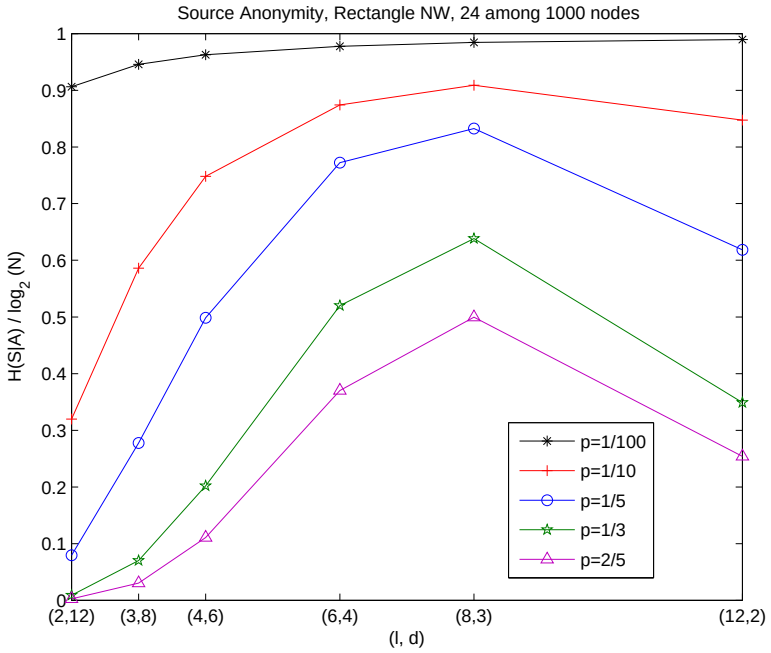
Note that we consider the only case of a sink randomly located within the subgraph. We omitted the analysis and result for the case of deterministic sink location such as a sink located at the end of a subgraph. The source anonymity is not affected from sink location. However, for source-sink pair anonymity and sink anonymity, the performance of deterministic sink location is always worse than that of the random sink location.

3.3.3.2 Performance versus Length and Width

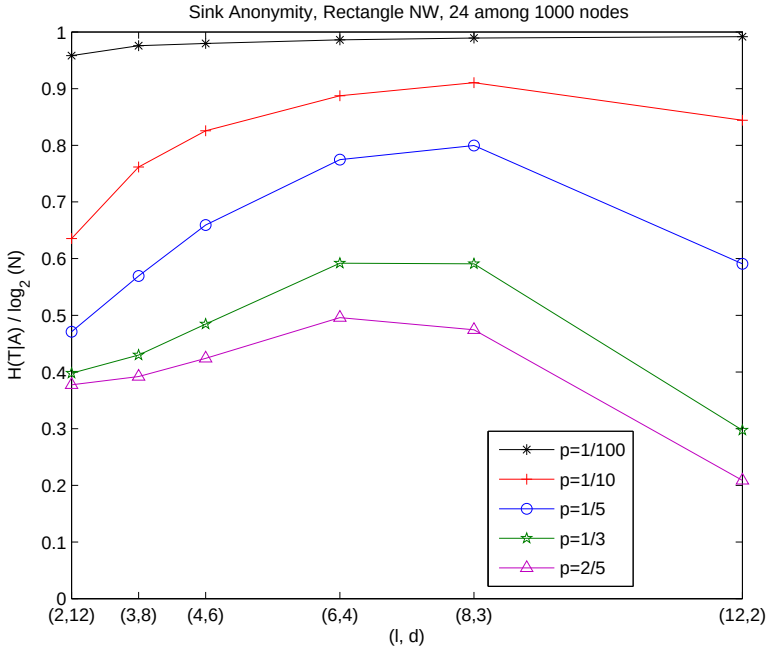
In this section, we discuss the individual effect of subgraph length and width on the source-sink pair anonymity.

For a fixed subgraph width, longer subgraph length brings more chances to form a vertex-cut and lowers the entropy. In contrast, shorter subgraph length is vulnerable as to form adversarial long paths easily. Therefore, we expect that for a fixed width, increasing subgraph length improves the performance until some point and worsens afterward.

[6] Long path is defined as a path whose length is at least half of the subgraph length, $m \geq l/2$.



(a) Source Anonymity



(b) Sink Anonymity

Figure 3.9: Plot of (a) $\frac{H(S|\mathcal{A})}{\log_2 N}$ and (b) $\frac{H(T|\mathcal{A})}{\log_2 N}$ for different (l, d). The plots show the different behaviors of source anonymity and sink anonymity, which is due to the tension between vertex-cut and long-path. Note that the optimal choices of (l, d) are sometimes different for source and sink anonymity.

Fig. 3.10a shows the source-sink pair entropy versus subgraph length for a fixed subgraph width ($l = 4$), where the optimal lengths for different adversarial probabilities are marked in purple circles. For small adversarial probabilities, the optimal length is relatively large so that for the subgraph size of interest (< 100), we can conclude that the performance improves as length becomes longer.

For a fixed subgraph length, narrower subgraph width makes it easier to form a vertex-cut. On the other hand, to form a path each layer should contain at least one, but not all—to avoid a vertex-cut—adversarial nodes with probability of $\beta = 1 - p^d - (1-p)^d$ for $d > 1$. Note that β is a monotonically increasing function over d so that increasing d makes it easier to form a long path and accordingly lowers the entropy. Therefore, we expect too narrow or too wide widths lower the entropy. Fig. 3.10b shows the source-sink pair entropy versus subgraph width for a fixed subgraph length ($l = 10$), where the optimal widths for different adversarial probabilities are marked in purple circles.

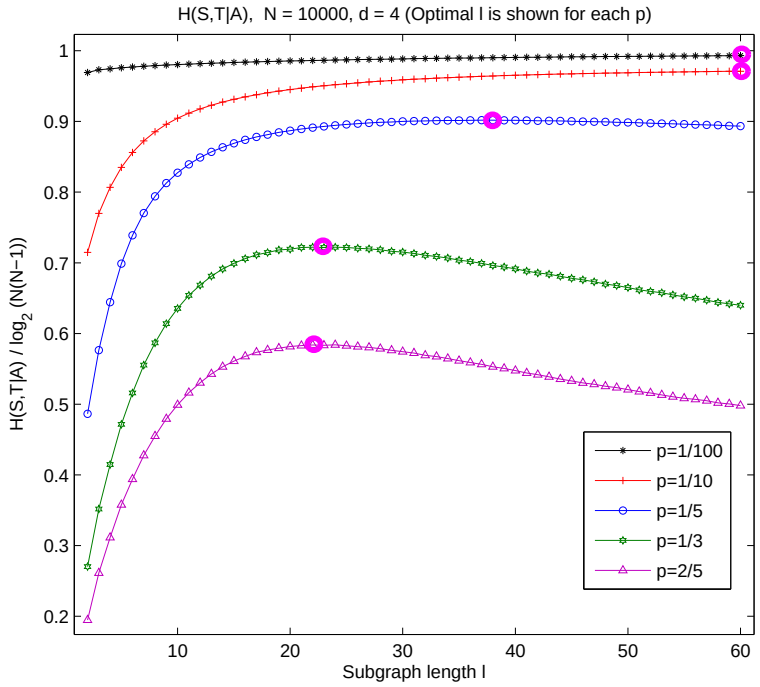
We observe from simulations that for the subgraph length of interest ($l \leq 20$), optimal width is either 3 or 4 in most cases, regardless of N and p. If we extend our interest to longer but impractical subgraph length, the optimal width grows larger gradually (e.g., for $20 < l < 50$, $d_{opt} = 5$ and for $50 < l < 80$, $d_{opt} = 6$ and so on.).

3.3.4 Prediction of the Optimal Subgraph Shape

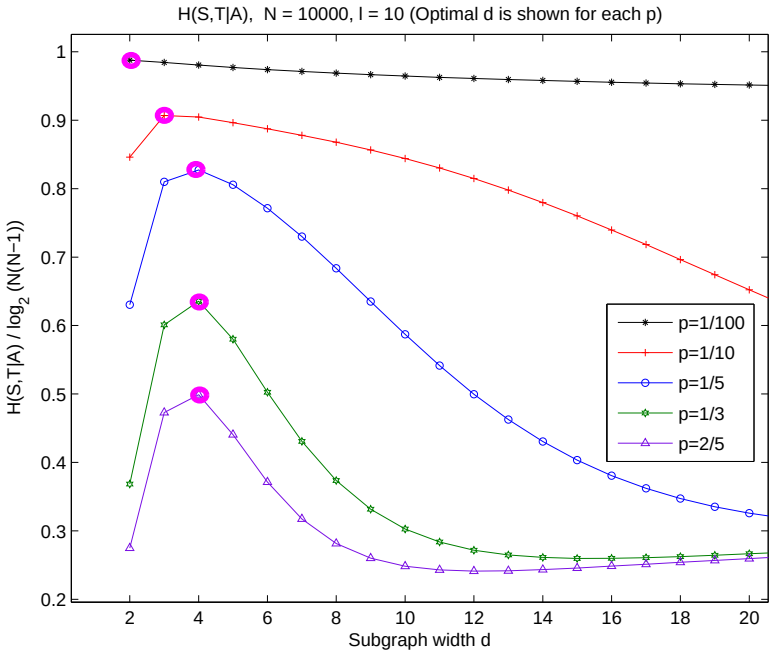
In Section 3.3.3, we show that unless the compromising probability is very small ($p > 0.01$), the entropy differs significantly depending on the subgraph shape for a given subgraph size. One question of interest is how to find an optimal subgraph shape for a given setting (e.g., subgraph size, network size, and compromising probability)? In this section, we answer the question to provide an efficient method of predicting optimal subgraph parameters (length and width) for a given subgraph size. The method is supported by extensive simulations.

Conjecture 3.1. *For a rectangle-shape subgraph, the optimal parameters (length and width) can be calculated approximately as the values for which the probability of an adversarial vertex-cut intersects with the probability of an adversarial path of length at least half of the subgraph length. The optimal length is one of the two divisors of the subgraph size closest to the intersection.*

As we showed previously, the probability of an adversarial vertex-cut is non-decreasing



(a) Fixed width ($d = 4$) and variable length



(b) Fixed length ($l = 10$) and variable width

Figure 3.10: The source-sink pair anonymity versus subgraph parameters: for the rectangle subgraph size of interest (< 100), anonymity is a concave function over length and width. The optimal parameter in each case is marked with a purple circle.

function of subgraph lengths whereas the probability of an adversarial long path is non-increasing function of subgraph lengths. To balance the tension between vertex-cut and long-path, vertex-cut can contain a connected path as before and path can also contain a vertex-cut, which is contrast to the definition of path in Section 3.3.2.1. To illustrate Conjecture 3.1, let's consider the following example:

Example 3.1. *Suppose that we are given a subgraph size* 72 *and compromising probability* $p = 0.25$ *(Fig. 3.11). The subgraph length can be one of divisors of* 72*, that is,* $\{1, 2, 3, 4, 6, 8, 9, 12, 18, 24, 36, 72\}$*. If we plot the probabilities of adversarial vertex-cut and long path versus subgraph length candidates, then the intersection occurs at* 20.0087*. From Conjecture 3.1, we claim that the optimal length would be either* 18 *or* 24 *(the two divisors closest to the intersection). In fact, the optimal subgraph length turns out to be* 24*.*

To support this conjecture, we simulated intensively with different compromising probabilities, network sizes, and subgraph sizes. When it comes to the subgraph size, the entropy does not monotonically increase with the subgraph size (Fig. 3.12), as mentioned in Section 3.3.3.1. If subgraph size n_1 has more divisors than another slightly larger subgraph size $n_2 > n_1$, then n_1 has more degree of freedom for subgraph selection and is more likely have better anonymity. In such cases, when we are given subgraph size n_2, we can use only n_1 nodes to construct a subgraph and simply discard $n_2 - n_1$ nodes. Since we select the best subgraph sizes having the largest normalized entropy up to a point, we call the optimal subgraph sizes "envelope", which is marked in solid red in Fig. 3.12. For a given network size N and compromising probability p, we can enumerate the envelope, and perform simulations with the envelope.

Simulations support Conjecture 3.1 in most of cases. There exist some exceptional cases where Conjecture 3.1 does not hold and optimal lengths are found elsewhere. When we simulate with all possible subgraph size, there are two types of the exceptional cases: 1) the optimal length is found at a value which is the next to the two divisors closest to the intersection; 2) the optimal length has nothing to do with the intersection. However, when we simulate with envelope only, all exceptional cases fall in case 1. Moreover, the difference between entropies from the optimal subgraph shape and the entropies from the suboptimal subgraph shape that is from Conjecture 3.1 is negligible—in most cases, the percentage difference is less than 0.5% and in the worst case observed, the percentage difference is less

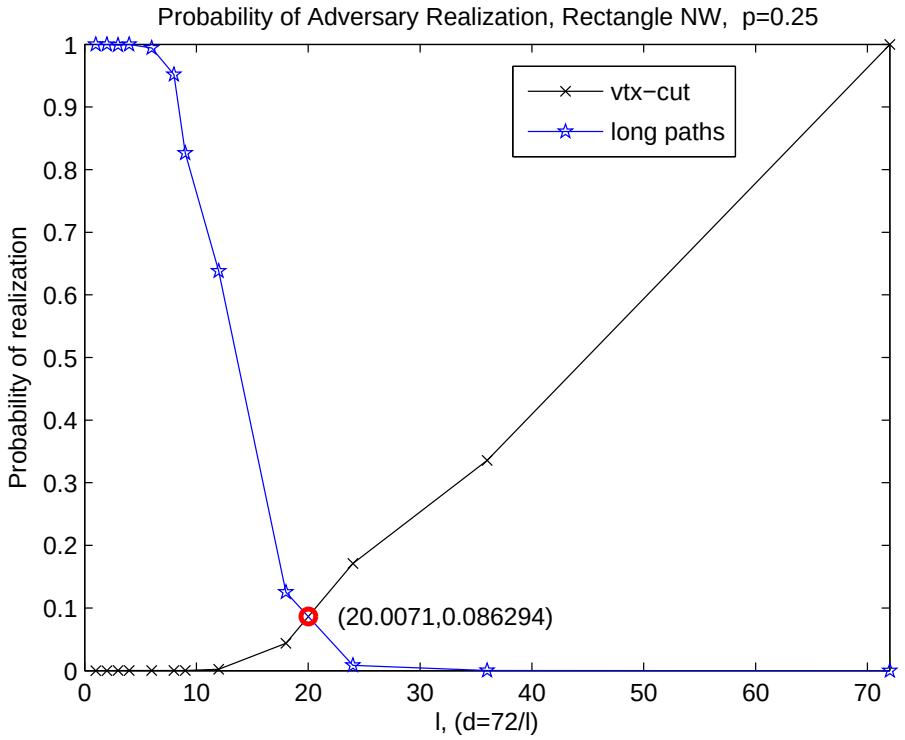


Figure 3.11: Optimal subgraph parameters search for given subgraph size $l \times d = 72$ and compromising probability $p = 0.25$. The probability of vertex-cut and the probability of long-path are close near $l = 20$. We conjecture that the optimal length is close to the intersection, which is indeed the case, $l_{opt} = 24$.

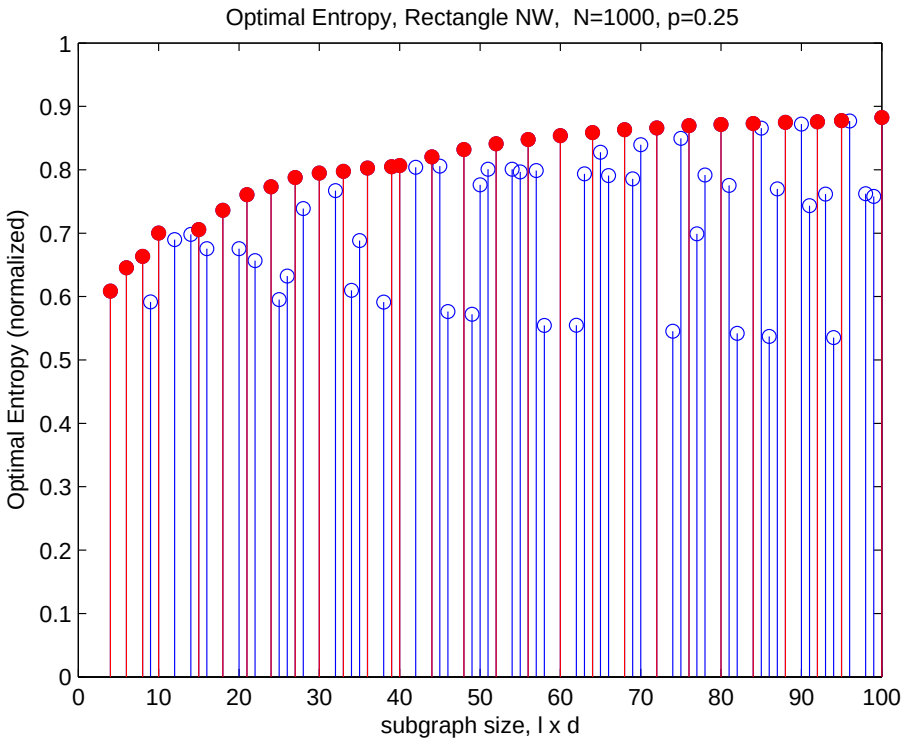


Figure 3.12: We take into account some subgraph sizes, instead of all subgraph sizes. We consider only the "envelope" that provides the largest entropy up to a given subgraph size (marked in solid red).

than 2%. Therefore, the suboptimal subgraph shape chosen from Conjecture 3.1 provides as good performance as the optimal shape.

In conclusion, we have not proved this conjecture yet, but it still provides a very useful rule of thumb to find the optimal subgraph shape efficiently, without calculating the entropies for all possible candidates and sorting them, which is demanding in terms of computational complexity. The method on how to select a subgraph for a given subgraph size is summarized as follows:

For a given subgraph size $n = l \times d$, adversarial probability p, and network size N:

1) Construct an "envelope table" as in Fig. 3.12.

2) Choose the best subgraph size $n' \leq n$ from the envelope table.

3) Use the Conjecture 3.1 and find l_{opt} and d_{opt}.

3.3.5 Randomized Subgraphs

To improve the anonymity, in this section we suggest a randomized strategy and analyze anonymity performance of a subgraph with randomized length. Note that randomizing width is not helpful at all, since all nodes know the width from in-degree and out-degree of flows. It is easily shown using non-negativity of a mutual information that the randomization will increase the entropy of source and sink pair conditioned on adversarial pattern as following:

$$I(S,T;L|\mathcal{A}) = H(S,T|\mathcal{A}) - H(S,T|L,\mathcal{A}) \geq 0 \tag{3.16}$$

From extensive simulations, we show that randomized length simultaneously provides better anonymity (higher entropies) as well as more efficient resource usage (shorter expected length of subgraph and smaller expected subgraph size) as shown in Fig 3.14. In this section, we provide the analysis details and discussion.

3.3.5.1 Calculating Entropy in Randomized Subgraphs

For the deterministic case where length l is fixed, we obtain $H(S,T|L=l,\mathcal{A})$ using the calculation in Section 3.3.2. For the randomized case, we calculate $H(S,T|\mathcal{A})$ by using the

following chain rule:

$$H(S,T|\mathcal{A}) = H(L|\mathcal{A}) + H(S,T|L,\mathcal{A}) - H(L|S,T,\mathcal{A}) \tag{3.17}$$

In (3.17), we can calculate each term as follows. The first term, $H(L|\mathcal{A})$, can be calculated by using conditional probability and total probability theorem:

$$\begin{aligned} H(L|\mathcal{A}) &= \sum_{a} p(a) H(L|\mathcal{A}=a) \\ &= \sum_{a,l} p(a,l) \log_2 \frac{1}{p(l|a)} \\ &= \sum_{a,l} p(l)p(a|l) \log_2 \frac{\sum_l p(l)p(a|l)}{p(l)p(a|l)} \end{aligned} \tag{3.18}$$

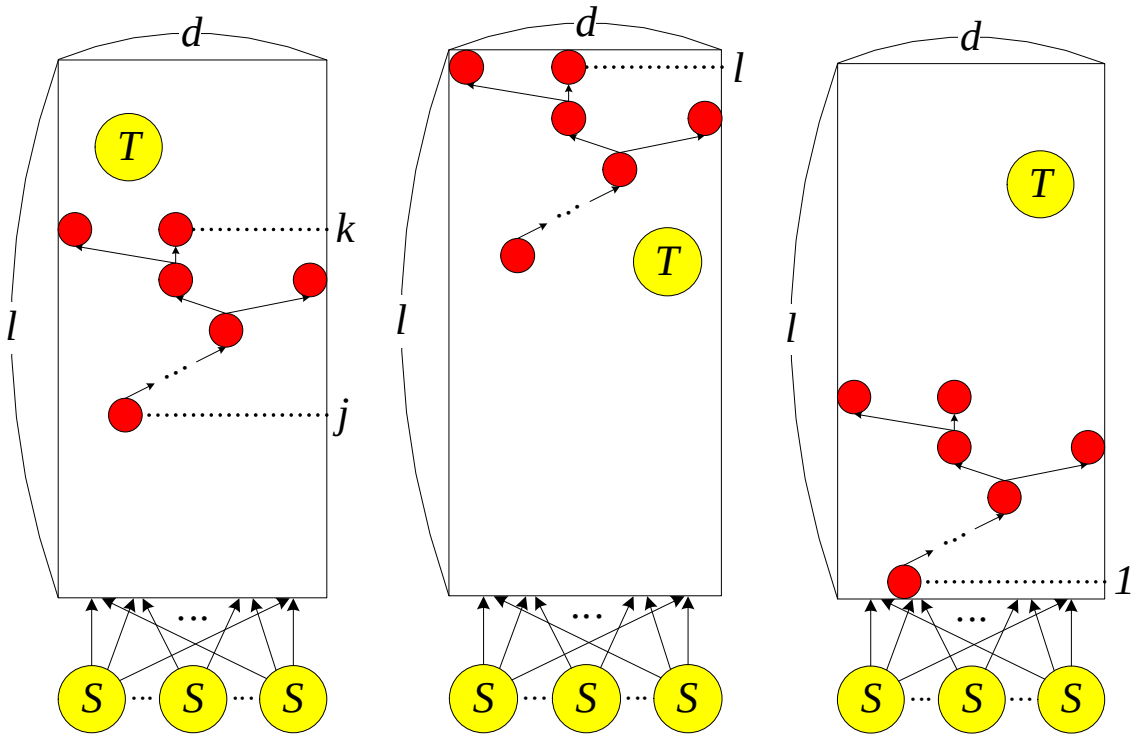
The second term, $H(S,T|L,\mathcal{A})$, can be calculated using conditional entropy:

$$\begin{aligned} & H(S,T|L,\mathcal{A}) \\ =& \sum_{a,l} p(l)p(a|l) H(S,T|L=l,\mathcal{A}=a) \\ =& \sum_{l} p(l) \sum_{a} p(a|l) H(S,T|L=l,\mathcal{A}=a) \\ =& \sum_{l} p(l) H(S,T|L=l,\mathcal{A}) \end{aligned} \tag{3.19}$$

The last term, $H(L|S,T,\mathcal{A})$, is uncertainty of the subgraph length when the source and sink identities with adversarial pattern are given. However, the uncertainty of length can be different depending on the location of adversaries and sink. Therefore, to calculate the last term, we instead consider $H(L|S,T,\mathcal{A},E)$, where $E \in \{E_1, E_2, E_3\}$ is a random variable describing the location of adversarial nodes as shown in Fig. 3.13. Furthermore, we have

$$\begin{aligned} H(L|S,T,\mathcal{A}) &= H(E|S,T,\mathcal{A}) - H(E|S,T,\mathcal{A},L) + H(L|S,T,\mathcal{A},E) \\ &= H(L|S,T,\mathcal{A},E) \end{aligned} \tag{3.20}$$

where the first two terms in the first equation are shown to be zero. Since condition reduces entropy, we have $H(E|S,T,\mathcal{A}) \geq H(E|S,T,\mathcal{A},L)$. To show both terms are zero, we will show $H(E|S,T,\mathcal{A}) = 0$. Note that source and sink identities as well as adversary realization



(a) Float (b) Touch the layer l (c) Touch the layer 1

Figure 3.13: E is a random variable describing the location of successive nodes known to adversaries. (a) the adversarial set touches none of the two ends of a subgraph. This event E_1 with $p(E_1) = \frac{l-m-1}{l-m+1}$, where m is the path length; (b) some adversaries do not have downstream neighbors. This event E_2 has probability $p(E_2) = \frac{1}{l-m+1}$. Since a vertex-cut has a full knowledge of the downstream network, a vertex-cut is classified into this case. For a vertex-cut, $p(E_2) = 1$; (c) Some adversaries are connected to the source, even though they do not know it is the source. This event E_3 has probability $p(E_3) = \frac{1}{l-m+1}$.

are given, which means adversaries know the IP addresses of source and sink, but not their locations. If adversaries identify source IP address from any of their neighbors, adversaries know this is the case for $E = E_3$. If adversaries in the last layer of the chain do not have any downstream neighbors, this is the case for $E = E_2$. Otherwise, $E = E_1$. Therefore, if source and sink identities as well as adversarial pattern are given, there is no uncertainty on the adversaries location E, i.e., $H(E|S, T, \mathcal{A}) = 0$. Now, we calculate $H(L|S, T, \mathcal{A}, E)$:

$$\begin{aligned} & H(L|S,T,\mathcal{A},E) \\ = & \sum_{s,t,a,e} p(s,t,a,e) H(L|S=s,T=t,\mathcal{A}=a,E=e) \\ = & \sum_{s,t,a,e} \sum_{l} p(l)p(a|l)p(e|a,l)p(t|e,a,l)p(s|t,e,a,l) \times H(L|S=s,T=t,\mathcal{A}=a,E=e) \end{aligned} \tag{3.21}$$

Since adversaries in $\mathcal{A}$ know the identities of all nodes connected to them, a path with length of m will know either m, $m+1$, or $m+2$ layers depending on its location and length. A vertex-cut is considered as a path that is connected to the end of a subgraph, since it has full knowledge of downstream network from the vertex-cut. If the vertex-cut is connected to a path starting from the j^{th} layer, then it is regarded as a path starting from the j^{th} and ending at the l^{th} layer.

In (3.21), $\mathcal{L} = \{l_1, l_2, \ldots, l_{max}\}$ and $p(l)$ are given. $p(a|l)$ can be found using mechanisms in Section 3.3.2.2. Calculation of the other terms is described below. We classify cases depending on adversaries location and sink location as follows:

1. If $E = E_1$:

$$\begin{aligned} p(e|a,l) &= \frac{l-m-1}{l-m+1} \\ p(t|e,a,l) &= \frac{m+2}{l}\frac{1}{(m+2)d(1-p)} + \frac{l-(m+2)}{l}\frac{1}{(N-(m+2)d)(1-p)} \\ p(s|t,e,a,l) &= \frac{1}{(N-(m+2)d)(1-p)-1} \end{aligned}$$

   (a) The sink is connected to the adversaries:

$$H(L|S=s,T=t,\mathcal{A}=a,E=e) = -\sum_{l\in\mathcal{L}\ :\ l\geq m+3} p(l)\log_2 p(l)$$

(b) The sink is NOT connected to the adversaries:

$$H(L|S=s,T=t,\mathcal{A}=a,E=e) = -\sum_{l\in\mathcal{L}\ :\ l\geq m+4} p(l)\log_2 p(l)$$

2. If $E = E_2$:

$$\begin{aligned} p(e|a,l) &= \frac{1}{l-m+1} \\ p(t|e,a,l) &= \frac{m+1}{l}\frac{1}{(m+1)d(1-p)} + \frac{l-(m+1)}{l}\frac{1}{(N-(m+1)d)(1-p)} \\ p(s|t,e,a,l) &= \frac{1}{(N-(m+1)d)(1-p)-1} \end{aligned}$$

(a) The sink is connected to the adversaries:

$$H(L|S=s,T=t,\mathcal{A}=a,E=e) = -\sum_{l\in\mathcal{L}\ :\ l\geq m+2} p(l)\log_2 p(l)$$

(b) The sink is NOT connected to the adversaries:

$$H(L|S=s,T=t,\mathcal{A}=a,E=e) = -\sum_{l\in\mathcal{L}\ :\ l\geq m+3} p(l)\log_2 p(l)$$

3. If $E = E_3$:

$$\begin{aligned} p(e|a,l) &= \frac{1}{l-m+1} \\ p(t|e,a,l) &= \frac{m+1}{l}\frac{1}{(m+1)d(1-p)} + \frac{l-(m+1)}{l}\frac{1}{(N-(m+1)d)(1-p)} \\ p(s|t,e,a,l) &= 1 \end{aligned}$$

(a) The sink is connected to the adversaries:

$$H(L|S=s,T=t,\mathcal{A}=a,E=e) = -\sum_{l\in\mathcal{L}\ :\ l\geq m+2} p(l)\log_2 p(l)$$

(b) The sink is NOT connected to the adversaries:

$$H(L|S=s,T=t,\mathcal{A}=a,E=e) = -\sum_{l\in\mathcal{L}\ :\ l\geq m+3} p(l)\log_2 p(l)$$

4. Vertex-cut case:

Suppose that there is a vertex-cut at the k^{th} layer that is connected to a path starting from the j^{th} $(j < k)$ layer. $H(L|S = s, T = t, \mathcal{A} = a, E = e)$ is same as the case of $E = E_2$. The other probabilities are as following:

$$\begin{aligned} p(e|a,l) &= 1 \\ p(t|e,a,l) &= \frac{l-k}{l-1} + \frac{k-j}{l-1}\frac{1}{(k-j)d(1-p)} + \frac{j-1}{l-1}\frac{1}{(N-(l-j+1)d)(1-p)} \\ p(s|t,e,a,l) &= \begin{cases} 1 & \text{if } j = 1 \\ \frac{1}{(N-(m+1)d)(1-p)-1} & \text{otherwise} \end{cases} \end{aligned}$$

3.3.5.2 Result and Discussion

Once we calculate each term in (3.17), we obtain $H(S, T|\mathcal{A})$. Total number of nodes in the network, N, compromising probability, p, and subgraph length set, $\mathcal{L}$ are given network parameters. The control variables include length distribution, $P(l)$, $l \in \mathcal{L}$ and subgraph width, d.

In each simulation, we are given parameters setup such that $N = 10000$, p is fixed among $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, and d is fixed among $\{2, 3, 4, 5\}$. The source first chooses a subgraph length from $\mathcal{L} = \{5, 6, 7, 8, 9, 10\}$ with respect to some given probability distribution $P(l)$, and then constructs a subgraph. In Fig. 3.14, we plot normalized entropies conditioned on adversaries realization. Each rectangle represents a group of a same adversarial probability p, within which subgraph widths vary—one of $\{2, 3, 4, 5\}$. The red star and black circle markers represent deterministic cases with fixed subgraph length $l = 10$ and $l = 9$, respectively. The blue square marker represents a random case with probability distribution $P(l) = \{\frac{1}{32}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}\}$ for $l = \{5, 6, 7, 8, 9, 10\}$, respectively. The randomized case has expected length of 9.031.

In Section 3.3.3.2, we conclude that for the subgraph size of interest ($n \leq 100$), anonymity improves as subgraph length increases for fixed subgraph width. Accordingly, for deterministic cases, $l = 10$ outperforms $l = 9$ (and all the other shorter subgraph lengths that are not shown in the plot) as expected.[7] However, randomized case always outperforms all deterministic cases. Note that the randomized case has smaller expected length (9.031) than

[7] In Fig. 3.14, $l = 9$ outperforms $l = 10$ for the case of $d = 2$. This is the case because narrow width $d = 2$ makes a vertex-cut to appear so easily and therefore, shorter subgraph length is preferred for better anonymity.

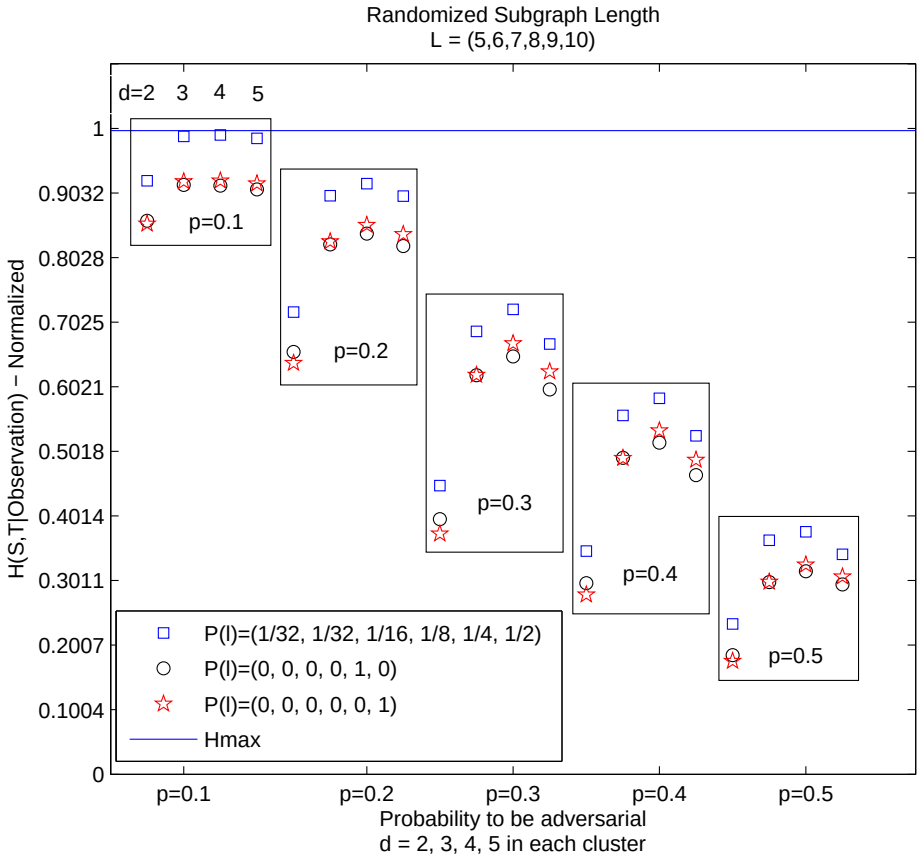


Figure 3.14: Randomized subgraph: a source randomly chooses a subgraph length from a probability distribution, $\mathcal{L}$, and constructs a subgraph. Randomizing subgraph length simultaneously improves the anonymity (larger entropy) and the resource usage efficiency (smaller subgraph size) compared to the deterministic case.

the best deterministic case from $l = 10$. Therefore, randomization improves the anonymity and resource usage efficiency at the same time.

In Fig. 3.14, we also notice that $d = 4$ always achieves the best anonymity in the simulation. Recall from Section 3.3.3.2 that for fixed subgraph length, anonymity is a non-monotonic function with respect to the subgraph width. In the range of subgraph width $5 \leq l \leq 10$, $d = 4$ is optimal for all compromising probabilities $0.01 < p < 0.5$.

Lastly, we consider the optimal probability distribution of $\mathcal{L}$ that maximizes the source-sink pair anonymity. From simulations, we observe that the performance depends on the probability distribution of $\mathcal{L}$. For example, probability distribution $P(l) = \{\frac{1}{32}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}\}$ in Fig. 3.14 outperforms all deterministic cases, but uniform distribution achieves smaller entropy than some of deterministic cases ($l = 9, 10$). The rule of thumb for a good probability distribution is that it should concentrate more on the longer subgraph length and the longer length should have larger probability than the shorter length (i.e., the probability monotonically increases with the length). The expected length should not be too smaller than the longest subgraph length. If the resource usage is sacrificed too much, then the anonymity will worsen as well. Once the criterion is satisfied, the difference in performances of different probability distributions is negligible (less than 1%) whereas the percentage difference between the randomized case and the deterministic case is over 10%.

3.3.6 Reverse Path Construction

In a practical communication system, not only forward path but also reverse path take a significant role. For example, a source wants to open and maintain duplex anonymous communication session with a sink, where all nodes in the network including the sink do not know the source identity (e.g., anonymous web browsing). Furthermore, in some protocol, a feedback from the destination to the source is required (e.g., ACK and NACK signals in TCP). Therefore, it is worthwhile to investigate the reverse-path scenario in an anonymous system that hides both source and sink identities. In this section, we assume that only source knows the identities (IP addresses) of sinks and relaying nodes but no nodes in the network know the source identity.

The original onion routing uses long-lived reply onions for a hidden server [67]. Tor introduces an improved mechanism using rendezvous points for a hidden server [24], which also protects the responder's anonymity as well as the initiator's anonymity. A hidden server

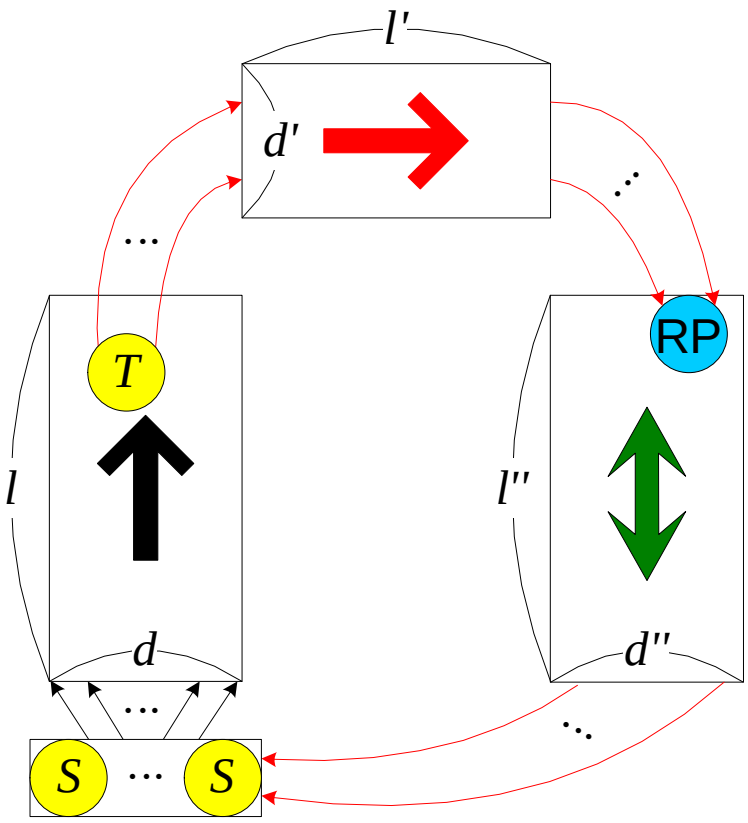


Figure 3.15: Reverse transmission scenario for one source and one sink. The sink T first sends the feedback message to a special node called "rendezvous point" (labeled as RP in blue circle) by constructing an anonymous subgraph (with a red arrow inside) in the same way as the source constructs a subgraph (with a black arrow inside) to send a message to the sink. Between the source and the rendezvous point, there is bidirectional subgraph (with a green bidirectional arrow inside) that is used for rendezvous point to deliver the feedback message to the source.

advertises a contact point and a client negotiates with the contact point to open an anonymous "rendezvous" channel. The idea of rendezvous points for anonymous communication has been introduced for different applications in many papers [29, 36, 37, 43, 64, 66].

Using the concept of a rendezvous point, we can easily construct a reverse path using our protocol without PKI, for which a sink is not required to know the source identity to send back a feedback message (Fig. 3.15). Suppose that a source opens multiple unicast sessions with sink nodes. As described in the protocol design in Section 3.3.1, the source randomly picks relaying nodes and constructs a subgraph for each sink. Each relaying node should appear only once in each subgraph, but can appear in different subgraphs (i.e., subgraphs may have overlapping nodes). In addition, the source constructs one more subgraph for

a special node, called "rendezvous point". The identity of rendezvous point is known to everyone in the network. To improve the security, the system may place more than one rendezvous points and construct corresponding subgraphs. The role of a rendezvous point is to send messages that it receives from other nodes back to the source node. While the source constructs a subgraph for the rendezvous point, all nodes in the subgraph identify their neighbors in the adjacent layers. Therefore, rendezvous point sends message to its neighbors in the previous hop without knowing anything beyond its neighboring layer. By repeating this backward-relaying procedure, the message can be delivered back to the source. Note that each node relays the message in a topologically backward direction, but any node including the ones in the first layer never know the source identity unless adversaries form a path spanning the subgraph. Even vertex-cut nodes do not know the source identity since they know only about downstream network.

Therefore, to send a feedback message back to the source, each sink does not need to know the source identity, but sends the feedback message to the rendezvous point without revealing sink identity. To hide the sink identity, it constructs a subgraph for rendezvous point as if the sink were a source node and the rendezvous point were an intended sink.

In terms of source-sink pair anonymity, the analysis remains same as the forward path. We can apply the same analysis to each subgraph.

3.3.7 Overhead Comparison Between Onion Routing and Protocol Without PKI

In this section, we compare the overhead cost to set up a rectangle subgraph for the protocol with PKI (e.g., onion routing) and the protocol without PKI (e.g., what we propose in this paper). We define the overhead as the total amount of messages traveled across the network to set up a deterministic subgraph.[8] Suppose that the rectangle subgraph has length of l and width of d.

When it comes to the protocol without PKI, each of d duplicated sources sends unique messages to d neighbors. Therefore, the source sends d^2 messages to its neighbors, and these messages will be relayed by intermediate nodes in the subgraph. In each layer, there are d^2 packet transfers. Due to the random symbol padding strategy, the packet size is

[8] Alternatively, we can also define the overhead as the total amount of messages the source node transmits to the network. For this definition, the protocol without PKI has overhead of $O(d^{l+1})$ whereas the onion routing has overhead of $O(ld^2 + l^2d)$. The detailed derivation is omitted.

constant across the layers. We calculate the packet size at the first layer, which does not have padded extra random symbols. Suppose that the IP addresses of nodes are symbols of size q from a finite field $\mathbb{F}$. For a general scenario, we assume that the sink-flag, the last-hop flag, and the secret have size m in total. Recall the notation $h_w = (\phi_w, \psi_w, \theta_w)$. In the $(l-1)^{th}$ layer, the message is $\left[w,\ h_w - \sum_{i=1}^{d-1} K_{v_i}^w\right]$ size of $q+m$, since nodes in the l^{th} do not have any outgoing packets. In the $(l-2)^{th}$ layer, the message is in the form of $\left[v,\ (h_w, g_v^{w_1}, \ldots, g_v^{w_d}) - \sum_{i=1}^{d-1} K_{u_i}^v\right]$ size of $q + m + d(q+m)$. Therefore, we have the following recursive equation for the message size:

$$\begin{aligned} a_1 &= q + m \\ a_i &= q + m + d \times a_{i-1},\ i \geq 2 \end{aligned} \tag{3.22}$$

To solve (3.22), we substitute $b_i = a_i + (q+m)/(d-1)$ for a_i.

$$\begin{aligned} &\begin{cases} b_1 = (q+m)\frac{d}{d-1} \\ b_i = d \times b_{i-1} = d^{i-1} b_1,\ i \geq 2 \end{cases} \\ \Rightarrow\quad & b_i = (q+m)\frac{d^i}{d-1} \\ \Rightarrow\quad & a_i = \frac{q+m}{d-1}(d^i - 1) \end{aligned} \tag{3.23}$$

Note that messages size remains same as the first layer by padding random symbols. Therefore, the total amount of messages traveled across the subgraph is

$$\begin{aligned} & ld^2 \left(\frac{q+m}{d-1}(d^l - 1)\right) \\ \sim\ & O(ld^{l+1}(q+m)) \end{aligned} \tag{3.24}$$

which has an exponential complexity.

Next, for the onion routing, the source sends the neighboring information (both children and parents) to all nodes in the subgraph (ld nodes). If the message is for a node in the i^{th} layer, it has a payload with size of $2dq$ ($2d$ IP addresses) and i headers, each of which has q unit of data. As the message is relayed to the next hop, a header is removed one by one.

Therefore, the total amount of messages traveled across the subgraph is

$$\begin{aligned} &\sum_{i=1}^{l}\sum_{j=1}^{i} d(2d+j)q \\ &= d^2 l(l+1)q + \frac{dq}{2}\left(\frac{l(l+1)(2l+1)}{6} + \frac{l(l+1)}{2}\right) \\ &\sim O(l^2 d^2 q + l^3 dq) \end{aligned} \tag{3.25}$$

which has a polynomial complexity.

Note that the overhead definition considered here does not take into account the effort to distribute and maintain key infrastructure for the onion routing. When the setup cost for cryptographic security is added, the onion routing requires larger overhead on top of the analysis established in this section.

In spite of an exponential overhead cost, our protocol is still useful for public key distribution in any system requiring PKI (cryptographic security). For onion routing, in case where trustworthy third party or secure channel are not available, the system can distribute public key and session keys to target nodes using our protocol. Afterwards, the system may use any other protocol that requires PKI.

In addition, the practical system for the anonymous communication has a relatively small size. In the TOR system, communication path has a few onion routers. The system we consider using the protocol consists of around few tens of nodes. Therefore, since the system size of interest is relatively small, we emphasize that our protocol is a computationally tractable solution.

3.3.8 Hybrid Strategy

As shown in Section 3.3.7, the overhead cost of our protocol to set up a subgraph has an exponential complexity. Since the dominant term is ld^{l+1}, reducing the subgraph length can significantly improve the overhead cost. To reduce the subgraph length, we investigate a "hybrid" scheme that takes advantage of using PKI within our protocol design.

The basic idea is as follows: PKI is costly in terms of key distribution and key management. We aim to balance the usage of PKI in our framework (protocol without PKI) to minimize the use of PKI and to reduce the overhead of our protocol. Accordingly, in the hybrid scheme, not all participating nodes but only a few nodes share a public key with the

source.

Suppose that there are an adversarial vertex-cut in the i^{th} layer and a trusted node v in the j^{th} layer that shares a public key with the source. If node v locates in the downstream network of a vertex-cut ($i < j$), then adversaries in a vertex-cut cannot obtain all information of the downstream network as before, since they cannot decode v's message without knowing v's private key. Instead, they can identify messages of all nodes in layers $k \in [i,\ j]$ (except for v) and identities of all nodes in layers $k \in [i,\ j+1]$. Therefore, a vertex-cut is less successful to reduce the entropy of source and sink pair than the original strategy.

One question of interest is where to place node v that shares a public key with the source to make the use of PKI most effective. If v is located in the upstream network of a vertex-cut, the use of PKI is not helpful at all regardless of whether it is connected to an adversarial node or not. If v is located in the downstream network of a vertex-cut, the use of PKI is more helpful when v is in the layer close to the vertex-cut. Let i and j denote the layer indexes having node v and a vertex-cut, respectively. We define Φ_i as follows:

$$\Phi_i = \begin{cases} i-j, & i > j \\ l-j, & i < j \end{cases}$$

where l is the length of a subgraph. To make the vertex-cut least effective, we need to find i that minimizes Φ_i. Note that a vertex-cut can be equally located over all layers in a rectangular-shape subgraph (i.e., $j \in \{1, \ldots, i-1, i+1, \ldots, l\}$ is equally probable). We show that placing node v in layer $l/2$ is most effective to limit adversaries' knowledge. Since j is probabilistic, we find i that minimizes $\mathbb{E}_j[\Phi_i]$.

$$\begin{aligned} \mathbb{E}_j[\Phi_i] &= \sum_{j=1}^{i-1} \frac{i-j}{l-1} + \sum_{j=i+1}^{l} \frac{l-j}{l-1} \\ &= \frac{1}{2(l-1)} \left(2i^2 - 2li + l^2 - l\right) \\ \frac{\partial \mathbb{E}_j[\Phi_i]}{\partial i} &= 4i - 2l \\ &= 0 \end{aligned} \tag{3.26}$$

Fig. 3.16 and Fig. 3.17 illustrate the anonymity performance with different subgraph shapes (lengths and widths) and compare the original strategy with the hybrid schemes employing one crypto node and two crypto nodes, respectively. As shown in Fig. 3.16 (a)

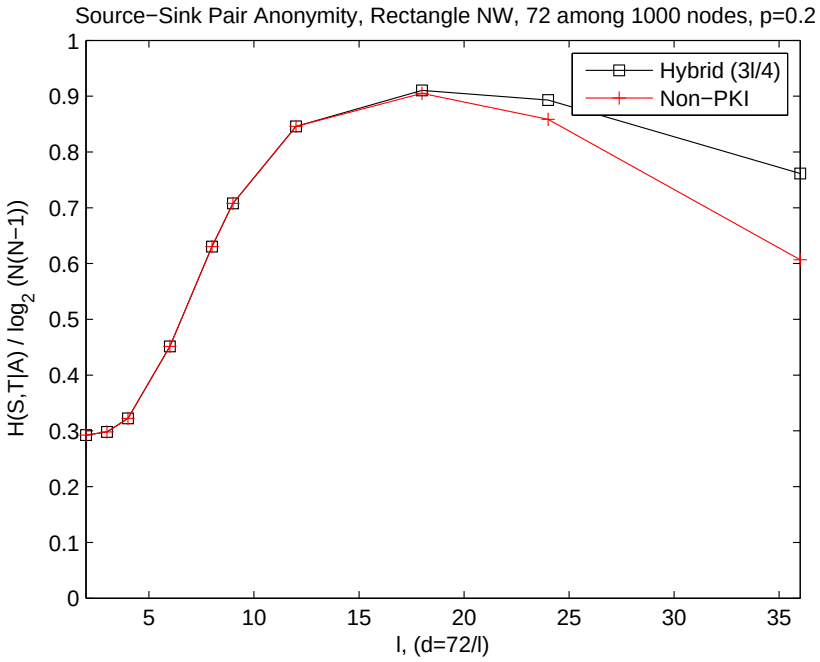
and Fig. 3.17 (a), the performance gain from introducing crypto node(s) is negligible for adversarial probabilities of interest (i.e., $p < 0.2$) even for the case with two crypto nodes. Though the performance gain becomes significant for a long subgraph, the performance achieved from the optimal subgraph shape for a given subgraph size is barely improved. However, for the region of high adversarial probabilities (i.e., $p > 0.3$), the hybrid scheme provides non-trivial performance gain as well as robustness gain against different choices of a subgraph shape (Fig. 3.16 (b) and Fig. 3.17 (b)).

In conclusion, the hybrid scheme does not improve the anonymity significantly for a small adversarial probability, even with multiple nodes sharing a public key with the source. However, for high adversarial probabilities, placing nodes sharing a public key with the source noticeably improves anonymity as well as robustness against different choices of a subgraph shape simultaneously.

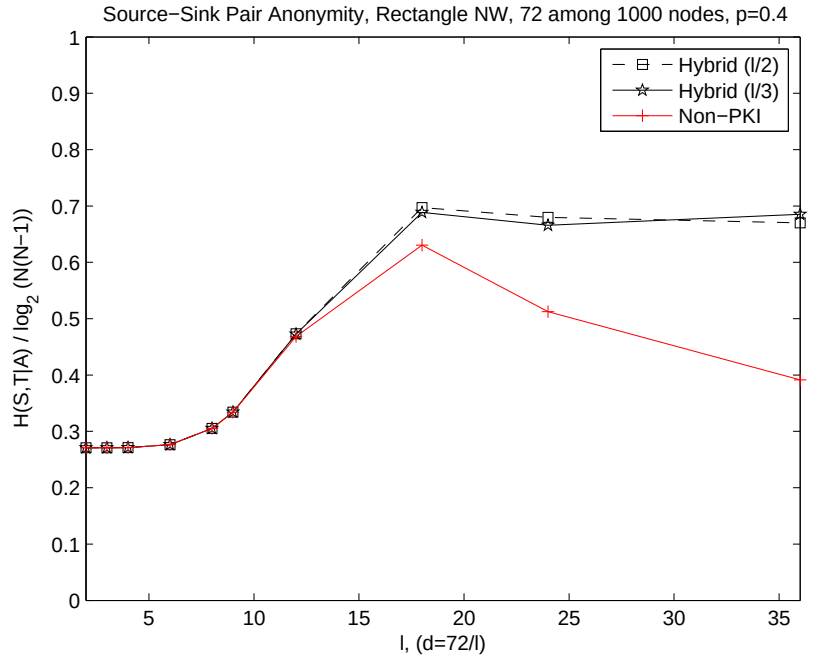
3.3.9 Practical Issues in Implementation

Besides the performance analysis of a suggested protocol, practical issues in implementation—estimating overhead cost, guessing the system parameters that were assumed to be given in the analysis, and so on—are another important topics to investigate. For onion routing, many researchers have studied practical issues to improve the protocol (e.g., improving efficiency and simplicity of TOR system [62]). In this section, we investigate some of important practical issues for our proposed protocol in implementation:

1) To design a P2P anonymous system, we start with choosing N, l, d, and p. Among them, we first focus on the compromising probability p. If an optimal subgraph selection varies depending on the adversarial presence, we need to guesstimate adversarial probability p precisely. However, from the simulations, we observe that the optimal subgraph selection is robust enough to provide a good performance across different p's. Suppose that we consider a case of $N = 1000$ and subgraph size $n = 72$, for which we decide to pick $l = 18$ and $d = 4$. This subgraph shape is optimal for $p \in [0.08,\ 0.5]$. For $p \in [0,\ 0.08)$, this subgraph shape is suboptimal but the performance loss (entropy difference) from the optimal shape is negligible (far less than 1%). In the region of very small $p < 0.1$, the conditional entropies are all large enough to make the entropies insensitive to the choice of subgraph shapes. This is the case for other N and n, which is supported by extensive simulations. In conclusion, if we choose an optimal shape for "not too small" compromising probability (e.g., $p \in [0.1,\ 0.4]$)

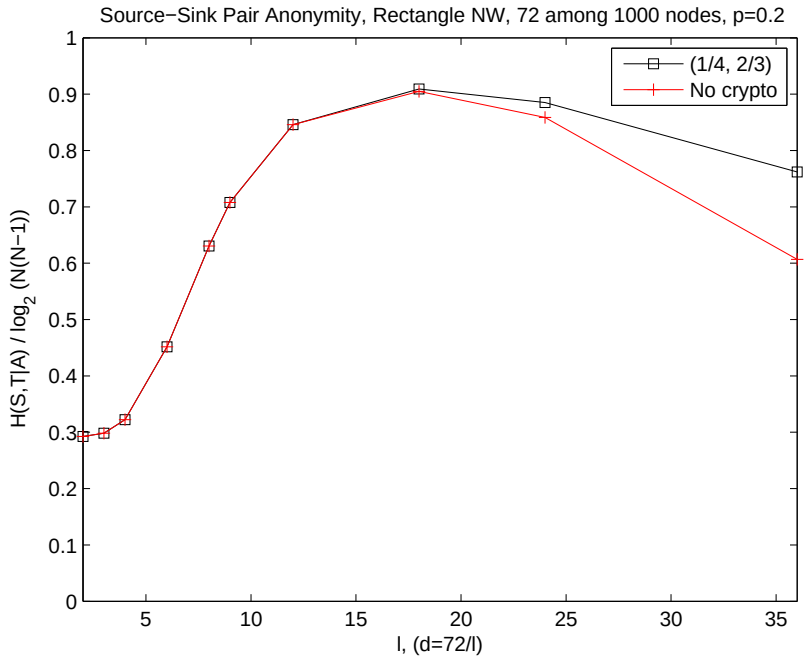


(a) One crypto node at $\frac{3l}{4}$ with $p = 0.2$

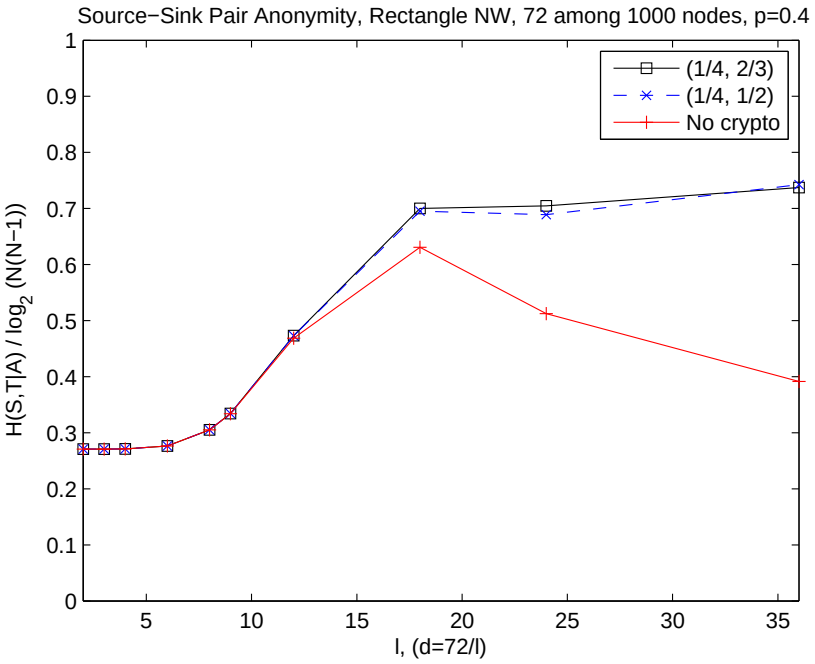


(b) One crypto node at $\frac{l}{2}$ or $\frac{l}{3}$ with $p = 0.4$

Figure 3.16: Hybrid scheme with one crypto node. The performance gain is non-trivial for "long" subgraphs and with high adversarial probability p. For high adversarial probability, hybrid scheme can also provide a robustness against different choices of a subgraph shape.



(a) Two crypto nodes at $\frac{l}{4}$ and $\frac{2l}{3}$ with $p = 0.2$



(b) Two crypto nodes at $\left(\frac{l}{4}, \frac{l}{2}\right)$ or $\left(\frac{l}{4}, \frac{l}{2}\right)$ with $p = 0.4$

Figure 3.17: Hybrid scheme with two crypto nodes. The performance behavior is similar to the case of one crypto node. The additional performance gain beyond one crypto node case is marginal.

using Conjecture 3.1, then it will work well for any compromising probability $p \in [0,\ 0.5]$.

2) We also consider the effect of the system size N. To construct a subgraph, the source should know all the nodes out of which it is going to build the subgraph. The question of interest is how many nodes would need to be in the network to provide anonymity. The number of nodes necessary in the system depends on the target entropy. Once the threshold of entropy is given, we can choose N. However, too large N makes the protocol impractical since the source needs to know all IP addresses of N nodes. To resolve this problem, we take advantage of structured approaches to build scalable P2P system, used in AP3 [56]. In a structured P2P overlays [70,75], every node is assigned a unique identifier called nodeID. To ensure the uniqueness, the identifies are drawn from a large and sparse space. Also, the system generates random keys, each of which is dynamically mapped to a node in the network. Multiple keys can be mapped to one node, but the number of keys mapped to each node is statistically balanced. As in AP3 [56], the source picks the nodes from the network as follows: first, the source chooses a random key K from the ID space. Then, it sends lookup request to the nodes closest to K and decides the node closest to K among the nodes that respond back to the lookup request. By repeating this procedure, the source constructs a subgraph from the large overlay network efficiently.

3) As shown in Section 3.3.7, the overhead cost of our protocol to set up a subgraph has an exponential complexity. Since the dominant term is $d^{l+1}q$, reducing the subgraph length can significantly improve the overhead. To reduce the subgraph length, we suggest a "hybrid" scheme that takes advantage of using PKI in our protocol design. The basic idea is that PKI is costly in distribution and management so that we try to minimize the use of PKI. Accordingly, in the hybrid scheme, not all participating nodes but only a few nodes share a public key with the source. Suppose that there is one node v in the j^{th} layer that shares a public key with the source. If node v locates in the downstream network of a vertex-cut, then adversaries cannot obtain all information of downstream network as before. Instead, since they cannot decode v's message content without knowing v's private key, they can identify messages of nodes up to the j^{th} layer (except v) and identities of all nodes up to the $(j+1)^{th}$ layer. Since a vertex-cut can be equally located over all layers, placing node v in layer $l/2$ is most effective to limit adversaries' knowledge. Therefore, if there is at least one node in layer $l/2$ sharing a public key with the source, the hybrid scheme improves the performance. In other words, for a given anonymity requirement, the smaller subgraph is

sufficient to meet the requirement.

3.4 Data Transfer Phase

In this section, we assume availability of a subgraph setup scheme (either PKI-based as in [24, 67] or coding-based as in the previous section) that can be used to distribute coding/forwarding instructions to each node. We consider the data transfer phase using network coding. We study how the subgraph shape and connectivity affect anonymity and congestion, and seek to design subgraph parameters that provide good tradeoffs between anonymity and congestion.

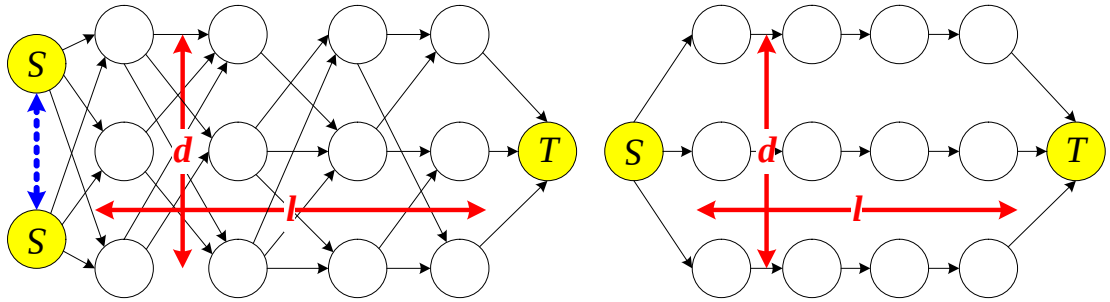
3.4.1 Problem Description

There is a set of N participating nodes, an unknown subset of which may be adversarial and collude. For the adversarial model, we again deal with passive attacks only. Adversaries eavesdrop and observe the packet to reveal source and sink identities information.

Each source constructs a subgraph for anonymous communication to a sink node using a randomly chosen subset of available nodes. In the subgraph construction phase, signaling to set up subgraphs can either rely on cryptography (e.g., create edge, merge edges, split edges, and destroy edge) or use our proposed protocol in Section 3.3. Once the instruction is distributed over nodes, data transfer phase uses network coding for traffic shaping to prevent traffic analysis (i.e., limiting traffic volume on overlay links and nodes). For data security, we use cryptographic keys for end-to-end encryption.

In this part, we provide analysis of 1) anonymity for families of subgraphs and parameters, and 2) congestion arising from traffic shaping constraints. By combining the results into the joint analysis, we study the tradeoff between the anonymity and congestion. We consider two types of rectangular subgraphs parameterized by length, width and connectivity: random subgraph and parallel path subgraph (refer to Fig. 3.18).

For a random subgraph, there are l layers, each of which consists of d nodes. Between the two consecutive layers, nodes are connected with probability r, subject to the following constraints. 1) Each node's in-degree should be at least two $\{2, \ldots, d\}$ so that nontrivial network coding occurs; 2) Each node's out-degree should be at least one $\{1, \ldots, d\}$ to avoid a dead-end. To make the nodes neighboring the source have in-degree at least 2, the source



(a) Random subgraph (b) Parallel path subgraph

Figure 3.18: Family of rectangular subgraphs parameterized by length, width and connectivity. Note that to make the in-degree of source's neighbors two, random subgraph requires duplicated source that can be connected by a secure channel (blue dashed bidirectional link).

has to be duplicated (at least 2 sources). As suggested in Section 3.3, a source can use multiple IP addresses or the duplicated sources share secure channels available between them. Network coding is carried out over the subgraph. In the random subgraph, only end-to-end cryptography is used, i.e., there is no cryptographic transformation at each hop. If one outgoing link is congested, node can shift flow to other outgoing links if available. More precisely, nodes use "as equal as possible" algorithm to split flows over outgoing links (refer to Algorithm 3.1 in Section 3.4.3.1).

For comparison purposes, we consider a parallel path subgraph with the same size. In a parallel path subgraph, there are d parallel paths between the source and the destination, where each path consists of l nodes. Source transmits the routing instruction messages using cryptographic tunneling, equivalent to Tor [24] on each path. In the data transfer phase, cryptographic transformation is used at each hop.

3.4.2 Anonymity

We use an entropy measure for the anonymity performance metric as in Section 3.3, which quantifies the amount of uncertainty about the source-sink pair identities. This is the amount of additional information required to identify the source-sink pair. Let $\mathcal{A}$ be the realization of adversarial nodes. We calculate the entropy of (S, T) conditioned on the adversary's knowledge (i.e., the messages received by adversarial nodes and the identities of nodes that they are connected to) and aim to maximize it (rewriting (3.7)):

$$H(S,T|\mathcal{A}) = \sum_{a} \mathcal{P}(a) H(S,T|\mathcal{A}=a) \tag{3.27}$$

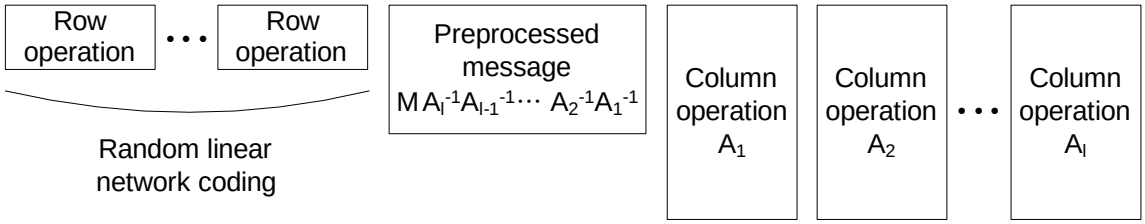


Figure 3.19: In addition to the random linear network coding (row operation), each node performs column operation to the received packets that are preprocessed by the source in a systematic way.

where $\mathcal{P}(a)$ denotes the probability of a particular adversaries realization $\mathcal{A} = a$.

For a parallel path subgraph, the entropy calculation is similar to the calculation in Section 3.3.2 except for handling the vertex-cut. When it comes to the hop-by-hop encryption, a vertex-cut does not reveal any further information than a path including it (i.e., adversaries do not have a benefit from colluding). To see this, we first show adversaries that are not connected do not know if they are in the same subgraph. Suppose that nodes u, v, and w are connected in the order of $u \to v \to w$, where nodes u and w are adversaries. Although u and w know they are connected to the common neighbor v in the order of $u \to v \to w$, they cannot conclude that they are in the same subgraph unless they decrypt the message sent from v to w using v's secret key, since there are many concurrent sessions in the network. Therefore, adversarial nodes that are not connected cannot relate their connections. Accordingly, adversarial nodes in different paths do not know if they serve the same session so that adversaries in a vertex-cut cannot collude. Therefore, for anonymity, we calculate the entropy of the longest adversarial path among all adversarial paths.

For a random subgraph where no cryptographic transformation is used at each hop, we cannot simply assume that the adversaries that are not connected cannot collude. In the network coding system, the messages of the nodes close to the source and the sink are more likely to overlap significantly in the subspace. Therefore, by correlating the messages, adversaries at the end of subgraph surmise they are in the same session, even if they are not connected. To avoid the vulnerability, we propose a novel and inexpensive technique to effectively get rid of correlation between the messages.

In the network coding system, relaying nodes linearly combine the received packets with coefficients randomly chosen from a sufficiently large field (i.e., row operation of the original message) [40, 48, 50]. In the proposed scheme, before the row operation, relaying nodes in

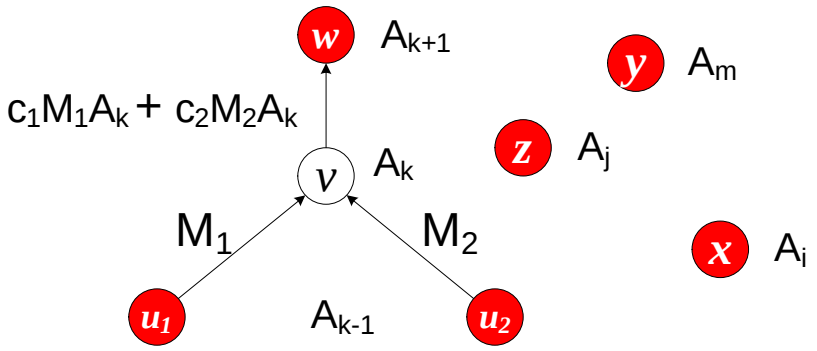


Figure 3.20: Adversaries u_1, u_2, and w that are not connected are hardly able to collude to conclude that they are in the same subgraph.

each layer performs a column operation with respect to a particular matrix A_i, determined by the source. Matrix A_i for the i^{th} layer is instructed by the source in the subgraph setup phase, and a matrix for one layer is different from the matrices for the other layers (i.e., the matrix is different hop-by-hop). The source node generates invertible matrices A_i $(i = 1, \ldots, l)$ independently at random, where the elements of a matrix are drawn from some distribution. In a practical implementation, instead of sending the whole matrix elements to all nodes in the subgraph, the source sends each random seed to the nodes for efficiency (cost reduction). Each random seed must produce an invertible random matrix. Using the random seed, each node can generate corresponding matrix for the column operation. When the source sends out a message M along a path, it preprocesses the message by multiplying $A_l^{-1}A_{l-1}^{-1}\cdots A_2^{-1}A_1^{-1}$. Fig. 3.19 illustrates this scheme. Note that the message in the first layer (close to the source) $MA_l^{-1}A_{l-1}^{-1}\cdots A_2^{-1}$ and the message in the last layer (close to to the sink) M have different subspaces.

We will show how this scheme prevents not-connected adversaries from relating their connectivity. Suppose that a trusted node v in layer k are connected to adversaries u_1, u_2, and w, as illustrated in Fig. 3.20. v receives messages M_1 and M_2 from u_1 and u_2, respectively, and sends $M_v = c_1M_1A_k + c_2M_2A_k$ to w, where c_1 and c_2 are random coefficients from a sufficiently large field, and A_k is the column operation matrix for nodes in layer k. For adversaries u_1, u_2, and w to conclude that they are in the same subgraph, they also need to know A_k. Adversaries try all column operation matrices they possess (A_i, A_j, and A_k in Fig. 3.20) to match A_k. Therefore, only if adversaries know all of M_1, M_2, M_v, and A_k, they can conclude that they serve the same session. However, for adversarial proba-

bility of interest ($p \in [0.1, 0.3]$), it is unlikely to control all of them so that adversaries are hardly able to correlate their messages. Note that larger in-degree demands more degree of freedom for adversaries to control over so as to obfuscate adversaries more, but requires more duplicated sources, which can be expensive. Hence, we focus on in-degree no less than 2 in this chapter.

Now, by virtue of the proposed scheme, we can assume that adversarial nodes that are not connected cannot collude effectively and that they do not know if they are serving in the same subgraph. Furthermore, an adversarial vertex-cut cannot reveal the data or connection information of downstream network unless they possess all column operation matrices in the downstream network. This is equivalent to the case when adversaries form a path in the downstream network. Therefore, a vertex-cut cannot reveal any further information than a path containing it.

The calculation of entropy is also similar to the calculation provided in Section 3.3.2 with differences in calculation of $\mathcal{P}(a)$ in (3.27) and ignoring the vertex-cut issue. The probability of adversarial path $\mathcal{P}(a)$ depends on the connectivity (i.e., connection probability between nodes in the consecutive layers). To form a path, each layer should contain at least one adversarial node and a link exists between a pair of adversaries.

To calculate the probability $\mathcal{P}(a)$, we first consider

$$P(m, l) \triangleq \text{ Prob}\left[\exists \text{a path of length} \geq m \text{ in a rectangular network of length } l\right].$$

Let x_i denote the number of adversarial nodes in layer i. Then, the probability that layer i contains k adversaries is $Pr[x_i = k] = \binom{d}{k} p^k (1-p)^{d-k}$. Let $Pr[i \sim j]$ denote the probability to form an adversarial path from layer i to layer j (and $Pr[i \nsim j] = 1 - Pr[i \sim j]$). Then, we calculate $Pr[i \sim j]$ as follows:

$$\begin{aligned} Pr[i \sim j] &= \sum_{k_i,\ldots,k_j \in [1,\ d]} Pr[x_i = k_i, \ldots, x_j = k_j] Pr[i \sim j | x_i = k_i, \ldots, x_j = k_j] \\ &= \sum_{k_i,\ldots,k_j \in [1,\ d]} \prod_{m=i}^{j} \binom{d}{k_m} p^{k_m} (1-p)^{d-k_m} \prod_{l=1}^{j-1} \left(1 - (1-r)^{k_l k_{l+1}}\right) \end{aligned} \tag{3.28}$$

Now, we calculate $P(m,l)$ as follows (refer to Fig. 3.7):

$$\begin{aligned} P(m,l) = Pr[1 \sim m] + \sum_{i=2}^{m} Pr[(i-1) \nsim i] Pr[i \sim (m+i-1)] \\ + \sum_{j=m+1}^{l-m+1} (1 - P(m, j-1))\, Pr[(j-1) \nsim j] Pr[j \sim (m+j-1)] \end{aligned} \tag{3.29}$$

For a given l, we calculate $P(m,l)$ for each $m \leq l$, recursively, and construct a table of $P(m,l)$. Then, we can calculate the desired probability as

$$\begin{aligned} \mathbb{P}(m,l) &= \text{Prob}\left[\exists \text{a path of length } m \text{ in a rect. network of length } l\right] \\ &= P(m,l) - P(m+1,l) \end{aligned} \tag{3.30}$$

for $m \leq l-1$. For $m = l$, $\mathbb{P}(l,l) = P(l,l)$.

3.4.3 Congestion

Traffic shaping to prevent traffic analysis limits traffic volume on overlay links/nodes. In this chapter, we assume that the constraints arising from traffic shaping are given. The constraints consist of link capacity as well as node out-degree capacity. A link between a pair of nodes has a capacity. When the flow over a link exceeds its capacity, the sending node tries to shift flow to other outgoing links if available. Link congestion is measured by the probability that the flow of a link exceeds its available capacity. On the other hand, if a node has connections to too many nodes, it can be a fingerprint. Therefore, a node can reject or drop connections in excess of an appropriate number for the out-degree capacity. If a node runs out of its out-degree capacity, it first limits the number of connections to other nodes, and ultimately rejects to serve the following communication sessions. Then, other nodes ought to serve the sessions, which results in more load on other links. Therefore, introducing tighter node out-degree capacity induces more link congestion.

We analyze the link congestion for a rectangular subgraph based on extensive simulations using a greedy algorithm (Algorithm 3.2). The details are described in the following subsection.

3.4.3.1 Greedy Algorithm to Calculate the Congestion for Random Subgraphs

Due to the exponential number of combinations to analytically calculate the link congestion probability and accordingly the session congestion probability, we instead calculate them numerically via random simulations. By the sufficient number of iterations, the numerical result approaches the actual value very close (e.g., with $10,000$ times of random simulations, the gap between the actual value and simulation result becomes less than 0.001—proven via central limit theorem [49]). In this section, we provide a greedy algorithm to calculate the probability of congestion for random subgraphs.

The input to the algorithm consists of the size of network $N = |\mathcal{N}|$, the number of sessions S, and the node out-degree capacity ODC, subgraph shape (length l and width d for a rectangular shape), and connection probability r. In each iteration (itr), we randomly select link capacities for all pairs of nodes in the network from some distribution $\mathbf{C}(i,j),\ \forall(i,j) \in \mathcal{N}^2\ (i \neq j)$.

In each session (s), a source picks $l \times d$ relaying nodes at random from the network (which have not run out the node out-degree until the session) and tries to send a unit size of data to the sink anonymously. Within a subgraph, each relaying node selects the neighbors among d nodes in the adjacent layer at random, according to the connection probability and node out-degree capacity. Recall that node in-degree (id) must be at least 2 (for nontrivial network coding) and node out-degree (od) must be at least 1 (to avoid a dead-end). Due to this asymmetry, we first calculate the probability distribution of node in-degree. The probability of node in-degree ($x \geq 2$) is a binomial distribution normalized by $\sum_{x=2}^{d} p(x)$—excluding the cases of 0 and 1 out-degrees.

$$P[id = x] = \begin{cases} \binom{d}{x} \frac{r^x (1-r)^{d-x}}{1-(1-r)^d - dr(1-r)^{d-1}} & 2 \leq x \leq d \\ 0 & x = 0, 1 \end{cases} \tag{3.31}$$

Depending on the node in-degree, each node is connected to id nodes in the upstream neighboring layer at random such that all nodes in the upstream neighboring layer have out-degree at least 1. Once the connections are determined, nodes in the upstream neighboring layer confirms their node out-degrees accordingly. If a node out-degree cumulated up to the current session exceeds ODC, the node adjusts out-degree in the current session such that the total out-degree equals to ODC and all nodes in the downstream neighboring layer

Algorithm 3.1 "As equal as possible" algorithm

Input $x, \mathbf{Cap}, ODC$ ▷ $|\mathbf{Cap}| = \sharp$neighbors
Output Congestion Flag, Load for each link
$Cap(i) \leftarrow \max\{Cap(i), 0\}$ ▷ eligible links only
if $\sum_i Cap(i) < x$ **then** ▷ insufficient link capacities
  Congestion Flag $\leftarrow$ 1
  $\mathbf{Load} = \mathbf{Cap} + \frac{x - \sum_i Cap(i)}{|\mathbf{Cap}|}$ ▷ exceeds capacities
else
  while $x > 0$ **do**
    $\mathbf{A} \leftarrow$ eligible neighbors s.t. $Cap(i) > 0$
    if $\min(\mathbf{Cap}(\mathbf{A})) < \frac{x}{|\mathbf{A}|}$ **then**
      $\mathbf{Load}(\mathbf{A}) \leftarrow \mathbf{Load}(\mathbf{A}) + \min(\mathbf{Cap}(\mathbf{A}))$
      $x \leftarrow x - |\mathbf{A}| \times \min(\mathbf{Cap}(\mathbf{A}))$
      $\mathbf{Cap}(\mathbf{A}) \leftarrow \mathbf{Cap}(\mathbf{A}) - \min(\mathbf{Cap}(\mathbf{A}))$
    else
      $\mathbf{Load}(\mathbf{A}) \leftarrow \mathbf{Load}(\mathbf{A}) + \frac{x}{|\mathbf{A}|}$
      $x \leftarrow 0$
      $\mathbf{Cap}(\mathbf{A}) \leftarrow \mathbf{Cap}(\mathbf{A}) - \frac{x}{|\mathbf{A}|}$
    end if
  end while
  Congestion Flag $\leftarrow$ 0
end if
  return Load and Congestion Flag

have in-degree at least 2. And the node is excluded from the network so that it does not serve any following sessions.

After the connections are determined, we calculate the incoming flows to each node. Each node distributes the incoming flow over the outgoing links using the "as equal as possible" algorithm (Algorithm 3.1). The input to the algorithm consists of incoming flow x, vector of available link capacities between the node v and each neighbors $Cap(i) = \mathbf{C}(v, i) -$ (cumulated load until the previous session) (where i is a neighbor of v), and out-degree capacity ODC. The algorithm outputs the load on each link (to each neighbor) and the congestion flag showing if any link originating from the node exceeds its capacity.

Once all flows over links are calculated, we check if the subgraph (session) is congested. A communication session is declared as congested if the session contains at least one link whose cumulated flow exceeds the link capacity. In a greedy fashion, we consider all communication sessions in turn, and calculate the fraction of the congested sessions to the number of sessions that can be served. Note that due to the node out-degree capacity, a node that runs out of its out-degree is ruled out from the selection for the subgraph construction in the following

sessions. If there left insufficient number of nodes in the network for subgraph construction, all the following sessions cannot fulfill the communication between the source and the sink. Therefore, we keep track the number of sessions that can be supported for given subgraph shape and parameters.

After the congestion probability is calculated, we repeat the iteration with different link capacity assignment.

The greedy algorithm is summarized in Algorithm 3.2.

3.4.3.2 Greedy Algorithm to Calculate the Congestion for Parallel Path Subgraphs

Congestion for the parallel path subgraph can be obtained using the same greedy algorithm in Section 3.4.3.1 with simpler settings.

As before, we randomly select link capacities for all pairs of nodes in the network in each iteration. In each session, a source picks $l \times d$ relaying nodes at random from the network (which have not run out the node out-degree until the session) and tries to send a unit size of data to the sink anonymously. In the parallel path subgraph, all nodes have in-degree and out-degree 1. Since the source node has no prior knowledge of link capacity information nor adversarial nodes realization, it equally distributes load over d paths. Therefore, all nodes have same amount of incoming and outgoing flows $1/d$.

Now, to declare session congested, we check if any link exceeds the link capacity. In a greedy fashion, we consider all communication sessions in turn, and calculate the fraction of the congested sessions to the number of sessions that can be served. Then, we repeat the iteration with different link capacity assignment.

3.4.4 Simulation Results

For given problem parameters (network size $N = 100$, node out-degree capacity $ODC = \{10, 20\}$, and the fraction of adversarial nodes $p = 0.2$), we analyze the anonymity and congestion by controlling subgraph shape (l, d) and subgraph connectivity $r = \{0.2, 0.4, 0.6\}$. For a subgraph shape, we avoid the case of $d = 2$ that results in a degenerated case.[9] Due to the constraint of (in-degree) ≥ 2, any subgraph with $d = 2$ has a complete connection

[9] We also avoid trivial cases of $l = 1$ and $d = 1$. For $l = 1$, all adversaries are connected to source and sink. On the other hand, $d = 1$ results in trivial network coding.

Algorithm 3.2 Greedy algorithm to calculate the probability of congestion

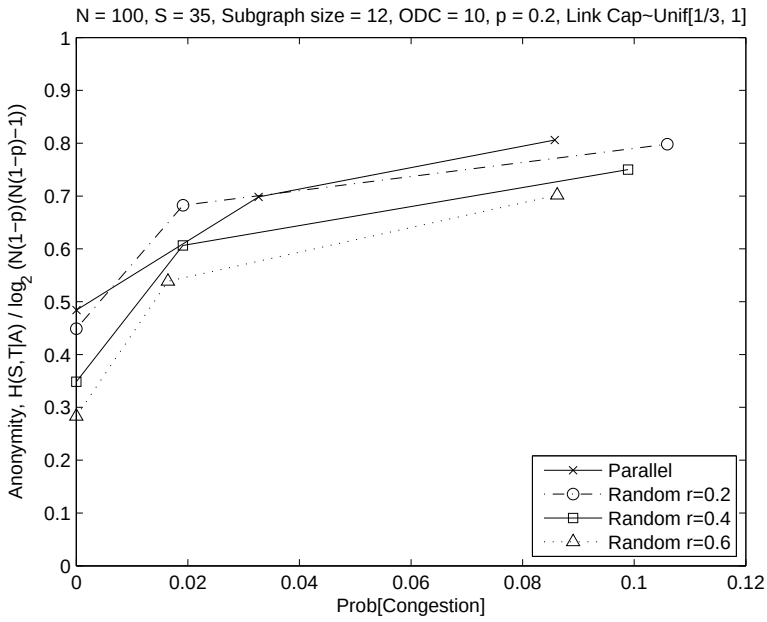
Input $S, \mathcal{N}, l, d, r, ODC, MAXITR$
Output $\Pi \triangleq$ Prob[Congestion]
$\Pi \leftarrow 0$
while $itr \leq MAXITR$ **do**
  $N \leftarrow |\mathcal{N}|$
  $\mathbf{C} \leftarrow RAND(N, N)$ ▷ uniform random matrix
  for $s = 1 \rightarrow S$ **do** ▷ session index s
    $N \leftarrow |\mathcal{N}|$
    if $N < ld$ **then** ▷ insufficient nodes for a subgraph
      $\alpha \leftarrow s - 1$ ▷ $s-1$ sessions can be served in this iteration
      terminate for-loop at session s
    else
      $\alpha \leftarrow s$
      Selects ld nodes from $\mathcal{N}$
      Each node determines id_s from (3.31)
      Each node determines od_s ▷ out-degree in s
      if $\sum_{i=1}^{s} od_i(n) > ODC$ for some node n **then**
        $od_s(n) \leftarrow ODC$
        Remove $od_s(n) - ODC - \sum_{i=1}^{s-1} od_i(n)$ links
        $\mathcal{N} \leftarrow \mathcal{N} \setminus n$ ▷ not appear in session $> s$
      end if
      incoming flows to all nodes in layer $1 \leftarrow 1/d$
      for layer $= 2 \rightarrow l$ **do**
        for all nodes in this layer **do**
          Update incoming flows
          Call Algorithm 3.1
            $\rightarrow$ get link load $\mathbf{y}$ and Congest flag
          $\mathbf{Load} \leftarrow \mathbf{Load} + \mathbf{y}$ ▷ update link load
          $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{y}$ ▷ update available capacity
          Update the session congestion flag
        end for
      end for
    end if
    $s \leftarrow s + 1$
  end for
  $\Pi \leftarrow \left((itr - 1) \times \Pi + \frac{|\{s\text{: congested session}\}|}{\alpha} \right) / itr$
  $itr \leftarrow itr + 1$
end while
  return Π =Prob[Congestion]

between the two adjacent layers (i.e., all nodes have in and out-degree 2). Then, adversaries neighboring sink can identify sink, since they have out-degree 1. Accordingly, the smallest width in our study is $d = 3$, for which the incoming flow to each node is 1/3 in average. To make all nodes able to serve at least one session without exceeding link capacity, we consider the link capacities at least 1/3. In the simulation, the link capacity is chosen independently at random from a uniform distribution in $[1/3,\ 1]$. In addition, we do not consider too high connectivity, which also results in a degenerated case—complete connection between the two adjacent layers. Therefore, we restrict the connectivity to $r \leq 0.6$.

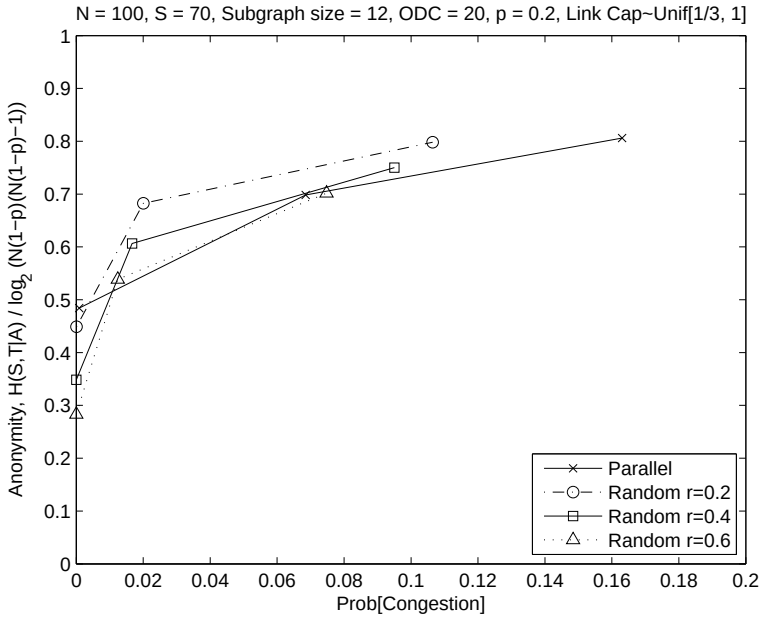
Fig. 3.21 illustrates the tradeoff between anonymity and congestion probability for different out-degree capacities, when the subgraph size is fixed $ld = 12$ (candidate shapes: $(l, d) = \{(2, 6), (3, 4), (4, 3)\}$). Note that the anonymity metric $H(S, T|\mathcal{A})$ is normalized by the number of all possible combinations for source-sink pair among all trusted nodes in the network, that is, $N(1-p)\,(N(1-p)-1)$ in average. A wide (large d) and short (small l) subgraph has good congestion performance but bad anonymity, since the flow is split more (less load on each link and therefore, good congestion) but it is easy to form an adversarial path spanning the whole length, revealing the source and sink identities (bad anonymity).

If out-degree capacity is stringent, small number of sessions can be served since the network runs out of nodes quickly. We observe that for $N = 100$, subgraph size 12, and $ODC = 10$, the smallest number of sessions (among candidates) that run out of available nodes is 38.25 in average when $(l, d, r) = (2, 6, 0.6)$. Therefore, for fair comparison (all candidates serve the same number of sessions without running out of nodes), we consider at most $S = 35$ in this case to compare the performances of different subgraph shapes and connectivity (Fig. 3.21(a)). Likewise, for $ODC = 20$, 73.1 sessions can be served in average when $(l, d, r) = (2, 6, 0.6)$ so that we set the number of sessions to $S = 70$ (Fig. 3.21(b)).

One question of interest is how many sessions can be served for given subgraph size and node out-degree capacity before the exhaustion of available nodes. Table 3.1 shows the numbers of sessions that can be served by a subgraph size of 18 for $ODC = 10$ and 20 before running out of available nodes. For a parallel path subgraph, all nodes have out-degree 1. Hence, the total out-degree of a node over all sessions is same as the number of sessions that the node serves. It is proportion to ld/N so that the number of sessions served without exhaustion is constant regardless of subgraph shape for fixed subgraph size. For a random subgraph, it is more complicated. For dense connectivity (large r), wide subgraph shape



(a) Out-degree capacity = 10



(b) Out-degree capacity = 20

Figure 3.21: Comparison for different subgraph shape, connectivity, and out-degree capacities. (a) The smallest number of sessions (among the candidates) that start running out of nodes is 38.25 in average from $(l, d, r) = (2, 6, 0.6)$ for $ODC = 10$. For a fair comparison, we set $S = 35$. (b) For $ODC = 20$, the smallest number of sessions (among the candidates) is 73.1 in average. For a fair comparison, we set $S = 70$. In both cases, $(l, d, r) = (3, 4, 0.2)$ provides good anonymity and congestion at the same time.

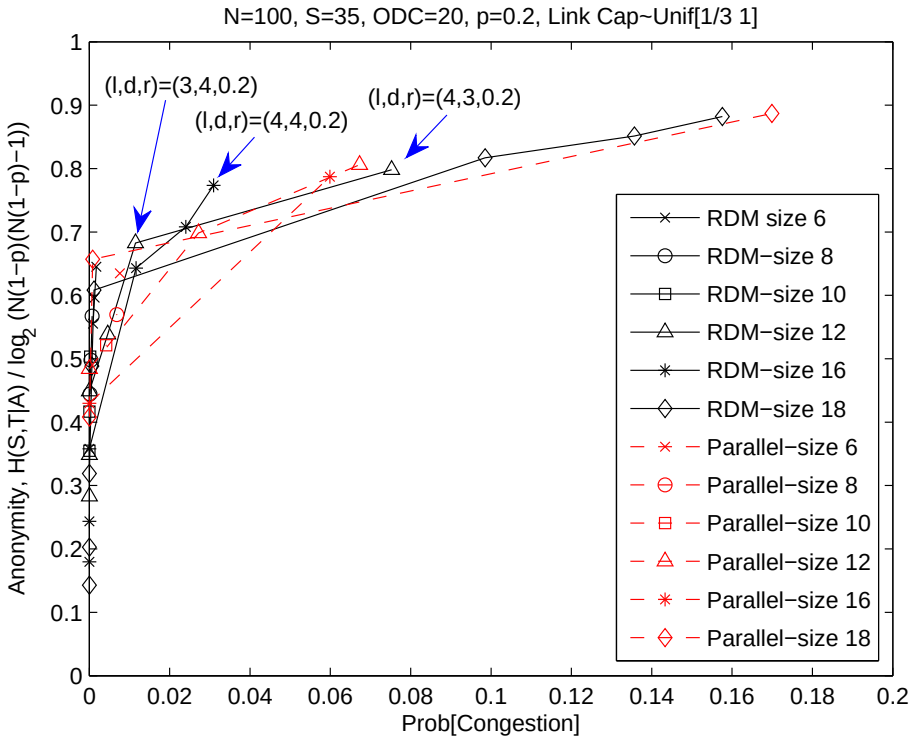


Figure 3.22: To find the best subgraph shape and connectivity, we plot the "envelopes" from each subgraph size. For $ODC = 20$, the smallest number of sessions (among the candidates) that can be served before running out of available nodes is 36.28 in average from $(l, d, r) = (3, 6, 0.6)$. For a fair comparison, we set $S = 35$. In this example, the best subgraph is either $(l, d, r) = (3, 4, 0.2)$, $(l, d, r) = (4, 4, 0.2)$, or $(l, d, r) = (4, 3, 0.2)$ depending on the relative importance between anonymity and congestion.

Table 3.1: The number of sessions that can be served until the exhaustion for given subgraph shape (size of 18), connectivity r, and out-degree capacity $ODC = 10$ and 20

$ODC = 10$	$(l, d) = (2, 9)$	$(l, d) = (3, 6)$	$(l, d) = (6, 3)$
$r = 0.2$	31.17	30.16	29.85
$r = 0.4$	24.59	25.78	28.62
$r = 0.6$	19.59	21.24	27.00
Parallel	54.10	54.10	54.10

$ODC = 20$	$(l, d) = (2, 9)$	$(l, d) = (3, 6)$	$(l, d) = (6, 3)$
$r = 0.2$	61.17	58.82	58.66
$r = 0.4$	47.02	49.83	56.14
$r = 0.6$	36.28	40.58	52.88
Parallel	109.05	109.05	109.05

(large d) consumes more out-degree in each session, and therefore runs out of available nodes quickly. For small r, most nodes are forced to have in-degree 2. Accordingly, out-degrees of nodes concentrate on 2 in average. Note that all nodes neighboring sink have out-degree 1. Therefore, layers 1 to $(l-1)$ dominate the exhaustion of available nodes, that is, $(l-1)d$ nodes are dominant. Therefore, for small r, the subgraph with larger $(l-1)d$ runs out of nodes more quickly (this effect is negligible for large r).

To find the subgraph shape and connectivity that provide the best and well-balanced anonymity and congestion, we plot the "envelopes" selected from each subgraph size in Fig. 3.22. For each subgraph size, we obtain the plot of anonymity versus congestion as in Fig. 3.21. Large entropy and small probability of congestion are preferred so that the point closest to the upper-left corner is most desirable. To obtain the "envelope", we select a point that has shorter distance from the upper-left corner than other neighboring points. For the fair comparison, we choose the number of session $S = 35$ as the smallest number of sessions that can be served until the exhaustion among the candidates.[10] For the scenario of $N = 100$, $S = 35$, $p = 0.2$, and $ODC = 20$, the best subgraph is either $(l, d, r) = (3, 4, 0.2)$, $(l, d, r) = (4, 4, 0.2)$, or $(l, d, r) = (4, 3, 0.2)$ depending on the relative importance between anonymity and congestion.

[10] Among candidates, $(l, d, r) = (3, 6, 0.6)$ can serve at most 19.59 and 36.28 in average until the exhaustion for $ODC = 10$ and $ODC = 20$, respectively—refer to Table 3.1.

3.5 Conclusions

In this chapter, we provided a comprehensive study on peer-to-peer anonymous routing system in two parts: subgraph setup phase and data transfer phase.

For the subgraph construction phase, we have established a problem formulation of peer-to-peer anonymous routing system without using PKI. We suggested a protocol with an information theoretic security characterization. Based on the performance metric of entropy measure and the proposed protocol, we have derived a source-sink entropy for a given adversarial observation in a rectangular network and examined the effect of varying network parameters on the performance metric. We also provide a mechanism to find optimal network parameters for a given problem setting (network size, subgraph size, and compromising probability). Furthermore, we analyze the effect of randomizing the subgraph length and show that the randomization improves the performance and resource usage at the same time. For a hidden service, we consider the reverse (feedback) path scenario, for which we suggest a simple reverse path construction method. Our study includes overhead comparison with other strategy using PKI and discussion for applications. Last but not least, we consider some important practical issues in implementation of our suggested protocol.

For the data transfer phase, we use network coding to hide the identities of communicating parties and less relies on cryptographic keys. Each source node constructs a subgraph to communicate with the sink node anonymously by randomly selecting a subset of available nodes. Furthermore, to hinder traffic analysis by adversaries, we use traffic shaping that limits traffic volume on overlay links and nodes. With these settings, we studied anonymity performance and congestion arising from the traffic shaping constraints in the family of rectangular subgraphs parameterized by length, width, and connectivity. The joint analysis of anonymity and congestion shows the tradeoff and suggests a good subgraph shape and connectivity.

Chapter 4

Robust Network Coding Subgraph Construction under Uncertainty

In this chapter, we study the problem of network construction under uncertainty about link loss rates. For scenarios with uncertainty, we provide a robust optimization-based formulation to construct a single subgraph that works relatively well across all scenarios. We show that this problem is coNP-hard in general for both objectives: minimizing cost of subgraph construction and maximizing throughput given a cost constraint. Our approach finds an approximate solution by introducing path constraints that result in a polynomial time-solvable solution, and outperforms the previous approaches in terms of performance and stability.

4.1 Introduction

In addition to network throughput benefits, network coding can provide robustness to uncertain communication links and uncertain network topologies [30]. In this chapter, we consider the problem of network coding subgraph construction that is robust against uncertainty about link loss rates. For a given set of scenarios specified by an uncertainty set of link loss rates, the goal is to construct a single subgraph that works relatively well across all scenarios. To achieve the goal, an optimization problem with unknown variables is considered. In such problems, the most developed approaches are worst-case analysis and stochastic optimization. Unfortunately, scenario-based stochastic optimization cannot be used unless the uncertainty is probabilistic. On the other hand, worst-case analysis

The work in this chapter was presented at the 42nd Asilomar Conference on Signals, Systems and Computers [13].

generally results in an unnecessarily conservative solution. In this chapter, we follow the robust optimization approach of optimizing the worst-case performance introduced in [7].

Recently, robust optimization research has concentrated on robust convex optimization (including linear optimization). Robust optimization has been applied to a number of network optimization problems. Applegate *et al.* suggested a robust routing which guarantees a nearly optimal utilization of a network against uncertain traffic demands [4]. Mudchanatongsuk *et al.* showed that the network optimization problem with demand uncertainty can be solved in polynomial time if additional path constraints are imposed [57]. Ordóñez *et al.* provided conditions for demand and cost uncertainty sets to make the network optimization problem tractable [61]. In contrast to those polynomial time-solvable problems, Atamtürk *et al.* showed that a two-stage robust optimization for a multicommodity network flow and design problem with discrete design variables under demand uncertainty is NP-hard [5]. Chekuri *et al.* proved coNP-hardness of the single-source robust network design problem under demand uncertainty [17], which we state precisely in Section 4.4.

Many of the robust optimization problems in the literature of networking have considered uncertainty about demand or link cost. In contrast, we focus on uncertainty about link status—link loss rates (or, interchangeably, link success probabilities)—in this chapter. The contribution of this research is as follows: we prove the coNP-hardness of the problem for objectives of minimizing cost and maximizing throughput; we also provide a polynomial-time solvable problem formulation by introducing path constraints that approximate the problem.

4.2 Background on Robust Optimization

Robust optimization is a field of optimization theory that deals with uncertainty in variables and constraints. Robust optimization has been studied in parallel with operation research, control theory, and economics since the establishment of modern control theory in the 1950s. In an optimization problem with unknown variables, the most developed approaches are worst-case analysis, stochastic optimization, and deterministic optimization with sensitivity analysis.

The worst case analysis dates back to Wald's maximin model [76–78]. This is a non-probabilistic model for which the best decision provides a worst outcome at least as good

as any other decisions' worst outcome.

$$\begin{aligned} &\max_{x\in X} \min_{u\in U(x)} f(x,u) \\ \Leftrightarrow &\max_{x\in X, v\in\mathbb{R}} \{v : v \le f(x,u),\ \forall u \in U(x)\} \end{aligned} \tag{4.1}$$

where x is a decision variable in the decision space X, and $U(x)$ is the set of possible values of u associated with x. Equation (4.1) provides the best worst-case solution over all possible values of u. However, the worst case analysis sometimes results in more conservative solutions.

Stochastic optimization relies on the probability distribution functions of the uncertain variables and quantifies the realization of the probabilistic variables [74]. Therefore, the scenario-based stochastic optimization approach cannot be used unless the uncertainty is probabilistic and the probability distribution functions are known.

The last approach is to simply ignore the uncertainty. The uncertain variables are replaced by some nominal values and the optimization problem is solved. Then, sensitivity analysis is used to measure the solution's vulnerability to uncertainty. This method is meaningful if the perturbation of the uncertain variables is within the stability conditions.

In this chapter, we focus on robust convex optimization (e.g., [7, 26]). First, we start with a single-stage robust optimization[1] (called the Robust Counterpart (RC) problem). We follow the notation introduced in [7]. Consider the following optimization problem:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x,u) \\ &\text{subject to } F(x,u) \in \mathbf{K} \subseteq \mathbb{R}^l \end{aligned} \tag{4.2}$$

where $x \in \mathbb{R}^n$ is a decision vector and $u \in \mathbb{R}^m$ is the data element of the problem. The dimensions of vectors x and u, the mappings $f(\cdot,\cdot)$, $F(\cdot,\cdot)$, and the convex cone $\mathbf{K}$ are given structures of the problem. In (4.2), the uncertainty set $\mathcal{U} \subset \mathbb{R}^m$ is given, but we do not have perfect knowledge of u. The choice of a variable x is made before the actual realization of the uncertain data u becomes known. The constraint $F(x,u) \in \mathbf{K}$ must be satisfied for all

[1] The origin of the name comes from all decision variables being determined in one stage before the actual realization of the uncertain data becomes known. It becomes clearer when it is compared to the two-stage optimization in the following paragraph.

realizations of u (i.e., $F(x,u) \in \mathbf{K},\ \forall u \in \mathcal{U}$). To ensure the best possible guaranteed value, we have the following optimization problem, which is the same as (4.1):

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \sup_{u\in\mathcal{U}} f(x,u) \\ &\text{subject to } F(x,u) \in \mathbf{K} \subseteq \mathbb{R}^l,\ \forall u \in \mathcal{U} \end{aligned} \tag{4.3}$$

The solution of (4.3) is called a robust optimal solution to the uncertain optimization problem (4.2). In addition, (4.3) is called a robust counterpart of (4.2). Note that (4.3) is a semi-infinite optimization problem, and hence cannot be efficiently solved in general. Therefore, many research results have been proposed to convert the robust counterpart problem to a convex optimization problem for which well-known optimization techniques can be applied such as interior point methods [44, 60] and other convex optimization techniques [8].

In some particular scenarios of a robust optimization, the optimization variables can be partitioned into two sets: those that are determined before the realization of uncertain parameters and those that are determined after the realization. The first part is called non-adjustable variables and the latter part is called adjustable variables. This type of uncertain optimization problem is a two-stage optimization problem (called the Adjustable Robust Counterpart (ARC) problem [6, 61]). For the single-stage optimization, decision variables become feasible only if they satisfy the constraints for all realizations of the uncertain variables. However, for the two-stage optimization, the treatment of adjustable variables provides more flexible solutions. Accordingly, since ARC has a larger robust feasible set, ARC is less conservative than the usual RC and therefore the solution for ARC is at least as good as the one for RC. However, a two-stage optimization increases the problem complexity significantly. For example, while the RC of an uncertain linear program is computationally tractable, this is not the case for ARC. It is known that the ARC problem of a linear program with polyhedral uncertainty set is NP-hard in general [38].

To solve the ARC tractably, the Affinely Adjustable Robust Counterpart (AARC) is proposed in [6]. In AARC, the adjustable variables are assumed to be affine functions of the uncertain data. AARC provides a tractable convex optimization problem solution (e.g., linear program and semi-definite program) in some important scenarios, and a tight approximate solution for the other cases. In this chapter, we take advantage of robust convex optimization techniques such as ARC and AARC.

4.3 Network Model and Problem Formulation

We consider a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of links (arcs). $m = |\mathcal{N}|$ is the number of nodes and $n = |\mathcal{E}|$ is the number of links. To simplify the problem, we consider a network coding problem with a single source node and a single sink node, where network coding provides a robustness benefit.

For performance metrics, we have two different objective functions: minimizing cost of subgraph construction and maximizing throughput for a given cost constraint. For the former problem, we want to find a min-cost network coding subgraph that satisfies a demand requirement for the sink node. For the latter problem, we want to find a max-throughput network coding subgraph that satisfies a cost constraint on the chosen subgraph.

We denote by $\mathcal{P}_{ij}$ the set of simple paths from node i to node j ($\mathcal{P} := \bigcup_{ij} \mathcal{P}_{ij}$). We use **c** and **cap** for a given path cost vector and link capacity vector, respectively. Variable bgt is the budget for subgraph construction when the objective function is the throughput of network, and D is the demand required for the sink node when the objective function is the cost of subgraph construction. We introduce a budget, bgt, and a demand, D, to avoid trivial solutions such as flooding and zero flow, respectively.

We have the following uncertainty and decision variables:

- Uncertainty: **w** (vector of path success probabilities) taking values in an uncertainty set W
- Optimization variables:
  - **k** (network coding subgraph; k_p is the maximum feasible flow on path $p \in \mathcal{P}$) determined prior to the realization of the path success probabilities.
  - **h** (vector of actual path flows; h_p is the flow along the subgraph, k_p, in the presence of path loss rates, i.e., $h_p \leq k_p w_p, \ \forall p \in \mathcal{P}$) determined after the realization of the path success probabilities.

The result is a two-stage optimization problem called the ARC problem [6, 61]. This formulation fully captures the robustness properties of network coding. Since the actual flow is determined after the realization of path success rates, network coding intrinsically exploits the best routes for the actual flow within the capacity constraints from the predetermined subgraph. In contrast, a single-stage robust optimization called the RC problem [7] would

require fixing a flow $\mathbf{h}$ feasible under any realization of the link qualities, resulting in a much more conservative solution. Since ARC has a larger robust feasible set, the solution for ARC is at least as good as the one for RC. However, a two-stage optimization increases the problem complexity significantly. It is known that the ARC problem of a linear program with polyhedral uncertainty set is NP-hard in general [38]. We investigate the tractability of this problem in the next section.

4.4 Hardness of Robust Subgraph Selection

4.4.1 Min-Cost Criterion

The robust minimum cost subgraph selection problem with the demand requirement in terms of path flows is as follows:

$$\begin{aligned} \min_{\mathbf{k}} \quad & \sum_{p\in\mathcal{P}} c_p k_p \\ s.t. \quad & \sum_{p\in\mathcal{P}:l\in p} k_p \le cap(l),\ \forall l \in \mathcal{E} \\ & \text{for all } \mathbf{w}\in W,\ \exists \mathbf{h}: \quad \begin{cases} h_p \le k_p w_p, \forall p \in \mathcal{P} \\ \sum_{p\in\mathcal{P}} h_p \ge D \end{cases} \end{aligned} \tag{4.4}$$

Note that any network flow problem can be formulated using path and cycle flows and vice versa [2, Theorem 3.5]. Thus, we can find the corresponding link formulation as follows. If d is a scalar for the demand, then we can write a supply-demand vector as $d(\mathbf{e}_s - \mathbf{e}_t)$, where $\mathbf{e}_s, \mathbf{e}_t$ are canonical vectors for source and sink, respectively. Let $\mathbf{N}$ denote a node-arc incidence matrix [2]. Note that all variables with tildes are arc-flow variables and $\tilde{W}$ is the uncertainty set for the link success probabilities. Then, the corresponding link formulation is as follows:

$$\begin{aligned} \min_{\tilde{\mathbf{k}}} \quad & \sum_{l\in\mathcal{E}} \tilde{c}(l)\tilde{k}(l) \\ s.t. \quad & 0 \le \tilde{k}(l) \le cap(l), \forall l \in \mathcal{E} \\ & \text{for all } \tilde{\mathbf{w}}\in \tilde{W},\ \exists \tilde{\mathbf{h}}, d: \quad \begin{cases} \tilde{h}(l) \le \tilde{k}(l)\tilde{w}(l), \forall l \in \mathcal{E} \\ \mathbf{N}\cdot\tilde{\mathbf{h}} = d(\mathbf{e}_s - \mathbf{e}_t) \\ d \ge D \end{cases} \end{aligned} \tag{4.5}$$

We consider hardness of (4.4) and (4.5) under polyhedral uncertainty sets for path success probabilities W and link success probabilities $\tilde{W}$, respectively. We show the complexity of this problem by reduction from the single-source robust network design problem under demand uncertainty, which is known to be coNP-hard for undirected and directed graphs [17]. An instance of the latter is defined by a given graph $G = (\mathcal{N}, \mathcal{E})$, a link cost vector $\mathbf{c}$, a single source node $s \in \mathcal{N}$ and a convex polyhedral set $\mathcal{D}$ of demand matrices such that for each $D \in \mathcal{D}$, the demanded flow D_{ij} from node i to node j is zero for non-source nodes $i \neq s$. The objective is to find the least cost vector of link capacity reservations $\mathbf{u}$ sufficient to support a multi-commodity fractional routing for each demand matrix in $\mathcal{D}$, i.e.,

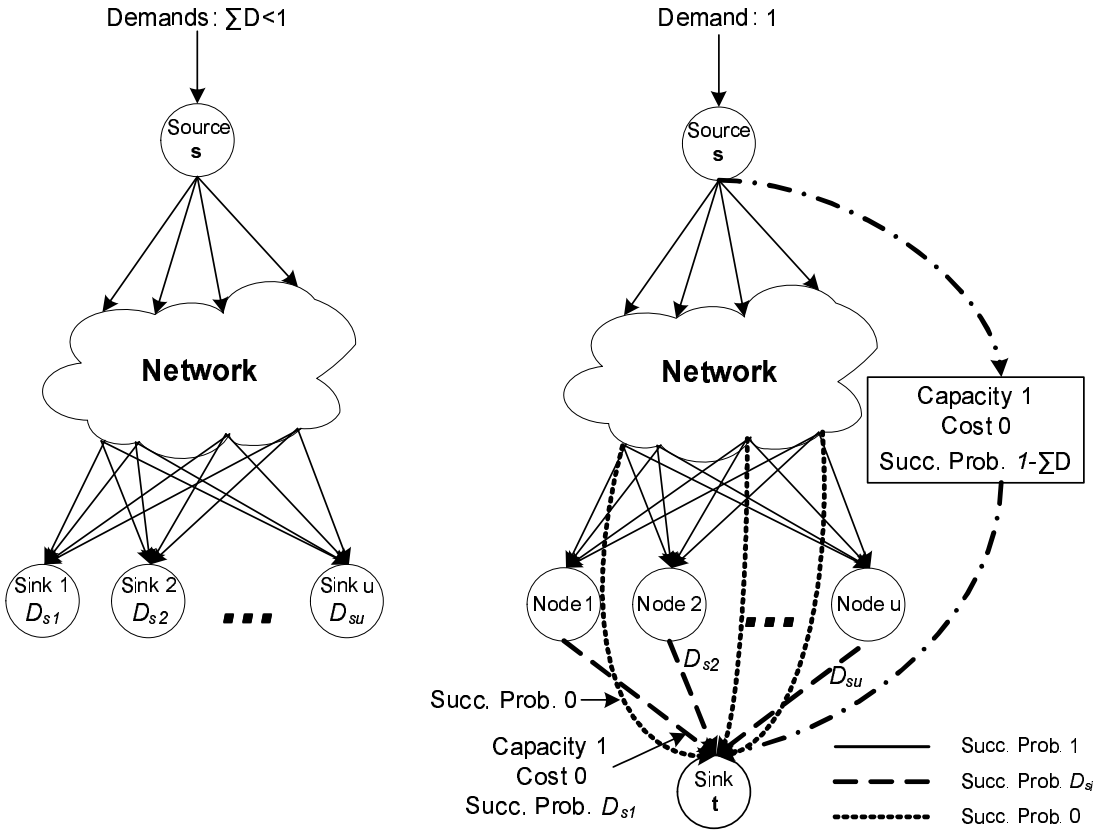
$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{l} c(l)u(l) \\ s.t. \quad & \text{for all } D \in \mathcal{D},\ \exists \mathbf{f} \text{ satisfying} \\ & \begin{cases} \sum_{p \in \mathcal{P}_{ij}} f_p = D_{ij},\ \forall\, i, j \in \mathcal{N} \\ \sum_{ij} \sum_{p \in \mathcal{P}_{ij} : l \in p} f_p \leq u(l),\ \forall\, l \in \mathcal{E} \end{cases} \end{aligned} \tag{4.6}$$

where $\mathbf{f}$ is a vector specifying the flow f_p on each path p. Note that the routing may change for different demand matrices—thus, it is not a path-constrained problem. We will show that an instance of our problem (4.5) is equivalent to Fig. 4.1 (a).

Theorem 4.1. *The robust minimum cost subgraph selection problems (4.4) and (4.5) are coNP-hard for polyhedral uncertainty sets W and $\tilde{W}$, respectively.*

Proof. Note that for the robust network design problem (4.6), if each demand matrix in $\mathcal{D}$ is scaled by a constant factor, then the optimal link capacity reservation vector $\mathbf{u}$ is scaled by the same factor. Thus without loss of generality, we consider an instance H of problem (4.6) with graph $G = (\mathcal{N}, \mathcal{E})$, source s and convex polyhedral set $\mathcal{D}$ of demand matrices such that $\sum_{ij} D_{ij} < 1\ \forall\, D \in \mathcal{D}$, illustrated in Fig. 4.1 (a).

From this we obtain an instance H' of the robust minimum cost subgraph selection problem (4.5), illustrated in Fig. 4.1 (b), as follows. We add to network G an additional node t; s and t are the source and sink nodes, respectively. We introduce an additional link l_i of capacity 1, cost 0 and success probability w_{it} from each node $i \in \mathcal{N}$ to t. The vector



(a) Uncertain Demands (b) Uncertain Link Loss Rates

Figure 4.1: Two equivalent network optimization problems with different uncertainties: (a) Minimizing cost under demand uncertainty (b) Minimizing cost under uncertainty of link success probabilities

$\mathbf{w}$ of success probabilities w_{it} lies in the uncertainty set

$$\begin{aligned} w_{st} &= 1 - \textstyle\sum_{i,j} D_{ij} \\ w_{it} &= D_{si} \;\; \forall\, i \neq s \\ D &\in \mathcal{D} \end{aligned} \tag{4.7}$$

which is a convex polyhedral set. The links in $\mathcal{E}$ have success probability 1 and no capacity constraints. The demanded $s-t$ flow is 1. This requires the solution to have $\tilde{k}(l_i) = 1$ and, for each $D \in \mathcal{D}$, there must exist a multicommodity flow $\tilde{\mathbf{h}}$ of size D_{si} from s to each other node $i \in \mathcal{N}$ satisfying $\tilde{h}(l) \leq \tilde{k}(l)$. Thus, an optimal solution for the robust minimum cost subgraph selection problem also solves the single-source robust network design problem. Note that since no $s-t$ path contains more than one of the uncertain links, (4.7) also corresponds to a polyhedral uncertainty set in terms of path success probabilities: all paths through link l_i $(i \neq s,\ i \in \mathcal{N})$ have path success probability D_{si}, and the path through the link (s,t) has path success probability $1 - \sum_{i,j} D_{ij}$. Then by using the same proof, the path formulation (4.4) is also coNP-hard. This completes the proof. □

4.4.2 Max-Throughput Criterion

The path formulation for the robust maximum throughput subgraph selection problem with the cost constraint is as follows:

$$\begin{aligned} \max_{\mathbf{k},\mathbf{h}} \quad & \sum_{p \in \mathcal{P}} h_p \\ s.t. \quad & \sum_{p \in \mathcal{P}: l \in p} k_p \leq cap(l),\ \forall l \in \mathcal{E} \\ & \mathbf{c}^T \cdot \mathbf{k} \leq bgt \\ & \text{for all } \mathbf{w} \in W,\ \exists \mathbf{h}: \quad h_p \leq k_p w_p, \forall p \in \mathcal{P} \end{aligned} \tag{4.8}$$

Much like the min-cost problem, we can formulate the corresponding link formulation

as follows:

$$\begin{aligned} \max_{\tilde{\mathbf{k}}} \quad & d \\ s.t. \quad & 0 \le \tilde{k}(l) \le cap(l), \forall l \in \mathcal{E} \\ & \sum_l \tilde{c}(l)\tilde{k}(l) \le bgt \\ & \text{for all } \tilde{\mathbf{w}} \in \tilde{W},\ \exists \mathbf{h}, d: \quad \begin{cases} \tilde{h}(l) \le \tilde{k}(l)\tilde{w}(l), \forall l \in \mathcal{E} \\ \mathbf{N} \cdot \tilde{\mathbf{h}} = d(\mathbf{e}_s - \mathbf{e}_t) \end{cases} \end{aligned} \tag{4.9}$$

The link formulation corresponding to (4.9) can be found similarly. Now, we prove coNP-hardness of the max-flow subgraph selection problem.

Theorem 4.2. *The robust maximum flow subgraph selection problems (4.8) and (4.9) are coNP-hard for polyhedral uncertainty sets W and $\tilde{W}$, respectively.*

Proof. We consider an instance H of problem (4.6) with graph $G = (\mathcal{N}, \mathcal{E})$, source s and convex polyhedral set $\mathcal{D}$ of demand matrices such that $\sum_{ij} D_{ij} < 1\ \forall\ D \in \mathcal{D}$, and construct the instance H' of problem (4.5) exactly as in the proof of Theorem 4.1. Let C be the optimal cost of H.

From H' we construct an instance of problem (4.9) as follows. We add another link l' from s to t that has cost $c' > n \max_{l \in \mathcal{E}} c(l)$, success probability 1 and no capacity constraint, and set $bgt > C$. The optimal solution has $k(l') = (bgt - C)/c'$, and contains a solution for H. Similarly to the min-cost problem, the max-flow problem (4.8) with polyhedral uncertainty for path success probabilities can also be shown to be coNP-hard. This completes the proof. □

4.5 Path Formulation

Now we consider the introduction of path constraints to the network optimization problem. Under these constraints the problem becomes polynomial time-solvable. In this section, we describe the approximate solution for the max-throughput problem only. The approximate solution for the min-cost problem is analogous.

We introduce a path constraint, $h_p = k_p w_p$, to replace the last equation in (4.8): the actual flow is an affine function of the uncertainty (called the AARC [6]). Thus, it becomes

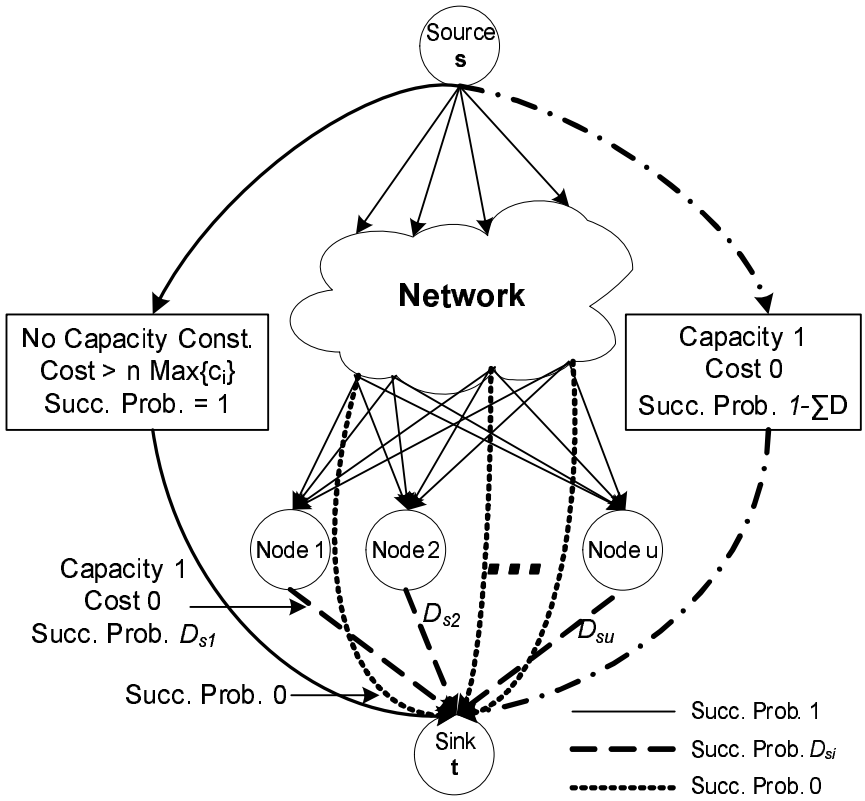


Figure 4.2: One instance of robust maximum flow subgraph construction problem. All graph parameters are same as in Fig. 4.1 (b), except for an additional link (with no link uncertainty) between the source and the sink node.

an approximate problem. Now we can replace $\sum_{p\in\mathcal{P}} h_p$ with $\sum_{p\in\mathcal{P}} k_p w_p$ and formulate a $\max_k - \min_w$ problem as follows:

$$\begin{aligned} \max_{\mathbf{k}} \min_{\mathbf{w}} \quad & \sum_{p\in\mathcal{P}} k_p w_p \\ s.t. \quad & \sum_{p\in\mathcal{P}:l\in p} k_p \le cap(l),\ \forall l \in \mathcal{E} \\ & \mathbf{c}^T \cdot \mathbf{k} \le bgt \\ & \mathbf{w} \in W \end{aligned} \tag{4.10}$$

By using duality of the minimization problem, we can combine the minimization problem with the maximization problem. If W is a convex polyhedron set, then the combined problem becomes a single linear program (LP). Let z_{ROS} and $\mathbf{k}_{ROS}$ denote the optimum objective value (Max-throughput) and the optimum subgraph, respectively. Then

$$\begin{aligned} z_{ROS} = \max_{\mathbf{k},r,\lambda,\mu} \quad & r \\ s.t. \quad & \sum_{p\in\mathcal{P}:l\in p} k_p \le cap(l),\ \forall l \in \mathcal{E} \\ & \mathbf{c}^T \cdot \mathbf{k} \le bgt \\ & -\lambda^T \cdot \mathbf{g} - \mu^T \cdot \mathbf{g}_{eq} \ge r \\ & \mathbf{k} + \mathbf{H}^T \cdot \lambda + \mathbf{H}_{eq}^T \cdot \mu \ge 0 \\ & \mathbf{k}, \lambda \succeq 0, \quad r \ge 0 \end{aligned} \tag{4.11}$$

Therefore, for a case with tractable problem size where we have a polyhedral uncertainty set for path success probabilities, we can solve this problem by using any efficient LP solving algorithm such as Interior-Point method [44]. Although we have a single LP formulation, however, the problem size, particularly the number of paths, grows exponentially in the size of the network in general. Therefore, for a large network, LP formulation (4.11) can be intractable due to the enormous problem size. In such a case, we speculate that we may use the column generation approach [2], but we have not proved its application.

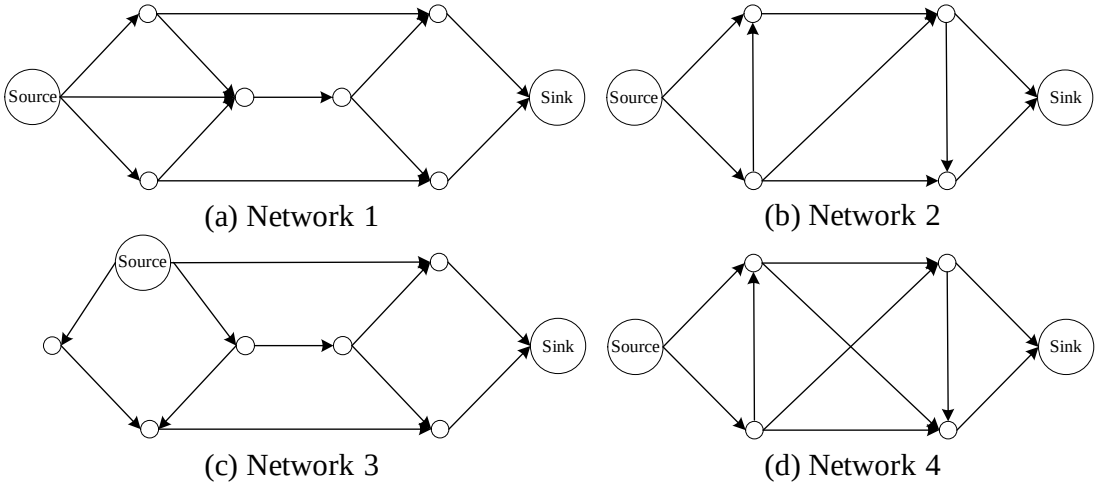


Figure 4.3: Example networks

4.6 Evaluations for Path Formulation

In this section, we evaluate the performance of the path formulation in four simple network examples shown in Fig. 4.3. We compare the two-stage robust optimization with the non-robust optimization in terms of the worst-case performance. For simplicity, we consider a box uncertainty set with a linear equality constraint:

$$W = \{\mathbf{w} : a_i \leq w_i \leq b_i,\ \forall i \in \mathcal{P},\ \mathbf{H}_{eq} \cdot \mathbf{w} = \mathbf{g}_{eq}\}$$

Each path has a nominal path success probability, $\overline{w}_i = (a_i + b_i)/2$, with variations around the nominal value; in addition, the equality constraint, $\mathbf{H}_{eq} \cdot \mathbf{w} = \mathbf{g}_{eq}$, satisfied by nominal values as well, is introduced to avoid a trivial solution. Furthermore, the equality constraint normalizes the max-flow over the uncertain scenarios.

4.6.1 Deterministic Case with Nominal Values

First, we consider the deterministic case with nominal values for uncertain path success probabilities, $\overline{\mathbf{w}}$. Let z_{DTM} and $\mathbf{k}_{DTM}$ denote the optimum objective value (max-throughput) and the optimum subgraph, $\mathbf{k}$, respectively.

$$\begin{aligned} z_{DTM} = \max_{\mathbf{k}} \quad & \sum_{p \in \mathcal{P}} k_p \overline{w}_p \\ s.t. \quad & \sum_{p \in \mathcal{P}: l \in p} k_p \leq cap(l),\ \forall l \in \mathcal{E} \\ & \mathbf{c}^T \cdot \mathbf{k} \leq I \end{aligned} \tag{4.12}$$

Note that this method corresponds to a single-stage non-robust optimization strategy.

4.6.2 Maximum Flow with Given Subgraph

To make the problem (4.8) tractable, we assumed that the second stage variables, $\mathbf{h}$, are affine functions of the uncertainty, (i.e., $h_p = w_p h_p$), but in reality, the second stage variables can change arbitrarily. Therefore, we can evaluate the max-flow for a given subgraph with arbitrarily chosen second stage variables. To evaluate the approximate solution, we compare the worst-case performances (max-flow) for given subgraphs. We can formulate an optimization problem for the worst-case performance with a new capacity bound, $u_l(S, \mathbf{w})$, the usage of link l of a subgraph S as follows:

$$\begin{aligned} z_{WC}(S) = \min_{\mathbf{w}\in W} \max_{\mathbf{h}} \quad & \sum_{p\in\mathcal{P}} h_p \\ s.t. \quad & \sum_{p\in\mathcal{P}:e\in p} h_p \le u_l(S,\mathbf{w}),\ \forall l \in \mathcal{E} \end{aligned} \tag{4.13}$$

where $u_l(S, \mathbf{w})$ can be found as follows:

$$\begin{aligned} u_l(\mathbf{k}_{ROS}, \mathbf{w}) &\triangleq \textstyle\sum_{p\in\mathcal{P}:l\in p} k_{ROS_p} w_p \quad \text{from (4.11)} \\ u_l(\mathbf{k}_{DTM}, \mathbf{w}) &\triangleq \textstyle\sum_{p\in\mathcal{P}:l\in p} k_{DTM_p} w_p \quad \text{from (4.12)} \end{aligned}$$

However, when we reduce (4.13) into one single optimization problem, the inequality constraint results in minimizing a non-convex quadratic function, which is NP-hard in general. Therefore, instead, we generate 5,000 random samples for path success probabilities uniformly from a given uncertainty set W and compare the minimum values among them. Using this random simulation, we compare the robust optimization solution and the deterministic solution in terms of the worst-case performance in the network examples, shown in Fig. 4.3.

4.6.3 Results

From the simulation, we observe that the robust optimization solution results in better performance than the non-robust optimization in terms of worst-case performance over 90% of times. The performance gain of the robust optimization solution against the non-robust strategy is around 10% for all examples. In addition, the robust optimization solution

Table 4.1: Frequency when $\mathbf{k}_{ROS}$ provides better worst-case performance than $\mathbf{k}_{DTM}$ (Total 200 random trials)

Network	$z_{WC}(\mathbf{k}_{ROS}) \geq z_{WC}(\mathbf{k}_{DTM})$	Avr. Perf. Gain
1	95%	12.3%
2	97%	8.7%
3	97.5%	38.3%
4	98%	11.8%

Table 4.2: Comparison of Problems Tractability

Uncertainty	Routing (Fixed Paths)	Network Coding
Demands	P [4,57]	coNP-hard [17]
Links	**P**	**coNP-hard**

provides more stable outputs whereas the deterministic solution sometimes results in poor performance.

Recall that the path constraint that is introduced to make the problem tractable results in an approximation of the original problem. So, the worst-case max-flow for a robust optimization solution without path constraints is always at least as good as the solution with path constraints, i.e., $z_{WC}(\mathbf{k}_{ROS}) \geq z_{ROS}$ always holds.

The simulation results are summarized in Table 4.1 and Fig. 4.4.

4.7 Conclusions

We have described the problem of network coding subgraph construction in networks where there is uncertainty about link success probabilities. We formulated the problem using the best worst-case guaranteed robust optimization technique. However, we proved that the problem is coNP-hard for the min-cost objective and the max-throughput objective. Accordingly, we suggested a tractable approximate solution using path constraints. The tractability of network optimization problems with different problem formulations and different uncertainties is summarized in Table 4.2.

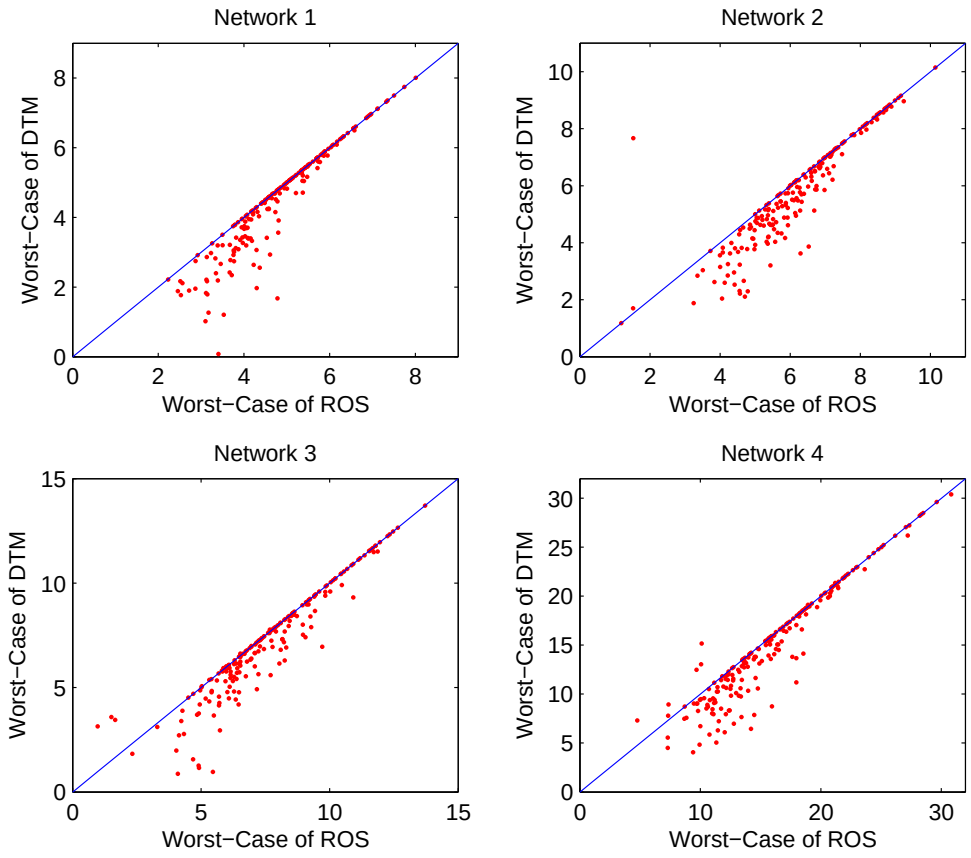


Figure 4.4: Worst-case of robust optimization solution (ROS) versus worst-case of deterministic solution (DTM). For all network examples, more dots are located in the lower triangle, meaning $z_{WC}(\mathbf{k}_{ROS}) \geq z_{WC}(\mathbf{k}_{DTM})$.

Chapter 5

Coding for Prioritized Communication

In this chapter, we study prioritized communication on a simple network topology that consists of the source and destination pair connected by multiple parallel links, each of which is subject to uncertain availability. We provide an optimal solution to maximize a payoff that assigns weights based on the worth of data and the probability of successful transmission. Ideally, the choice of what information to send over the various links will provide protection of high value data when very few links are available, yet result in communication of significant additional data when most links are available. Our approach applies a linear programming formulation to find the optimal timesharing strategy among a given set of simple inter-link codes.

5.1 Introduction

When communicating through a noisy one-way communication link, it is well known that it is often practical to achieve a high level of reliability by protecting the communicated data with error-correcting codes, provided the attempted data rate is not too high. However, suppose that with some probability the communications link may fail. Clearly, error-correcting codes are of no use in protecting against this type of link failure. When there are multiple unreliable links, the communicated data can be protected, at least to some extent, with error-correcting codes applied between the links. For example, if there are three links and identical data is sent on all links (a simple repetition code), then the data is protected

The work in this chapter was presented at the 44th Annual Conference on Information Sciences and Systems (CISS) [15].

against failure of any two links.

More generally, we want to send multiple messages of varying worth (or priorities) from one point to another in a network that contains unreliable links. Ideally, we would want to achieve the maximum available throughput at all times, in a way that: (1) protects higher-value data, and (2) does not require prior knowledge of the network state. Roughly speaking, we would like to provide protection of higher-value data when a large fraction of the links in a network are unavailable, and to achieve transmission of significant additional data when most links are working properly.

For simplicity, we restrict our attention to one simple type of network: a single source node and a single destination node connected by parallel unreliable links. Each link is either available for the entire duration of the communication attempt or totally unavailable during the communication. Specifically,

- a given link fails with some known probability; otherwise the link provides reliable communications with a known capacity;
- link failures are independent;
- the sender does not know the status of the links;
- there are messages to be sent, and the messages have known worths (i.e., priorities or values) and sizes;
- the worth of a partial message is proportional to its size (i.e., partial credit is given for partial messages); and
- the payoff of a given link usage strategy is the expected total value of the messages successfully decoded.

This network topology is used as a simplified model for studying scenarios like the one shown in Fig. 5.1. Consider a rover on Mars. In a given period of time, the rover can communicate to Earth in three ways: a direct-to-Earth link and two different relays through spacecraft orbiting Mars. It is assumed that these links have independent and non-negligible outage probabilities. It is also assumed that the resources (e.g., time and power) needed to utilize these links are small, so that the rover should always attempt to send information through all three routes. Given a list of messages and the values, we would like to find the

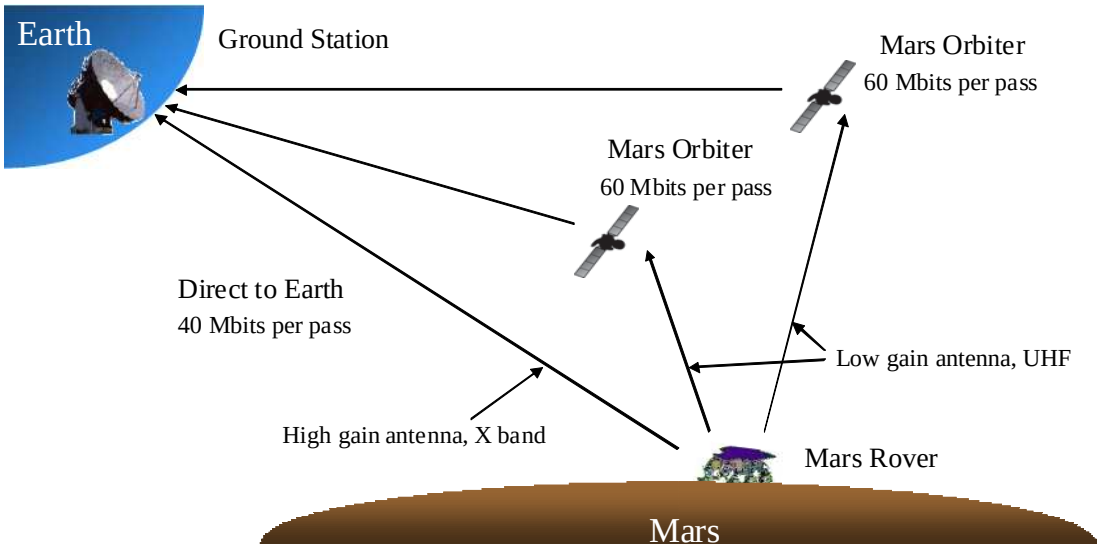


Figure 5.1: Possible application: a rover on Mars attempts to communicate to Earth by three independent paths.

best combination of messages to send over the three links. Note that the separate links do not have to be used simultaneously to fit our model. In fact, it is possible to use our model to represent a single link used at different times, provided that outage events are reasonably modeled as independent random variables.

We will present the model as though a given link capacity indicates the maximum amount of data that can be transmitted through the link (if it is up) during the communication attempt, and message sizes are also amounts of data on the same scale. Thus in a sense the communication attempt is an one-shot action. However, it would be equally valid to regard both capacities and message sizes as representing data rates in bits per unit time. From this viewpoint the model would pertain to communication for an indefinite period of time, during which it is not possible to inform the sender which links are working.

Communication strategies for our model involve timesharing among relatively simple inter-link codes. Given a collection of candidate inter-link codes, the optimal timesharing proportions can be determined using linear programming. However, it appears to be difficult to determine the set of all "useful" inter-link codes for a given number of messages and links. We discuss this problem and describe an algorithm that can assist in determining all potentially useful inter-link codes that use a subset of the links and use these links equally, for a given number of messages and links.

In most communications scenarios previously studied, one is either interested in achiev-

ing arbitrarily small loss or error rates (in theoretical studies), or merely very small loss or error rates (in practical systems). The communication model described here is conceptually different in that it need not deal with small loss rates. Under our model, we accept that losses will occur and we measure performance in terms of the messages successfully communicated. This model may be appropriate for, say, a spacecraft that is capable of gathering much more information than it can transmit to Earth.

Our simple network topology is a special case of a network coding problem. Most previously considered network coding problems fall into one of two categories: either they seek to maximize (a constant) throughput in a fixed network, or they seek robust communication at a constant rate in an unreliable network. Our model differs from both of these in that the throughput will depend on the network state even though a fixed coding scheme is used.

Indeed, even though we only need codes for correcting erasures and are primarily concerned with short block lengths, our particular model appears to make investigation of useful codes challenging. Not only are we interested in recovery of partial information when the whole codeword cannot be recovered, but the different codeword positions correspond to different links, and thus can have different erasure probabilities.

5.1.1 Related Work

One somewhat related investigation of note is the Priority Encoding Transmission (PET) scheme of Albanese *et al.* [3]. Also of note is the extension of Silva and Kschischang [73] which shows how PET can be used in a network coding system. Like our model, the PET model includes a number of messages to be transmitted, and each message has an associated priority. Transmitted packets may be lost, but a given message can be recovered if a sufficient fraction of packets arrive successfully, where the fraction depends on the message's priority. One key difference between our model and the PET model is that under the PET model, there can be a large number of packets that may be received or lost independently, while under our model all packets sent on a given link are either lost or successfully received together, and there are a small number of different links. Our model includes outage probabilities, while under PET the only concern is the fraction of packets successfully received. Under PET all information is protected with (possibly trivial) maximum distance separable (MDS) codes. Under our model MDS codes may be used, but the natural way to use them would be to put different symbols on different links. When

used this way the blocks lengths would never need to be larger than the number of links. Furthermore, we are concerned with partial decoding of codes, and this turns out to imply that non-MDS codes can be useful.

If the link capacities are all equal, then under our model one could regard the union of the parallel links as a single channel with an unknown state. One symbol for this channel would be a vector consisting of one symbol for each link. (This idea could also be extended to accommodate differing link capacities.) This channel now fits the assumptions of a compound channel [21], but since all links might be down, the usual compound channel capacity (the maximum guaranteed throughput) would be zero.

The model of a channel with an unknown state is also related to the broadcast channel model [20], where each channel state corresponds to a separate "receiver". The broadcast channel model is closely related to the unequal error protection model (see, e.g., [10]), which includes the concept of differing data worths. As a broadcast channel, with N parallel links our model would have $2^N - 1$ receivers (disregarding the state where all links are down). Thus even for small N it appears to be unwieldy to apply general broadcast channel results to our model.

5.1.2 Preliminaries

Let N be the number of parallel links. For $i \in \{1, \ldots, N\}$, link i has capacity c_i and outage probability p_i (equivalently, success probability $\bar{p}_i = 1 - p_i$). Let M be the number of messages. For $j \in \{1, \ldots, M\}$, message j has size s_j and worth per unit size π_j. The units of the link capacities and message sizes are arbitrary (but are the same for all links and messages).

Informally, for each link i, the sender sends a stream of data that is a function of the M messages and that is not larger than the link capacity c_i. The function can depend on all of the model parameters above. The receiver successfully receives the data on some subset of the links; this subset is known to the receiver. The receiver reconstructs the original messages to the extent possible from the data received. Let R_j be a random variable indicating how many size units of message j are recovered. The payoff from a communication attempt is the sum $\sum_{j=1}^{M} R_j \pi_j$. The payoff from a communication strategy is the expected value of this quantity.

It is implicit in this model that the message sizes are large and thus the messages

can be split into pieces closely approximating any given fraction. It is possible to more formally describe the space of communication strategies using codes on discrete alphabets and allowing limiting cases as the unit size becomes large; we stick with the above model for simplicity.

5.2 Optimization with Linear Programming

The assumptions of our model allow us to partition the messages into pieces and combine the pieces with inter-link codes. Essentially this amounts to timesharing among different codes. Given a list of candidate codes, we can determine how to optimally timeshare among them using linear programming.

Thus "simple" candidate codes form the building blocks of a communications strategy. For convenience in presenting examples, we introduce an informal concise notation for describing candidate codes. We describe this notation with examples. Consider the case of three links and two messages ($N = 3$ and $M = 2$). We label the messages A and B. One possible code consists of sending a portion of message A on all three links; we represent this code by (A, A, A). Similarly, a code could consist of sending a portion of message B on links 1 and 3; we represent this code by $(B, -, B)$. Another possible code consists of sending the same portion of message A on link 1, an equal-sized portion of message B on link 2, and the bitwise exclusive-or of these same portions on link 3; we represent this code by $(A, B, A + B)$, where the notation $A + B$ symbolizes addition in a finite field. A code can also use two different portions of the same message, as in $(A_1, A_2, A_1 + A_2)$. Codes such as (A, A, B) that can be obtained by timesharing simpler codes (here by equal parts of $(A, A, -)$ and $(-, -, B)$) do not need to be considered as candidate codes. Although this code notation is useful, we remark that it is not adequate to describe all possible codes.

We assume that the message sizes and message worths are all given. We also assume that the link capacities and outage probabilities are given. Suppose our list of codes contains n_c codes. Our objective is to find a column vector $\mathbf{z} = [z_1, \ldots, z_{n_c}]^T$ that describes how much we use each code.

The codes are described with an $N \times n_c$ matrix $\mathbf{K}$ that tells how much the codes use the links, an $M \times n_c$ matrix $\mathbf{L}$ that tells the message content of the codes, and an n_c element column vector $\mathbf{v}$ that gives the expected payoffs from the codes.

More specifically, in $\mathbf{K} = [\kappa_{i,k}]$ the entry $\kappa_{i,k}$ is the usage of link i by one unit of code k. In $\mathbf{L} = [\ell_{j,k}]$ the entry $\ell_{j,k}$ is the amount of message j sent with one unit of code k. In $\mathbf{v} = [v_1, \ldots, v_{n_c}]^T$, the entry v_k is the payoff (in expected received value) from one unit of code k; v_k is a function of the message worths and the link outage probabilities as well as of the properties of code k.

Observe that each code is described by an entry in $\mathbf{v}$ and a column in each of $\mathbf{K}$ and $\mathbf{L}$. As an example, consider the codes $(A, B, A+B)$ and (A_1, A_2, A_1+A_2) for the case of two messages and three links. We can describe $(A, B, A+B)$ by the columns $(1,1,1)$ and $(1,1)$ in $\mathbf{K}$ and $\mathbf{L}$, respectively. Note that the unit size is arbitrary here; scaling both columns by an arbitrary factor describes the same code. The entry in $\mathbf{v}$ would need to be scaled by the same factor. The code (A_1, A_2, A_1+A_2) can be described by $(1,1,1)$ and $(2,0)$.

With link capacity list $\mathbf{c} = (c_1, \ldots, c_N)^T$ and message size list $\mathbf{s} = (s_1, \ldots, s_M)^T$, the optimal code usage vector $\mathbf{z}$ can be determined via the following linear program:

$$\begin{aligned} \text{find } \mathbf{z} \text{ to maximize} \quad & \mathbf{v}^T\mathbf{z} \\ \text{subject to} \quad & \mathbf{K}\mathbf{z} \leq \mathbf{c} \\ & \mathbf{L}\mathbf{z} \leq \mathbf{s} \\ & \mathbf{z} \geq 0. \end{aligned} \tag{5.1}$$

As a concrete example, consider the case of 2 messages and 3 parallel links, with the following list of 17 codes:

$$\begin{aligned} \{&(A,-,-),\ (-,A,-),\ (-,-,A),\ (A,A,-),\ (A,-,A), \\ &(A,A,-),\ (A,A,A),\ (A_1,A_2,A_1+A_2),\ (B,-,-), \\ &(-,B,-),\ (-,-,B),\ (B,B,-),\ (B,-,B),\ (B,B,-), \\ &(B,B,B),\ (B_1,B_2,B_1+B_2),\ (A,B,A+B)\} \end{aligned}$$

Note that we have assumed the links are indexed so that $p_1 \leq \cdots \leq p_N$ (decreasing reliability) and that the messages are indexed in order of decreasing value (per unit size), $\pi_1 \geq \cdots \geq \pi_M$. With these assumptions, we do not need to consider any other permutations of symbols in the codes (A_1, A_2, A_1+A_2), (B_1, B_2, B_1+B_2), and $(A, B, A+B)$; for example, the code $(A, B, A+B)$ is always at least as good as $(B, A+B, A)$ by symmetry.

For this list of codes we have the following constants $\mathbf{K}$, $\mathbf{L}$, and $\mathbf{v}$:

$$\mathbf{K} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & \frac{1}{2} & 1 & 0 & 0 & 1 & 1 & 0 & 1 & \frac{1}{2} & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & \frac{1}{2} & 0 & 1 & 0 & 1 & 0 & 1 & 1 & \frac{1}{2} & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & \frac{1}{2} & 0 & 0 & 1 & 0 & 1 & 1 & 1 & \frac{1}{2} & 1 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{v}: \begin{cases} [v_1, v_9] &= [\pi_A, \pi_B]\ \bar{p}_1 \\ [v_2, v_{10}] &= [\pi_A, \pi_B]\ \bar{p}_2 \\ [v_3, v_{11}] &= [\pi_A, \pi_B]\ \bar{p}_3 \\ [v_4, v_{12}] &= [\pi_A, \pi_B]\,(\bar{p}_1 + \bar{p}_2 - \bar{p}_1\bar{p}_2) \\ [v_5, v_{13}] &= [\pi_A, \pi_B]\,(\bar{p}_1 + \bar{p}_3 - \bar{p}_1\bar{p}_3) \\ [v_6, v_{14}] &= [\pi_A, \pi_B]\,(\bar{p}_2 + \bar{p}_3 - \bar{p}_2\bar{p}_3) \\ [v_7, v_{15}] &= [\pi_A, \pi_B]\,(\bar{p}_1 + \bar{p}_2 + \bar{p}_3 - \bar{p}_1\bar{p}_2 - \bar{p}_1\bar{p}_3 - \bar{p}_2\bar{p}_3 + \bar{p}_1\bar{p}_2\bar{p}_3) \\ [v_8, v_{16}] &= [\pi_A, \pi_B]\,(0.5\bar{p}_1(1-\bar{p}_2)(1-\bar{p}_3) + 0.5\bar{p}_2(1-\bar{p}_1)(1-\bar{p}_3) \\ &\qquad + \bar{p}_1\bar{p}_2 + \bar{p}_1\bar{p}_3 + \bar{p}_2\bar{p}_3 - 2\bar{p}_1\bar{p}_2\bar{p}_3) \\ v_{17} &= \pi_A\bar{p}_1(1-\bar{p}_2)(1-\bar{p}_3) + \pi_B\bar{p}_2(1-\bar{p}_1)(1-\bar{p}_3) \\ &\qquad + (\pi_A + \pi_B)\,(\bar{p}_1\bar{p}_2 + \bar{p}_1\bar{p}_3 + \bar{p}_2\bar{p}_3 - 2\bar{p}_1\bar{p}_2\bar{p}_3) \end{cases}$$

For example, $v_4 = \pi_A(\bar{p}_1 + \bar{p}_2 - \bar{p}_1\bar{p}_2)$ because when one unit of the fourth code is used, one unit of message A can be decoded (with worth π_A) whenever at least one of the first two links are up (which happens with probability $\bar{p}_1 + \bar{p}_2 - \bar{p}_1\bar{p}_2$).

We solved the linear programming problem for a number of randomly generated scenarios (random message sizes and worths, and random link capacities and probabilities). We found that for each of the 17 codes, there were scenarios in which the code was needed in the optimal solution. The list of 17 codes can be verified to include all needed codes that use a subset of the links equally. For this relatively simple case, we think it is likely that no codes that use links non-uniformly are needed, i.e., we think the solutions obtained with these 17 codes would still be optimal if the candidate code list could be enlarged to include all possible codes.

The linear programming method clearly has limitations for our problem. For one thing, making the list of candidate codes seems to be a very complicated problem in general.

In addition, the number of candidate codes grows at least exponentially with the number of links, since there are already $2^N - 1$ possibilities just for repetition codes for the first message. However, the linear programming method appears to be viable for a small number of links.

5.3 Unit Size and Unit Capacity Problem

In this section, we consider a special case of the problem with unit message sizes and unit link capacities. The motivation for this is to try to characterize, or at least to be able to generate, complete sets of candidate codes for the general case. Here we consider only codes that use all links equally (not just a subset). We include codes that are formed by timesharing codes that use disjoint sets of links. We continue to allow arbitrary link outage probabilities and message worths.

Given a list of codes, we can generate a number of random outage probabilities and message worths, and in each case check which code in the list is best. In this way we can accumulate a smaller list of codes that can be best in some scenario, and we can attempt to characterize the properties of these codes. Generating the original list of codes is a formidable challenge, since even for small number of links there are an enormous number of possible codes. We could restrict our attention to linear codes, but the number of such codes is still huge. To improve tractability, we instead consider possible code properties instead of explicit codes. By code properties we mean a description of which messages can be decoded when any given erasure pattern occurs. To do this, we take advantage of a connection between entropy and matroids.

For our purposes, we need only the following characterization of matroids:

Corollary 5.1. *[63, Corollary 1.3.4] Let E be a set. A function $r : 2^E \to \mathbb{Z}^+ \cup \{0\}$ is the rank function of a matroid on E if and only if r satisfies the following conditions:*

(R1) If $X \subseteq E$, then $0 \leq r(X) \leq |X|$.

(R2) If $X \subseteq Y \subseteq E$, then $r(X) \leq r(Y)$.

(R3) If X and Y are subsets of E, then $r(X \cup Y) + r(X \cap Y) \leq r(X) + r(Y)$.

Now, we consider the connection between matroids and codes in the following subsection.

5.3.1 Codes and Matroids

Suppose we have a code C for M messages and N links, say $C = (X_1, \ldots, X_N)$ where each X_i is a function of $\mathbf{W} = (W_1, \ldots, W_M)$ and the W_j's are messages of equal size. We assume sizes are normalized so that the message sizes are all 1.

Suppose now that the messages, $W_1, \ldots, W_M$, are each random variables, independent and uniformly distributed on their possible values. Then $X_1, \ldots, X_N$ are also random variables since they are functions of $\mathbf{W}$. Let $U_1 = \{W_1, \ldots, W_M\}$, $U_2 = \{X_1, \ldots, X_N\}$, and $U = U_1 \cup U_2$. We consider the joint entropies of subsets of U. For convenience, we normalize the entropies so that any message W_j has unit entropy, i.e., $H(W_j) = 1$.

Now define $f : 2^U \to [0, \infty)$ by $f(S) = H(S)$, where 2^U is the power set of U and $H(S)$ is the joint entropy of the members of S (with $H(\varnothing) = 0$). The nonnegativity of conditional mutual information implies that this function is submodular, meaning $f(S_1 \cup S_2) + f(S_1 \cap S_2) \leq f(S_1) + f(S_2)$. Function f is also monotone, meaning $f(S_1) \leq f(S_2)$ whenever $S_1 \subseteq S_2$, and function f satisfies $f(\varnothing) = 0$. Therefore, f is a *polymatroid function* [25, 32, 52]. The fact that entropy is a polymatroid function is well known; see e.g., [25, 52].

For simplicity, we consider only codes for which f is integer-valued. This approach can be made more precise by limiting attention to a larger set of rational values that, when renormalized, maps to a set of integers.

To further simplify, we restrict attention to codes for which all X_i also have unit entropy (implying that the encoded symbols are "messages" of unit length). However, we think it is likely that there are codes that are "useful" and do not satisfy this condition.

With these further simplifications, the function f must be the rank function of a matroid (see [52, 63]), as it is integer-valued and satisfies $f(S) \leq |S|$.

Thus any code that satisfies our conditions has a corresponding matroid. However, the converse probably does not hold: it is likely that there are matroids that cannot be produced from a code as above. This is because it is known that there are matroids that are non-entropic, meaning there does not exist an ensemble of random variables with joint entropies corresponding to the matroids's rank function. In fact it is known that there are matroids that are not asymptotically entropic, which, loosely speaking, means they cannot be approximated closely by entropic polymatroids. See [52] for a more precise definition of

asymptotically entropic matroids and an example of a matroid that is not asymptotically entropic (the Vámos matroid).

Our interest in matroid rank functions for codes stems from two observations. First, the matroid rank function contains complete information about the performance of the code. Second, it appears to be much easier to systematically generate all matroid rank functions for a given number of messages and links than it is to generate all codes (or code properties).

We consider obtaining information about the performance of the code from the matroid rank function. First note that we require $f(S) = |S|$ when $S \subseteq U_1$, since the messages are independent. We also require $f(\{X_i\}) = 1$. The properties of matroid rank functions imply that if $S \subseteq U_2$ and $W_j \in U_1$, then either $f(S \cup \{W_j\}) = f(S)$ or $f(S \cup \{W_j\}) = f(S) + 1$. The latter implies W_j is independent of S, and so cannot be determined from S. The former implies W_j is completely determined from S, and so W_j can be recovered when the links corresponding to S are up.

We remark that our hypothesis that the messages are random and independent is only a tool for analysis; we do not require the messages to be random in an actual communications system. Clearly, if we want to be able to decode some W_j from a subset S of the code symbols, it must be necessary for this to be possible when the messages are random and independent. If it is possible in that case, then W_j must be deterministically obtainable from S no matter what the messages are.

As we mentioned earlier, some matroid rank functions may not correspond to any realizable code. However, if a code produces performance better or equal to that corresponding to any matroid rank function, we can still conclude the code must be optimal.

5.3.2 Automated Process: Generating Matroids and Calculating Performance Metrics

When we have M messages and N links, we consider a matroid with $(M + N)$ elements. By enumerating candidates for rank functions, we can count all possible matroids. Recall that the maximum rank is M and any singleton element has rank 1.

To efficiently generate all rank functions with given parameters, we use a backtracking algorithm [9]. Whenever we assign a possible value $(1, 2, \ldots, M)$ to an unassigned variable (rank function), we check the validity of that assignment by checking the conditions in Corollary 5.1.

Once we have a full list of all possible matroids, we can calculate the payoff for each matroid (each case) with an automated process. Recall that if $f(S) = f(S \cup \{W_j\})$, then W_j is decodable from the set of code positions (or links) corresponding to $S \subseteq U_2$. When we have N links, we need to check $2^N - 1$ combinations of links (excluding the empty set). For each combination, we check if the message W_j is decodable with that combination or not. For each combination, we have a corresponding probability that is weighted by the worth of the message and then added to the payoff.

5.3.3 Systematic Codes

If the performance of a given code is never better than the performance of another code, then the former code can be eliminated from consideration. As an example, consider the codes $(A, B, A+B, A+C)$ and $(A, B, A+B, C)$. In $(A, B, A+B, A+C)$, message C appears only once (link 4), and that is in combination with message A. Thus it is necessary but not sufficient for link 4 to be available to recover message C, and it can be verified that $A + C$ is never useful for recovering any other message. Therefore we cannot do any worse by replacing $A + C$ with C. More generally, by this reasoning we can eliminate from consideration any code for which a message is only involved in one code position, and the message is combined with one or more other messages there.

One might conjecture that this idea could be generalized further and we need only consider *systematic* codes, which for our purposes are codes in which any message involved in the code is sent directly (not combined with other messages) on some link. (For example, (A, A, A) and $(A, B, A + B)$ are systematic codes under this definition.) However, perhaps surprisingly, this conjecture appears to be false: for 5 links and 3 messages, our preliminary results indicate that there are combinations of outage probabilities and link failure probabilities for which some permutations of the codes $(A, B, A + B, A + C, B + \alpha C)$ or $(A, C, A + B, A + C, B + \alpha C)$ produce a higher payoff than any other code that does not divide up messages. (Here the codes are in a non-binary field and α is not 1. The two codes are different because we are assuming that A has higher worth than B which has higher worth than C.) These potential counterexamples were arrived at by enumerating what we believe to be all codes that could conceivably be optimal, then generating random scenarios (worths and outage probabilities) and checking which code gives the best payoff. This will be considered in more depth in Section 5.3.4.

When generating codes (or, more accurately, matroids representing codes) using the automated process described in Section 5.3.2, we can automatically determine if a given code is systematic with the following criterion:

(C1) For all message indices j such that $f(U_2) = f(U_2 \cup \{W_j\})$ (meaning that the message W_j is included in the code), there exists at least one link i such that $f(\{X_i, W_j\}) = 1$.

Note that this condition presumes that we have already required that $f(\{X_i\}) = 1$. If a code satisfies condition (C1), then the code is systematic.

5.3.4 Random Trial Results

We have run the described matroid search for 3 messages with 4 and 5 links. For the 4 links case, the search was completed. For this case we then randomly generated link outage probabilities and message worths, and in each case we determined which matroid achieved the best payoff. In a large number of trials, we did not encounter any case where the best matroid did not correspond to a non-systematic code. Furthermore, for every matroid that produced a maximum payoff, we could identify a corresponding code in a prior list of codes that we believed contained all codes that could be best.

For the 5 links case, generating the full list of matroids is computationally prohibitive. However we performed the same random trial search with a manually constructed list of codes in which we attempted to include every code that seemed likely to be best in some case. It was in these trials that we found that the previously mentioned non-systematic codes could give the best payoff.

5.4 Conclusions

We have presented a model for a communication scenario involving transmitting messages with different worths through an unreliable network. Our results concern a simple network that allows unicast communication over multiple parallel links.

We suggested a linear programming formulation that allows us to determine what combinations of messages to use across the multiple channels among a precomputed library of simple combinations of messages. We also described an algorithm for assisting in finding viable combinations of messages to populate this precomputed library.

Chapter 6

Summary and Future Work

6.1 Summary

This thesis investigates a number of issues in data communication networks with coding-based approaches.

We have studied theoretical limits on file download times in a peer-to-peer file distribution system. By applying a network coding optimization framework, our study overcomes optimization complexity issues and disproves a conjecture about optimal finish times without coding in previous related work. We also used this framework to study the effects of reciprocity, and show that coding provides performance gain in a dynamically changing network scenario.

Also in a peer-to-peer setting, we have investigated anonymous communication strategies using coding, in the presence of passive internal adversarial nodes that collude to determine the identities of communicating parties. We have analyzed the anonymity of the system in an information theoretic framework using an entropy measure which represents the amount of information the adversaries are short of to identify the source and sink pair. In the first part, we proposed an approach for anonymous subgraph construction which does not rely on public key infrastructure and has a formal information theoretic security characterization. We presented a practical heuristic approach for constructing a subgraph that works well across different adversarial models. A reverse path construction mechanism for feedback was also proposed. In the second part, we have studied the anonymous data transfer phase using network coding and proposed a coding-based technique against content correlation attacks. We have shown how the subgraph shape and parameters affect anonymity and networking performance.

Besides the peer-to-peer networking, we have investigated unreliable network problems. First, we have studied how to construct a network coding subgraph without knowing dynamically changing network parameters in advance. Instead of predicting or estimating the network parameters to be realized in the future, we suggested a robust subgraph construction algorithm that works relatively well across all scenarios. We showed that the problems of maximizing throughput and minimizing cost are computationally intractable (coNP-hard). We proposed a polynomial-time solvable strategy that optimizes over a subset of possibilities, and showed experimentally that it outperforms the conventional deterministic optimization strategy in most cases.

Lastly, we have investigated a prioritized communication over lossy links, for which the quality of service (QoS) matters. We considered a point-to-point communication scenario, where links have different capacities and link loss probabilities. Based on linear programming formulation and matroid theory, we presented strategies on how to determine message sizes and how to code them across the links. Our coding approach provided an insight on analysis and optimization of practical prioritized communication systems.

In summary, this thesis establishes a mathematical theory of practical system design in data communication networks. Our results have implications in both theory and practice. In theory, our investigation demonstrates scenarios where it is possible to mathematically analyze complicated communication systems using useful tools such as network coding, convex optimization, robust optimization, and matroid. Each analysis establishes a framework for optimizing the system performances. Technologically, our study considers practical issues that arise commonly in system design and implementation and suggests solutions for these scenarios.

6.2 Future Work

The analyses and results in this thesis raise a variety of interesting questions for future research.

For a peer-to-peer file distribution system, one question of interest is how to improve the scalability of the LP and a distributed algorithm. While the problem formulation provides a solution in time that is polynomial in the system size, the use of the time-expanded graph requires demanding computation performed by a centralized authority.

An alternative strategy might consider genetic algorithm (GA) to find a less costly coding solution by minimizing the number of coding nodes in the network [47]. Another important direction is investigation of practical issues such as transmission and queuing delay. In a practical network, there are constraints on transmission size, minimum packet size for coding, etc.

For the study on a peer-to-peer anonymous networking, it would be interesting to analyze more complicated networks such as triangle-shaped networks and non-complete graphs. As discussed in Section 3.3.2, vertex-cuts play an important role in performance. Triangular topologies may be useful. Since a vertex-cut reveals all downstream network information, and therefore the closer to the source a vertex-cut is, the lower the adversary's uncertainty about the sink location. Increasing the size of a cut near the source node could help alleviate this increased vulnerability, as shown in Fig. 6.1. Since we don't want the adversary to learn its subgraph location from its in-degree, non-complete graphs may be useful. Another interesting direction is to investigate a good distribution of sink location in the subgraph. In this thesis, we assumed that a sink is located uniformly at random in the subgraph, but this distribution is not necessarily optimal. Another very interesting family of questions arises if passive adversaries are replaced by adversaries capable of active attacks. Such scenarios may lead to the combination of the given coding framework with erasure and error correction coding. Moreover, by using the same framework as congestion analysis for the data transfer phase, we may study the robustness of the system against adversarial disruption.

For robust subgraph construction under uncertainty, one interesting open problem concerns extension of the results to more general network problems, such as multicast and multicommodity flow.

Finally, for a prioritized communication problem, we may consider more complicated networks such as multicast networks, for which one could take advantage of network coding.

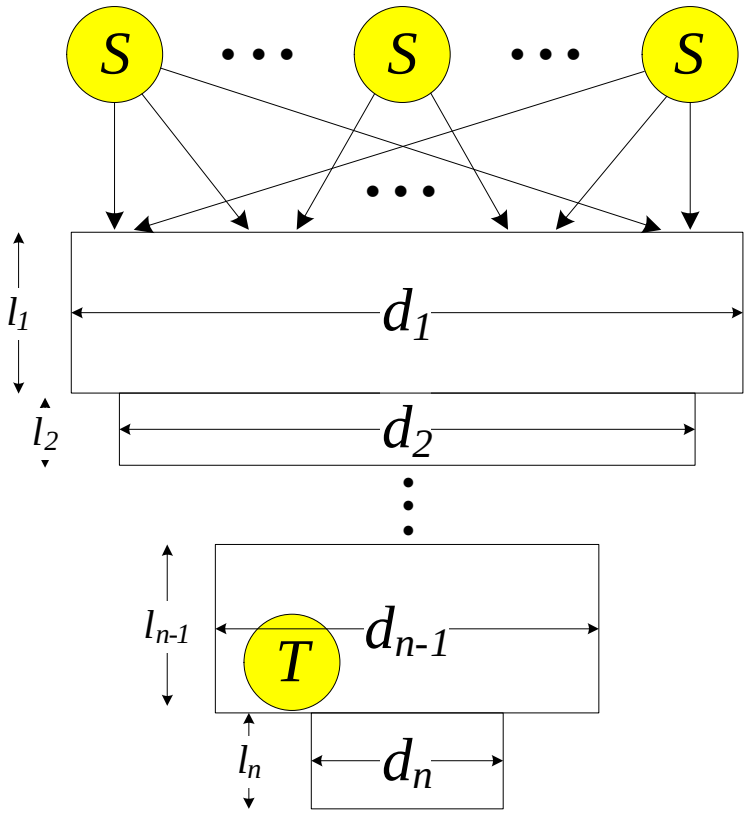


Figure 6.1: A generalized triangle-shape subgraph: to make it harder to form a vertex-cut close to the source node, we have $d_1 \geq d_2 \geq \ldots \geq d_n \geq 1$ and $l_i > 0, \forall i \in [1,\ n]$. Note that a sink is located randomly within the subgraph.

Bibliography

[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, *Network information flow*, IEEE Transactions on Information Theory **46** (2000), no. 4, 1204–1216.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network flows: theory, algorithms, and applications*, Prentice Hall, Upper Saddle River, NJ, 1993.

[3] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, *Priority encoding transmission*, IEEE Transactions on Information Theory **42** (1996), no. 6, 1737–1744.

[4] D. Applegate and E. Cohen, *Making routing robust to changing traffic demands: algorithms and evaluation*, IEEE/ACM Transactions on Networking **14** (2006), no. 6, 1193–1206.

[5] A. Atamtürk and M. Zhang, *Two-stage robust network flow and design under demand uncertainty*, Operations Research **55** (2007), no. 4, 662.

[6] A. Ben-Tal, A. Goryashko, E. Guslitser, and A. Nemirovski, *Adjustable robust solutions of uncertain linear programs*, Mathematical Programming **99** (2004), no. 2, 351–376.

[7] A. Ben-Tal and A. Nemirovski, *Robust convex optimization*, Mathematics of Operations Research **23** (1998), 769–805.

[8] S.P. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, Cambridge, UK, 2004.

[9] G. Brassard and P. Bratley, *Fundamentals of algorithmics*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1996.

[10] A.R. Calderbank and N. Seshadri, *Multilevel codes for unequal error protection*, IEEE Transactions on Information Theory **39** (1993), no. 4, 1234–1248.

[11] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach, *Secure routing for structured peer-to-peer overlay networks*, ACM SIGOPS Operating Systems Review **36** (2002), no. SI, 299–314.

[12] M. Castro, P. Druschel, A.M. Kermarrec, and A.I.T. Rowstron, *Scribe: A large-scale and decentralized application-level multicast infrastructure*, IEEE Journal on Selected Areas in Communications **20** (2002), no. 8, 1489–1499.

[13] C.S. Chang, T. Ho, and M. Effros, *On robust network coding subgraph construction under uncertainty*, Proceedings of the 42nd Asilomar Conference on Signals, Systems and Computers, 2008, pp. 2211–2215.

[14] C.S. Chang, T. Ho, M. Effros, M. Médard, and B. Leong, *Issues in peer-to-peer networking: A coding optimization approach*, Proceedings of the 6th IEEE International Symposium on Network Coding (NetCod), 2010, pp. 1–6.

[15] C.S. Chang and M.A. Klimesh, *Coding for parallel links to maximize expected decodable-message value*, Proceedings of the 44th Annual Conference on Information Sciences and Systems (CISS), 2010, pp. 1–6.

[16] D.L. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, Communications of the ACM **24** (1981), no. 2, 84–90.

[17] C. Chekuri, G. Oriolo, MG Scutella, and FB Shepherd, *Hardness of robust network design*, Networks **50** (2007), no. 1, 50–54.

[18] D.M. Chiu, R.W. Yeung, J. Huang, and B. Fan, *Can network coding help in P2P networks?*, 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (2006), 1–5.

[19] B. Cohen, *Incentives build robustness in BitTorrent*, Workshop on Economics of Peer-to-Peer Systems, vol. 6, 2003.

[20] T. Cover, *Broadcast channels*, IEEE Transactions on Information Theory **18** (1972), no. 1, 2–14.

[21] I. Csiszár and J.G. Körner, *Information theory: coding theorems for discrete memoryless systems*, Academic Press, Inc. Orlando, FL, USA, 1982.

[22] S. Deb, M. Médard, and C. Choute, *Algebraic gossip: A network coding approach to optimal multiple rumor mongering*, IEEE Transactions on Information Theory **52** (2006), no. 6, 2486–2507.

[23] C. Diaz, S. Seys, J. Claessens, and B. Preneel, *Towards measuring anonymity*, Proceedings of the 2nd International Conference on Privacy Enhancing Technologies, 2002, pp. 54–68.

[24] R. Dingledine, N. Mathewson, and P. Syverson, *Tor: The second-generation onion router*, Proceedings of the 13th Conference on USENIX Security Symposium, 2004, p. 21.

[25] R. Dougherty, C. Freiling, and K. Zeger, *Networks, matroids, and non-Shannon information inequalities*, IEEE Transactions on Information Theory **53** (2007), no. 6, 1949–1969.

[26] L. El Ghaoui, F. Oustry, and H. Lebret, *Robust solutions to uncertain semidefinite programs*, SIAM Journal of Optimization **9** (1998), 33–52.

[27] G.M. Ezovski, A. Tang, and L. Andrew, *Minimizing average finish time in p2p networks*, Proceedings of IEEE International Conference on Computer Communications (INFOCOM), 2009, pp. 594–602.

[28] Y. Fan, Y. Jiang, H. Zhu, J. Chen, and X. Shen, *Network coding based privacy preservation against traffic analysis in multi-hop wireless networks*, IEEE Transactions on Wireless Communications **10** (2011), no. 3, 834–843.

[29] H. Federrath, A. Jerichow, and A. Pfitzmann, *Mixes in mobile communication systems: Location management with privacy*, Proceedings of Information Hiding: the 1st International Workshop, vol. 1174, 1996, pp. 121–135.

[30] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, *Network coding: an instant primer*, SIGCOMM Comput. Commun. Rev. **36** (2006), no. 1, 63–68.

[31] M.J. Freedman and R. Morris, *Tarzan: A peer-to-peer anonymizing network layer*, Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS), 2002, pp. 193–206.

[32] S. Fujishige, *Submodular functions and optimization*, Elsevier, Amsterdam, The Netherlands, 2005.

[33] P. Gasti, A. Merlo, G. Ciaccio, and G. Chiola, *On the integrity of network coding-based anonymous p2p file sharing networks*, Proceedings of the 9th IEEE International Symposium on Network Computing and Applications (NCA), 2010, pp. 192–197.

[34] C. Gkantsidis, J. Miller, and P. Rodriguez, *Anatomy of a P2P content distribution system with network coding*, 5th International Workshop on P2P System (IPTPS) (2006).

[35] C. Gkantsidis and P. R. Rodriguez, *Network coding for large scale content distribution*, Proceedings of IEEE International Conference on Computer Communications (INFOCOM), 2005, pp. 2235–45.

[36] I.A. Goldberg, *A pseudonymous communications infrastructure for the internet*, Ph.D. thesis, UC Berkeley, 2000.

[37] D.M. Goldschlag, M.G. Reed, and P.F. Syverson, *Hiding routing information*, Proceedings of Information Hiding: the 1st International Workshop, vol. 1174, 1996, pp. 137–150.

[38] E. Guslitser, *Uncertainty-immunized solutions in linear programming*, Master's thesis, Technion-Israel Institute of Technology, 2002.

[39] T. Ho and D.S. Lun, *Network coding: An introduction*, Cambridge University Press, 2008.

[40] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, *A random linear network coding approach to multicast*, IEEE Transactions on Information Theory **52** (2006), no. 10, 4413–4430.

[41] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and Lmga Tolhuizen, *Polynomial time algorithms for multicast network code construction*, IEEE Transactions on Information Theory **51** (2005), no. 6, 1973–1982.

[42] K. Jain, L. Lovasz, and P. A. Chou, *Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding*, Distributed Computing **19** (2007), no. 4, 301–311.

[43] A. Jerichow, J. Muller, A. Pfitzmann, B. Pfitzmann, and M. Waidner, *Real-time mixes: A bandwidth-efficient anonymity protocol*, IEEE Journal on Selected Areas in Communications **16** (1998), no. 4, 495–509.

[44] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica **4** (1984), no. 4, 373–395.

[45] S. Katti, D. Katabi, and K. Puchala, *Slicing the onion: Anonymous routing without pki*, ACM HotNets, 2005.

[46] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, *XORs in the air: Practical wireless network coding*, IEEE/ACM Transactions on Networking **16** (2008), no. 3, 497–510.

[47] M. Kim, *Evolutionary approaches toward practical network coding*, Ph.D. thesis, Massachusetts Institute of Technology, MA USA, 2008.

[48] R. Koetter and M. Médard, *An algebraic approach to network coding*, IEEE/ACM Transactions on Networking **11** (2003), no. 5, 782–795.

[49] A. Leon-Garcia, *Probability and random processes for electrical engineering*, Addison Wesley, 1994.

[50] S. Y. R. Li, R. W. Yeung, and N. Cai, *Linear network coding*, IEEE Transactions on Information Theory **49** (2003), no. 2, 371–381.

[51] Z. Li and B. Li, *Network coding in undirected networks*, Proceedings of the 38th Annual Conference on Information Sciences and Systems (CISS), 2004.

[52] F. Matúš, *Two constructions on limits of entropy functions*, IEEE Transactions on Information Theory **53** (2007), no. 1, 320–330.

[53] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, *Scalable onion routing with torsk*, Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), 2009, pp. 590–599.

[54] M. Médard and A. Sprintson, *Network coding: Fundamentals and applications*, Academic Press, 2011.

[55] M. Mehyar, W. Gu, S.H. Low, M. Effros, and T. Ho, *Optimal strategies for efficient peer-to-peer file sharing*, Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), vol. 4, 2007, pp. 1337–1340.

[56] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D.S. Wallach, *Ap3: Cooperative, decentralized anonymous communication*, Proceedings of the 11th Workshop on ACM SIGOPS European Workshop, 2004, p. 30.

[57] S. Mudchanatongsuk, F. Ordóñez, and J. Liu, *Robust solutions for network design under transportation cost and demand uncertainty*, Journal of the Operational Research Society **59** (2008), no. 5, 652–662.

[58] J. Mundinger and R. Weber, *Efficient file dissemination using peer-to-peer technology*, University of Cambridge, Statistical Laboratory Research Report 2004-01 (2004).

[59] A. Nambiar and M. Wright, *Salsa: a structured approach to large-scale anonymity*, Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS), 2006, pp. 17–26.

[60] Y. Nesterov and A. Nemirovsky, *Interior point polynomial methods in convex programming*, Studies in Applied Mathematics **13** (1994).

[61] F. Ordóñez and J. Zhao, *Robust capacity expansion of network flows*, Networks **50** (2007), no. 2, 136–145.

[62] L. Øverlier and P. Syverson, *Improving efficiency and simplicity of tor circuit establishment and hidden services*, Proceedings of the 7th International Conference on Privacy Enhancing Technologies, 2007, pp. 134–152.

[63] J.G. Oxley, *Matroid theory*, Oxford University Press, New York, NY, 2006.

[64] A. Pfitzmann, B. Pfitzmann, and M. Waidner, *Isdn-mixes: Untraceable communication with very small bandwidth overhead*, Proceedings of GI/ITG-Conference on Communication in Distributed Systems, 1991, pp. 451–463.

[65] D. Qiu and R. Srikant, *Modeling and performance analysis of BitTorrent-like peer-to-peer networks*, SIGCOMM Comput. Commun. Rev., ACM New York, NY, USA, 2004, pp. 367–378.

[66] M. Reed, P. Syverson, and D. Goldschlag, *Protocols using anonymous connections: Mobile applications*, Proceedings of the 5th International Workshop on Security Protocols, 1998, pp. 13–23.

[67] M.G. Reed, P.F. Syverson, and D.M. Goldschlag, *Anonymous connections and onion routing*, IEEE Journal on Selected Areas in Communications **16** (1998), no. 4, 482–494.

[68] M.K. Reiter and A.D. Rubin, *Crowds: Anonymity for web transactions*, ACM Transactions on Information and System Security **1** (1998), no. 1, 66–92.

[69] M. Rennhard and B. Plattner, *Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection*, Proceedings of the ACM Workshop on Privacy in the Electronic Society, 2002, pp. 91–102.

[70] A. Rowstron and P. Druschel, *Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems*, Proceedings of the IFIP/ACM Middleware, 2001, pp. 329–350.

[71] R. Schollmeier, *A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications*, Proceedings of the 1st International Conference on Peer-to-Peer Computing, 2001, pp. 101–102.

[72] A. Serjantov and G. Danezis, *Towards an information theoretic metric for anonymity*, Proceedings of the 2nd International Conference on Privacy Enhancing Technologies, 2002, pp. 259–263.

[73] D. Silva and F.R. Kschischang, *Rank-metric codes for priority encoding transmission with network coding*, Proceedings of the 10th Canadian Workshop on Information Theory, 2007, pp. 81–84.

[74] J.C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*, vol. 64, John Wiley and Sons, 2003.

[75] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications, 2001, pp. 149–160.

[76] A. Wald, *Contributions to the theory of statistical estimation and testing hypotheses*, The Annals of Mathematical Statistics **10** (1939), no. 4, 299–326.

[77] ———, *Statistical decision functions which minimize the maximum risk*, Annals of Mathematics **46** (1945), no. 2, 265–280.

[78] ———, *Statistical decision functions*, John Wiley, New York (1950).

[79] J. Wang, J. Wang, C. Wu, K. Lu, and N. Gu, *Anonymous communication with network coding against traffic analysis attack*, Proceedings of IEEE International Conference on Computer Communications (INFOCOM), 2011, pp. 1008–1016.

[80] Y. Wu, Y.C. Hu, J. Li, and P.A. Chou, *The delay region for P2P file transfer*, Proceedings of IEEE International Symposium on Information Theory (ISIT), 2009, pp. 834–838.

[81] R.W. Yeung, *Information theory and network coding*, Springer-Verlag, 2008.