Undirected Multicast Network Coding Gaps via Locally Decodable Codes

Mark Braverman* Zhongtian He†

Abstract

The network coding problem asks whether data throughput in a network can be increased using coding (compared to treating bits as commodities in a flow). While it is well-known that a network coding advantage exists in directed graphs, the situation in undirected graphs is much less understood – in particular, despite significant effort, it is not even known whether network coding is helpful at all for unicast sessions.

In this paper we study the multi-source multicast network coding problem in *undirected* graphs. There are k sources broadcasting each to a subset of nodes in a graph of size n. The corresponding combinatorial problem is a version of the Steiner tree packing problem, and the network coding question asks whether the multicast coding rate exceeds the tree-packing rate.

We give the first super-constant bound to this problem, demonstrating an example with a coding advantage of $\Omega(\log k)$. In terms of graph size, we obtain a lower bound of $2^{\tilde{\Omega}(\sqrt{\log \log n})}$. We also obtain an upper bound of $O(\log n)$ on the gap.

Our main technical contribution is a new reduction that converts locally-decodable codes in the low-error regime into multicast coding instances. This gives rise to a new family of explicitly constructed graphs, which may have other applications.

*Princeton University, Research supported in part by the NSF Alan T. Waterman Award, Grant No. 1933331.
†Princeton University

Contents

1 Introduction

Optimizing network throughput is a key problem in both combinatorial optimization and coding theory. Relevant objectives include maximizing unicast and broadcast throughput, improving error resilience, enhancing security, etc. Traditional routing treats data packets as physical commodities, giving rise to combinatorial problems such as max-flow and multi-commodity-flow. Network coding extends the routing model by allowing data to be encoded and combined at intermediate nodes, potentially offering significant advantages in various settings. While the benefits of network coding in directed graphs are well understood (e.g., [ACLY00, LYC03, HMK+06, YLCZ06, FS+07, HL08]), key questions in the undirected setting remain open despite much effort.

In fact, a well-known conjecture[1] states that network coding offers *no* advantage in throughput for *multi-source unicast* in undirected graphs [LL04a, HKL04], meaning that the best achievable throughput can be obtained solely through multi-commodity flow routing. Not only the problem is interesting on its own, but it also has strong implications in complexity theory, where the positive resolution of the conjecture would imply lower bounds in external memory algorithm complexity [FHLS19], in cell-probe model [AHJ+06], and even super-linear circuit lower bounds for very natural mathematical problems such as integer multiplication [AFKL19]. Given these connections, it is not surprising that despite numerous attempts [LL04b, HKL06, KS06, LM09, XLWH14, BGS17, YLLW17, HWZ20] and resolutions on special classes of instances [OS81, AHJ+06, JVY06, KS06], the conjecture remains open. In contrast, the advantage is known to be $\Omega(n)$ for multi-source unicast in directed graphs of size n [LL04b, HKL06].

Another relevant setting is *single-source multicast*, where one source transmits information to multiple-destinations. In undirected graphs, the known network coding advantage, also called the *coding gap*, lies between 8/7 [AC04] and 2 [LL04a]. The coding gap is defined as the ratio of the network coding throughput to the *Steiner tree packing number*, where the latter serves as the non-coding multicast benchmark. We emphasize that a Steiner tree packing combinatorially implies a valid network coding solution: given such a packing, one can transmit information along the Steiner trees, since once a vertex receives some information, it is allowed to forward it to multiple successors. In directed graphs, the single-source multicast problem is studied in the seminal work on network coding [ACLY00], which shows that an exact optimal network coding solution can always be achieved. Moreover, the coding gap can be as large as $\Omega(\log n)$ [AC04, JSC+05], matching the integrality gap of linear programming relaxation for the directed Steiner tree problem.

Thereby a natural problem *multi-source multicast* generalizing both settings above arises. In this setting, we have multiple source nodes, each communicating its information to a set of destination nodes. The combinatorial non-coding throughput of this problem is *multi Steiner tree packing number*. We study the ratio of multi-source multicast network coding throughput to the corresponding multi Steiner tree packing number in undirected graphs:

"How large the advantage can network coding obtain for multi-source multicast throughput?"

Even though the multi-source multicast problem is well-studied, no general asymptotic bounds for the network coding gap in undirected graphs were previously known. In a seminal work, Ahlswede

[1] The conjecture is known by several names, including the Network Coding Conjecture, Li and Li's Conjecture, and the Multiple Unicast Conjecture.

et al. [ACLY00] study the multi-source multicast problem in directed graphs, establishing an optimal network throughput for a special case where all sources share the same set of sinks, while leaving the general case as an open problem. Since then, the multi-source multicast problem has been studied extensively from both theoretical and empirical perspectives. To name a few, this includes an exact algebraic formulation of feasibility [KM03], cut-set outer bounds (e.g., [Cov99, HKL06, TGC16]), extensions to noisy channels [LKEGC11], and settings with security guarantees [CCMG18]. Notably, Langberg and Médard [LM09] show that, for uniform demands, the multi-source unicast non-coding throughput is at least one-third of the multi-source multicast coding throughput. In this context, the multicast problem is defined by modifying each source's sink set to include all sinks from the unicast problem. This connection positions the multi-source multicast problem in undirected graphs as a partial step toward either proving or disproving the network coding conjecture.

In this work, we prove asymptotic lower and upper bounds for the multi-source multicast problem in undirected graphs. Let n denote the graph size and k the number of sources. We show the existence of network coding instances with a coding gap of $\Omega(\log k)$ for k-source multicast sessions, which is the first super-constant bound to this problem. To achieve these coding gaps, we develop novel connections between network coding and locally decodable codes (LDCs), which constitutes the main technical contribution of this work.

In terms of graph size n, this coding gap translates into $2^{\tilde{\Omega}(\sqrt{\log\log n})}$ [2], utilizing a particular family of LDCs known as matching vector codes [Efr09, DGY11, Yek12, BDL13, DH13], which excel in low-query regimes. Since our reduction is black-box, any improvement in LDC constructions would directly lead to stronger lower bounds on the network coding gap. We elaborate on this connection in the technique overview section.

Theorem 1.1. *There exists a family of network coding instances for k multicast sessions in undirected graphs, that have coding gaps at least $\Omega(\log k)$. In term of graph size n, the gap is at least $2^{\Omega\left(\sqrt{\log\log n/\log\log\log n}\right)}$.*

For the upper bound, we show that the coding gap for multi-source multicast sessions is at most $O(\log n)$. Our approach relates the network coding gap to the LP integrality gap for the dual of the multi Steiner packing problem. This integrality gap can, in turn, be bounded using an $O(\log n)$-approximate cut-tree packing [Räc08], a technique that has also been used in the context of oblivious routing.

Theorem 1.2. *The multi-source multicast coding gap in undirected graphs is at most $O(\log n)$.*

This matches the best known upper bound for the coding gap of multi-source unicast in the undirected graph. It is worth noting that, any improvement of the upper bound to $o(\log n)$ even in the multi-source unicast setting would yield a super-linear circuit lower bound for integer multiplication [AFKL19].

Unlike graph size n, in terms of the number of sources k, the best known upper bound remains the trivial $O(k)$. Closing this exponential gap is an intriguing open problem. In contrast, for multi-source unicast, an upper bound of $O(\log k)$ on the network coding gap is known, as a consequence of the approximate max-flow min-cut theorem (also known as the multicommodity flow-cut gap) [AR98, LLR95].

[2] We use the notation $\tilde{\Omega}(f(n))$ to denote $\Omega\left(f(n)/\operatorname{poly}\log f(n)\right)$, i.e., hiding $\operatorname{poly}\log f(n)$ factors.

The observations and results of this work raise several compelling open problems, as the multi-source multicast setting admits a variety of approaches from both coding theory and graph theory. In particular, improving the lower bound in terms of the graph size n may be possible through the construction of more efficient locally decodable codes (LDCs)—especially via a novel variant we introduce, called robust distance LDC codes, discussed in Section 4.2. Closing the exponential gap in terms of the number of sources k likely requires new combinatorial insights in graph theory. Beyond being interesting on their own, progress on these questions may also serve as intermediate steps toward resolving the network coding conjecture.

1.1 Related Works

Completion Time While it remains unknown whether network coding provides an advantage for the throughput of the multi-source unicast problem in undirected graphs, coding gaps do arise when considering completion time, also known as makespan, instead of throughput. The makespan problem is a generalization of throughput, as the maximum throughput can be defined as $\sup_{r\to\infty} r/C(r)$, where $C(w)$ is the makespan for the instance after increasing all demands by a factor of r. For the makespan of multi-source unicast in undirected graphs, Haeupler et al. [HWZ20] establish a polylogarithmic coding gap in terms of the number of sources k. Additionally, they prove an upper bound that is polylogarithmic in k and the ratio of the summation to the smallest demand for k-source unicast instances. It is an interesting open problem to bound the makespan coding gap for multi-source multicast sessions. The completion time of more general distributed computing tasks with security guarantees can also be improved using network coding [HPY22, Par23].

Graph Product Previous lower bounds on the network coding gap in undirected graphs are primarily obtained via the graph product technique—a powerful method that amplifies a constant gap into an asymptotically large one. This approach was used by Braverman et al. [BGS17] to establish a dichotomy in multi-source unicast throughput [BGS17]: the network coding gap for multi-source unicast thoughput is either exactly 1 or at least $\Omega((\log n)^c)$ for some constant $c > 0$. Haeupler et al. [HWZ20] later generalized this framework to show a polylogarithmic coding gap for completion time. While the graph product technique is very useful for establishing lower bounds for undirected multi-source unicast, its applicability to the undirected multicast setting remains unclear. In directed graphs, graph product has been applied to prove polynomial separations between linear and non-linear codes in network coding [BKL11, Lov14].

Locally Recoverable Codes and Fountain Codes Our use of LDCs to improve network coding throughput naturally connects to the broader landscape of fault-tolerant data access in distributed systems. In particular, locally recoverable codes (LRCs)[3] have been extensively studied for their role in reducing repair costs in distributed storage settings (e.g., [HSX+12, TB14, PD14, CM15, BTV17, GXY19, MPK19, Mic19, CMST21, KSCM23]). LRCs aim to minimize the number of nodes accessed during data recovery—an idea closely related to our goal of minimizing the number of queries in LDC-based constructions. Similar motivations underlie work on fountain codes which addressed efficient data dissemination under erasure [BLMR98, Lub02, Sho06].

[3] Also known as locally reconstructible or locally repairable codes.

1.2 Preliminaries

Network Coding A *multi-source multicast instance* $\mathcal{M} = (G, \mathcal{S})$ is defined over a communication network represented by an undirected graph $G = (V, E)$, with edge capacities given by a function $c : E \to \mathbb{R}_+$. The instance consists of k *sessions*, denoted by $\mathcal{S} = \{(s_i, R_i, d_i)\}_{i=1}^k$, where each session is specified by a source vertex $s_i \in V$, a set of sink vertices $R_i \subseteq V$, and a communication demand $d_i \in \mathbb{R}_+$. Let M_i denote the set of all messages that source s_i wishes to send, and define $M := \prod_{i=1}^k M_i$ as the set of all message combinations across sessions. Each message $m_i \in M_i$ is sampled independently from distribution $\mathcal{D}_i$ to produce a distribution $\mathcal{D}$ of M's.

The formal definition of a *network coding solution* is deferred to Section A. Here, we provide a simplified version that, while restricting the full expressive power of network coding, is sufficient for deriving our lower bounds. Given an undirected graph G, the solution specifies a direction for each edge e to form a directed graph $\hat{G}$, an associated alphabet $\Gamma(e)$, and an encoding function $f_e : M \to \Gamma(e)$ that determines the symbol transmitted along edge e. These functions must satisfy the following conditions:

- **Correctness:** Each sink must be able to recover the message from its corresponding source.
- **Causality:** The symbol transmitted on each edge e must be computable from the symbols received at its tail vertex.

We assume that $\hat{G}$ is a directed acyclic graph (DAG). Although the general case may involve cycles, the full definition—based on time-expanded graphs—is presented in Section A. That general framework is only required when we analyze upper bounds on network coding throughput in Section 3. The *throughput* of a network coding solution is defined as the largest value r for which there exists a *scale* $c^* \geq 0$ (i.e., allowing block length, time, or packet size expansion) such that the following conditions hold:

- For each source s_i, the entropy of its message satisfies $H(m_i) \geq r \cdot d_i \cdot c^*$.
- For each edge $e \in E$, the entropy of the information transmitted along e is at most $c_e \cdot c^*$.

The *throughput* of a multi-source multicast instance is the throughput of the network coding solution maximized over all possible distributions $\mathcal{D}$. Note that the common scale c^* in the throughput definition is important: in the network coding literature, throughput characterizes the asymptotic feasibility of the solution.

Multi Steiner Tree Packing The non-coding throughput of a multi-source multicast problem is captured by the *multi-Steiner packing number*. It is defined as the largest value τ such that, for each source s_i, there exists a Steiner tree packing with terminal set $S_i := R_i \cup \{s_i\}$ and total weight $\tau \cdot d_i$, subject to the constraint that the combined packing across all sources respects the edge capacities. This quantity characterizes the maximum achievable throughput in the absence of network coding—that is, when information can be duplicated but not encoded. A formal LP formulation is given in Section B.

Locally Decodable Codes Denote by $\Delta(\mathbf{x}, \mathbf{y})$ the Hamming distance between vectors $\mathbf{x}$ and $\mathbf{y}$. Denote by $\mathcal{A}^{\mathbf{y}}$ an algorithm that accesses the vector $\mathbf{y}$ through coordinate queries. Let $\mathbf{F}$ be a finite field. A code $\mathcal{C} : \mathbf{F}^k \to \mathbf{F}^N$ is said to be (q, δ, ϵ)-locally decodable if there exists a randomized decoding algorithm $\mathcal{A}$ such that:

1. For all $\mathbf{x} \in \mathbf{F}^k, i \in [k]$ and all $\mathbf{y} \in \mathbf{F}^N$ satisfying $\Delta(\mathcal{C}(\mathbf{x}), \mathbf{y}) \leq \delta$, we have

$$\Pr[\mathcal{A}^{\mathbf{y}}(i) = \mathbf{x}_i] \geq 1 - \epsilon,$$

   where the probability is over the internal randomness of $\mathcal{A}$.

2. The algorithm $\mathcal{A}$ makes at most q queries to $\mathbf{y}$.

The *rate* of a code is defined as the ratio of the message length to the codeword length, i.e., k/N. The parameter q, known as the *query complexity*, denotes the number of codeword symbols accessed during decoding. The parameter δ is referred to as the *decoding radius*—the maximum fraction of errors the decoder can tolerate while still recovering individual message symbols with high probability. We say a code satisfies the *linearity of decoding* property (or simply that the code is *linear*) if the decoder $\mathcal{A}$ always outputs a linear combination of the queried codeword symbols.

Definition 1.3 (Perfect Smoothness)**.** A code satisfies the *perfect smoothness* if, for any $\mathbf{x} \in \mathbf{F}^k$ and any $i \in [k]$, each query made by $\mathcal{A}(i)$ is uniformly distributed over the coordinate set $[N]$, and the decoder always returns the correct message $\mathbf{x}_i$ when given an uncorrupted codeword, i.e. $\mathcal{A}^{\mathcal{C}(\mathbf{x})}(i) = \mathbf{x}_i$.

The following fact will be useful in the construction of matching vector codes.

Fact 1.4 ([Tre04])**.** *Any code that satisfies perfect smoothness and makes q queries is also $(q, \delta, q\delta)$-locally decodable for all $\delta < \frac{1}{q}$.*

1.3 Technique Overview

Lower Bound. The core technical contribution is a reduction from the network coding gap to locally decodable codes.

Lemma 1.5 (Main Lemma)**.** *Suppose there exists a linear $(q, \delta, q\delta)$-locally decodable code $\mathcal{C} : \mathbf{F}^k \to \mathbf{F}^N$ satisfies perfect smoothness. Then, the undirected network coding gap for multi-source multicast is at least $\Omega((\delta \log k)/q)$, with the graph size in the gap instance given by $n = \Theta(Nk)$.*

This lemma allows us to establish network coding gaps using basic LDCs. For instance, the Hadamard code is a linear $(2, \frac{1}{4}, \frac{1}{2})$-locally decodable code with perfect smoothness, which immediately implies network coding gaps of $\Omega(\log k)$ and $\Omega(\log \log n)$. To achieve a stronger lower bound in terms of n, an LDC with a better rate is required.

In the sub-logarithmic query regime, the best rate is achieved by matching vector (MV) codes [Efr09], formally stated in Lemma 1.6, with the proof deferred to Section D. Notably, Lemma 1.6 differs from previous results in several ways. First, the MV codes in [Efr09] were analyzed in the

constant query regime using Grolmusz's set systems, which become suboptimal for a larger number of queries. Additionally, the MV codes in [DGY11, Yek12] require both the distance δ and error ϵ to be constant, which slightly increases the query complexity to $t^{O(t)}$. Here, we provide a version optimized for our setting.

Lemma 1.6 (Matching Vector Codes – optimized for network coding gap constructions)**.** *For $t, k \geq 2$, there exists a linear q-query locally decodable code over $\mathbf{F} = GF(2^b)$ with $q = 2^t, b = \tilde{O}(t)$ that encodes message of length k to codeword of length*

$$\exp\exp\left(O\left((\log k)^{1/t}(\log\log k)^{1-1/t}\cdot t\ln t\right)\right).$$

Moreover, the code is $(q, \delta, q\delta)$-locally decodable for any $\delta < 1/q$, and satisfies perfect smoothness.

Proof of Theorem 1.1. By applying Lemma 1.5 and the Hadamard code, which is a $(2, \frac{1}{4}, \frac{1}{2})$-locally decodable code, we immediately obtain the lower bound $\Omega(\log k)$.

To derive lower bounds in terms of n, we set $t = \frac{1}{4}\log\log k$ and $\delta = \frac{1}{4q}$ in Lemma 1.6, yielding a $\left((\log k)^{\frac{1}{4}}, \frac{1}{4}(\log k)^{-\frac{1}{4}}, \frac{1}{4}\right)$-locally decodable code. This encodes a message of length k into codewords of length

$$N = \exp\exp(O((\log\log k)^2\log\log\log k)).$$

Rearranging terms, we obtain

$$\log k = 2^{\Omega\left(\sqrt{\log\log N/\log\log\log N}\right)}.$$

Using this LDC and Lemma 1.5, we derive a network coding gap of $\Omega(\sqrt{\log k})$, which translates to $2^{\Omega\left(\sqrt{\log\log n/\log\log\log n}\right)}$ in terms of n, since n is polynomially related to N, and the relation between N and k follows from our previous derivation. □

In Section 4.2, we propose a relaxed variant of LDCs tailored for applications in network coding gaps, as our approach does not require their full error-correcting capabilities. We believe this variant will be useful in closing the gap between the lower bound in Theorem 1.1 and the $O(\log n)$ upper bound.

Upper Bound. We introduce a generalized notion of the sparsest cut in the multi-Steiner setting, defined as the objective of the integral solution to the dual LP of the multi-Steiner packing problem (see Section B). This quantity also serves as an upper bound on the network coding throughput, which we formally prove under the information-theoretic definition of network coding. This connection reduces the problem of bounding the network coding throughput to analyzing the LP integrality gap of the multi-Steiner packing problem. Leveraging the $O(\log n)$-approximate cut-tree packing developed for oblivious routing [Räc08], we obtain an $O(\log n)$ upper bound on the network coding gap. This result can thus be interpreted more strongly: the gap between network coding throughput and oblivious Steiner packing is at most $O(\log n)$.

Presentation Overview In Section 2, we demonstrate how to construct coding gap instances using LDCs. This section is divided into two parts: in Section 2.1, we present a simple sampling-based construction to illustrate how LDCs can be applied to network coding; and in Section 2.2, we prove the main lemma by amplifying the coding gap using random binary trees. Although the sampling construction in Section 2.1 is suboptimal, it has the merits of simplicity and motivates a new formulation of LDCs, which we introduce in Section 4.2. We briefly discuss the limitation of existing LDCs to network coding in Section 4.1. In Section 3, we establish an upper bound on the network coding gap.

2 Gap Instances via Locally Decodable Codes

A *network coding gap instance* $I = (G, \mathcal{S}, F)$ is a multi-source multicast instance $\mathcal{M} = (G, \mathcal{S})$ defined over a connected graph G, along with a designated set of cut edges $F \subseteq E(G)$. We restrict our attention to instances where all edge capacities and demands are unit-valued, i.e., $c_e = 1$ for all $e \in E(G)$ and $d_i = 1$ for all $i \in [k]$. Let $\text{dist}_G(u, v)$ denote the shortest-path distance between nodes u and v in G. We say that a gap instance has parameters (a, b, f, m, r) if the following conditions hold:

1. The k-source multicast instance $\mathcal{M}$ admits a network coding solution achieving throughput a.
2. Removing the edges in F disconnects s_i from R_i for every $i \in [k]$.
3. Let $\text{dist}_{G\setminus F}(U) := \min_{\substack{u,v\in U \\ u\neq v}} \text{dist}_{G\setminus F}(u, v)$. Then for all $i \in [k]$, it holds that $\text{dist}_{G\setminus F}(R_i) \geq b$.
4. The number of cut edges is $f = |F|$.
5. The total number of sinks is $r = \sum_{i\in[k]} |R_i|$.
6. The graph G contains at most m edges, i.e., $|E(G)| \leq m$.

Lemma 2.1. *Let I be a gap instance with parameters (a, b, f, m, r). Then the network coding gap is at least $\frac{a\cdot r}{f+2m/b}$.*

Proof. Let τ denote the non-coding throughput, characterized by the multi-Steiner packing number. By the LP duality in Section B, the dual objective is an upper bound on τ. Now we shall assign dual variables: let $z_i = |R_i| + 1$ for every demand i; $y_e = 1$ for every edge $e \in F$ and $y_e = 2/b$ otherwise. We show that the dual constraints are satisfied. Clearly, all the variables are non-negative. Recall that we define $S_i := R_i \cup \{s_i\}$. For any i and tree T such that $S_i \subseteq V(T)$, we need to show that the constraint $\sum_{e\in T} y_e \geq z_i$ is satisfied. Let $r_T := |R_i|$ be the number of sinks spanned by T, and let x_T denote the number of cut edges F used by T.

By the definition of a gap instance, in $G \setminus F$, any two sinks in R_i are at distance at least b. We say a sink $t \in R_i$ is *close* to a cut edge $e = (u, v) \in T \cap F$ if, in $G \setminus F$, the vertex u is connected to s_i, v is not, and $\text{dist}_{G\setminus F}(t, v) < b/2$. By the distance condition of the gap instance, at most one

sink in R_i can be close to any such cut edge. Therefore, the number of close sinks in T is at most x_T.

For each one of the remaining sinks s, let $N(s, b/2)$ be the edges in $(b/2)$-neighborhood of s in $T \setminus F$. Since s is remote, we have $|N(s, b/2)| \geq b/2$. In addition, for each two distinct $s_1 \neq s_2$ we have $\text{dist}_{T\setminus F}(s_1, s_2) \geq \text{dist}_{G\setminus F}(s_1, s_2) \geq b$, and thus $N(s_1, b/2) \cap N(s_2, b/2) = \emptyset$. Therefore, the total number of non-cut edges in T is at least $(r_T - x_T) \cdot \frac{b}{2}$. Thereby the dual constraint is satisfied,

$$\sum_{e\in T} y_e \geq \frac{2}{b} \cdot (r_T - x_T)\frac{b}{2} + 1 \cdot x_T = r_T = z_i.$$

With this assignment, the dual objective is,

$$\frac{\sum_{e\in E} c_e y_e}{\sum_i d_i z_i} \leq \frac{f + 2m/b}{r}$$

which upper bounds the packing number τ. Given that the coding throughput is a, the network coding gap is at least

$$\frac{a}{\tau} \geq \frac{a \cdot r}{f + 2m/b}.$$

□

2.1 Sampling Construction via LDCs

Fix a perfectly smooth $(q, \delta, q\delta)$ code $\mathcal{C} : \mathbf{F}^K \to \mathbf{F}^N$, we will show that the following graph construction is a (a, b, f, m, r) gap instance for $a = 1, b = \log k/\log\log k, f = N, m = \Theta(rq)$, and $r = \Theta((\delta N \log k)/q)$. We only consider gap instances where all sources are co-located at the same vertex S. Our construction is based on a tripartite graph $G = (A, B, C, E)$, structured as follows:

- The first part A contains only the common source vertex S, i.e. $s_i = S$ for all $i \in [k]$. In a network coding solution, the message of s_i is represented as $\mathbf{x}_i \in \mathbf{F}$. Thereby, vertex S knows the entire message vector $\mathbf{x} \in \mathbf{F}^k$.
- The second part $B = \{w_j\}_{j=1}^N$ consists of N vertices and each is connected to S by an edge. In the network coding solution, each vertex $w_j \in B$ will receive from S one coordinate of the codeword $\mathcal{C}(\mathbf{x})_j$.
- The third part C is the disjoint union of sink sets: $C = \bigsqcup_{i=1}^k R_i$, where $R_i \cap R_j = \emptyset$ for all $i \neq j$. Each sink in R_i connects to q vertices in B that enable decoding of the message $\mathbf{x}_i$, using the local decodability of the code. In the remainder of this subsection, we describe in detail how to construct the third part.

We begin by identifying the property that the third part should satisfy. To apply Lemma 2.1, we set the cut edges F to be all edges adjacent to S, and aim to ensure that the pairwise distances between sinks in each R_i are not too small in $G \setminus F$. This is achieved by first generating a near-linear number of candidate sinks for each i, then retaining only a $(\log k)/k$ fraction via sampling.

A pruning subroutine is applied to eliminate sinks that are too close to others associated with the same source. Both subroutines are formalized below. The pseudocode is shown in Algorithm 1.

Sampling Subroutine For q-query LDCs, each vertex in R_i must have q neighbors in the second layer B that can successfully decode the message $\mathbf{x}_i$. The code's decoding radius is used to generate many disjoint q-queries of near-linear size. Specifically, we maintain a set of used indices $U \subseteq [N]$. As long as $|U| \leq \delta N$, there exists a set $D \subseteq [N] \setminus U$ of size q such that $\mathbf{x}_i$ can be decoded by querying $\mathcal{C}(\mathbf{x})$ at coordinates in D (see Fact 2.2 and line 8 in Algorithm 1). Denote by $\textsc{Decode}(i, U)$ the oracle that returns the corresponding set D. All indices in D are added to U.

With probability $\log k/k$, we add a vertex v to the sinks R_i and add edges from v to the vertices in B indexed by D. Note that D is added to U regardless of whether v is sampled. The process terminates when $|U| > \delta N$. The sampling subroutine runs from line 5 to line 14 in Algorithm 1.

Fact 2.2. *Let $C : \mathbf{F}^k \to \mathbf{F}^N$ be a $(q, \delta, q\delta)$-locally decodable code with perfect smoothness. Then for any index $i \in [k]$ and any subset $U \subseteq [N]$ with $|U| \leq \delta N$, there exists a set $D \subseteq [N] \setminus U$ of size at most q such that the message symbol $\mathbf{x}_i$ can be successfully decoded by querying the codeword $\mathcal{C}(\mathbf{x})$ at positions in D.*

Proof. By the perfect smoothness property, each of the q queries made by the decoder is uniformly distributed over $[N]$, so the probability that any single query falls in the corrupted set U is at most $\frac{|U|}{N} \leq \delta$. By the union bound, the probability that any of the q queries falls in U is at most $q\delta$. Therefore, with probability at least $1 - q\delta$, all q queries lie entirely within the uncorrupted set $\overline{U}$.

By the probabilistic method, this implies the existence of a fixed set $D \subseteq \overline{U}$ of size q such that the decoder queries exactly the positions in D with some fixed randomness. Since all queried positions are uncorrupted and the decoder behaves correctly on valid codewords by the perfect smoothness assumption, $\mathbf{x}_i$ can be deterministically decoded from the values of $\mathcal{C}(\mathbf{x})$ at positions in D. □

Pruning Subroutine We prune the set of sinks C to ensure that, within each R_i, the sinks are sufficiently far apart in the graph $G \setminus F$. Specifically, for each $i \in [k]$ and each sink $u \in R_i$, if there exists another sink $v \in R_i$ such that the distance between u and v in $G \setminus F$ is less than $\log k / \log \log k$, we remove u from R_i.

The pruning threshold is chosen to ensure that only a small number of sinks are removed from C, as formally shown in Lemma 2.3. After pruning, we apply Lemma 2.1 to establish a lower bound on the coding gap.

Lemma 2.3. *Assume that $q = o(\log k)$. Then, in the pruning subroutine of Algorithm 1, an $o_k(1)$[4] fraction of the sinks in C is removed with high probability.*

Proof. We analyze the pruning process in the graph $G \setminus F$. Fix any $i \in [k]$ and any sink $u \in R_i$. Let P_j be the number of vertices in B within distance $2j + 1$ from u. Initially, $P_0 = q$. For $j > 0$, we have the recurrence

$$\mathbf{E}[P_j] \leq q \log k \cdot \mathbf{E}[P_{j-1}],$$

[4]That is, a quantity tending to 0 as $k \to \infty$.

Algorithm 1 Generating Gap Instance via LDCs

Set $A \leftarrow \{S\}$ and $s_i \leftarrow S$ for all $i \in [k]$.
Set $B \leftarrow \{w_i\}_{i=1}^{N}$.
Set $F \leftarrow \{(S, w_i) \mid i \in [N]\}$ and initialize $E \leftarrow F$.
Initialize $R_i \leftarrow \emptyset$ for all $i \in [k]$.
for each $i = 1$ to k **do** ▷ Sampling Subroutine.
    Initialize $U \leftarrow \emptyset$.
    while $|U| \leq \delta N$ **do**
        Let $D \leftarrow \textsc{Decode}(i, U)$. ▷ Select q coordinates in $[N] \setminus U$ for decoding.
        with probability $(\log k)/k$:
            Add a new vertex v to R_i.
            Add edges (w_j, v) to E for each $j \in D$.
        Update $U \leftarrow U \cup D$.
    end while
end for
for each $i \in [k]$ and **each** $u \in R_i$ **do** ▷ Pruning Subroutine.
    if there exists $v \in R_i \setminus \{u\}$ such that $\text{dist}_{G \setminus F}(u, v) \leq \log k / \log\log k$ **then**
        Remove v from R_i.
    end if
end for
Set $C \leftarrow \bigcup_{i=1}^{k} R_i$.

since each vertex in C has q neighbors in B, and each vertex in B has at most k potential neighbors in C, with each edge sampled independently with probability $\log k/k$.

Now let Q_j be the number of vertices in $R_i \setminus \{u\}$ within distance $2j$ from u. Clearly, $Q_0 = 0$. For $j > 0$, the expectation $\mathbf{E}[Q_j]$ equals $\mathbf{E}[Q_{j-1}]$ plus the expected number of vertices in $R_i \setminus \{u\}$ that are at distance exactly $2j$ from u. The latter quantity can be upper bounded by the expected number of vertices in B at distance $2j-1$ from u, multiplied by the probability that a vertex in B is connected to a vertex in R_i. This probability is at most $\frac{\log k}{k}$ as the sampling procedure, implying

$$\mathbf{E}[Q_j] \leq \mathbf{E}[Q_{j-1}] + \frac{\log k}{k} \cdot \mathbf{E}[P_{j-1}] \leq \frac{(q \log k)^{j+1}}{k} \ .$$

Let $d^* = \frac{1}{2} \cdot \frac{\log k}{\log\log k}$. The algorithm prunes u if there exists another sink in R_i within distance $2d^*$ in $G \setminus F$. Using the bound above, we get

$$\mathbf{E}[Q_{d^*}] \leq \frac{(q \log k)^{d^*}}{k} = o(1),$$

and therefore

$$\Pr[Q_{d^*} \geq 1] \leq \mathbf{E}[Q_{d^*}] = o(1),$$

which gives an upper bound on the probability that u is pruned.

Since this holds for every $u \in R_i$ and all $i \in [k]$, the expected fraction of sinks pruned is $o(1)$. Applying Markov's inequality concludes the proof. □

We name this instance $I_S = (G, \mathcal{S}, F)$. Applying Lemma 2.1 to our gap instance I_S, we obtain the following lemma.

Lemma 2.4. *Suppose $\mathcal{C} : \mathbf{F}^k \to \mathbf{F}^N$ is a $(q, \delta, q\delta)$-locally decodable code with perfect smoothness, as used in Algorithm 1. Then the network coding gap for the resulting instance I_S is at least*

$$\Omega\left(\min\left\{\delta, \frac{1}{\log\log k}\right\} \cdot \frac{\log k}{q}\right).$$

Proof. For the network coding throughput, we consider each source message to be uniformly distributed over $\mathbf{F}$. Each edge in the network carries a symbol in $\mathbf{F}$, and each sink decodes its target symbol using q symbols from the second layer. Each vertex in B receives one coordinate of the locally decodable codeword, and each vertex in the third layer accesses q such coordinates.

Each source has entropy $\log s$, where $s = |\mathbf{F}|$. Since all demands are unit-valued, we scale the capacities by $c^* = \log s$, as in the definition of network coding solution. Consequently, the effective capacity of each edge is $1 \cdot c^* = \log s$, since all edge capacities are unit-valued. Under this scaling, the network supports a coding solution that delivers the full message to each sink, and therefore the coding throughput of I_S is at least $a = 1$.

We now analyze the remaining parameters required by Lemma 2.1 for I_S. By the sampling subroutine, the number of sinks $r = \Theta\left(\frac{\delta N}{q} \log k\right)$, which dominates the total number of vertices n. The total number of edges is $m = \Theta(rq)$. The pruning subroutine ensures that the minimum pairwise distance among sinks in each R_i is at least $b \geq \log k / \log\log k$, and the number of cut edges is $f = N$.

Applying Lemma 2.1, the coding gap is at least

$$\frac{a \cdot r}{f + 2m/b} = \Omega\left(\min\left\{\frac{r}{f}, \frac{r}{2m/b}\right\}\right).$$

Substituting the estimates for r, m, and b gives:

$$\Omega\left(\min\left\{\frac{\delta \log k}{q}, \frac{\log k}{q \log\log k}\right\}\right),$$

as claimed. □

2.2 Boosting the Distance via a Binary Tree Gadget

In this section, we introduce a binary tree gadget to eliminate the low-order term in the lower bound on the coding gap. In the previous construction, the distance between sinks in each R_i was ensured through a sampling-based approach, which introduced an extra $\log\log k$ factor. Here, we replace high-degree vertices with structured binary trees to achieve the better distance guarantees without relying on sampling. The pseudocode is provided in Algorithm 2.

Binary Tree Gadget This construction can be viewed as a modification of the previous bipartite structure. Denote by $G[B, C]$ the bipartite graph induced by the vertex set $B \cup C$. For each edge

(u, v) in $G[B, C]$, we first introduce a new intermediate vertex t and replace the edge with two edges (u, t) and (v, t).

Next, for each vertex $u \in B \cup C$ with degree d in the graph, we remove all edges incident to u and replace them with a complete binary tree rooted at u with d leaves. The original neighbors of u are then assigned to the leaves via a uniformly random permutation—each leaf is connected to a unique neighbor of u according to this random ordering.

After this transformation, every intermediate vertex t (originally inserted to split edges) has degree exactly 2, so we remove t and directly connect its two neighbors with an edge. Although the construction in Algorithm 2 presents this process in a more direct manner, it is equivalent to the description above.

Notably, the sampling subroutine from the previous section is no longer required in this construction.

Algorithm 2 Generating Gap Instance via LDCs and Binary Tree Gadget

Set $A \leftarrow \{S\}$ and $s_i \leftarrow S$ for all $i \in [k]$.
Set $B \leftarrow \{w_i\}_{i=1}^{N}$.
Set $F \leftarrow \{(S, w_i)\}_{i=1}^{N}$ and initialize $E \leftarrow F$.
Initialize $R_i \leftarrow \emptyset$ for all $i \in [k]$.
for each vertex $u \in B$ **do** ▷ Insert binary tree for each B-vertex
    Insert a complete binary tree T_u with k leaves and root at vertex u.
    Label the leaves of T_u with a random permutation of $\{1, 2, \ldots, k\}$.
end for
for each $i = 1$ to k **do** ▷ Add sinks via LDC decoding
    Initialize $U \leftarrow \emptyset$.
    while $|U| \leq \delta N$ **do**
        Let $D \leftarrow \textsc{Decode}(i, U)$.
        Add a new vertex v to R_i.
        Insert a complete binary tree T_v with q leaves and root at vertex v.
        Let $p : D \to [q]$ be an arbitrary bijection (permutation of D).
        for each $j \in D$ **do**
            Add an edge from the leaf of T_{w_j} labeled i to the $p(j)$-th leaf of T_v.
        end for
        Update $U \leftarrow U \cup D$.
    end while
end for
for each $i \in [k]$ and **each** $u \in R_i$ **do** ▷ Prune close sinks
    if there exists $v \in R_i \setminus \{u\}$ such that $\mathrm{dist}_{G \setminus F}(u, v) \leq \frac{1}{4} \log k$ **then**
        Remove v from R_i.
    end if
end for

Pruning Subroutine Similar to the previous section, the pruning threshold is chosen to ensure that only a small fraction of sinks are removed. The key advantage of the current construction is

that the minimum required distance between sinks can now be set to a logarithmic function of k.

Intuitively, the analysis relies on the structure of the binary tree gadget. Starting from any vertex in a tree and growing a ball of increasing radius, the next labeled vertex encountered is essentially random due to the uniform permutation of labels. Thus, to encounter a vertex with the same label (i.e., corresponding to the same message index i), the ball must typically grow to logarithmic radius. This ensures that sinks associated with the same message are, with high probability, far apart in $G \setminus F$.

Lemma 2.5. *In the pruning subroutine of Algorithm 2, at most an $O(k^{-1/2})$ fraction of the sinks are removed with high probability.*

The correctness of Lemma 2.5 follows from the claim below.

Claim 2.6. *Before the pruning subroutine, for each $i \in [k]$ and each $u \in R_i$, with probability at least $1 - k^{-3/4}$,*

$$\min_{v \in R_i \setminus \{u\}} \mathrm{dist}_{G \setminus F}(u, v) \geq \frac{1}{4} \log k.$$

Proof. The only randomness in the algorithm arises from labeling the leaves of each binary tree using independent random permutations. For the sake of analysis, we may equivalently assume that all sinks are added first, and then the leaf labels of every binary tree are assigned uniformly at random.

Let D be the indices we used to create sink u. Consider a path from u to some $v \in R_i \setminus \{u\}$. The path should cross some binary tree T_{w_j} for some $j \notin D$. This motivates us to define the following set

$$P = \{\, v \in G \setminus S : \mathrm{dist}_{G \setminus F}(u, v) < \tfrac{1}{4} \log k \,\} \setminus \big(\cup_{j \in D} V(T_{w_j}) \cup V(T_u) \big).$$

Since the degree of each vertex is at most 3 in $G \setminus F$ and the degree of u is 2, we have $s := |P| < k^{\frac{1}{4}}$.

Let $C = \bigcup_{i \in [k]} R_i$. Let $B' = \bigcup_{w \in B} V(T_w)$ and $C' = \bigcup_{w \in C} V(T_w)$. For each vertex $v \in P$, we say that v is bad if either $v \in B'$ is labeled by i, or $v \in C'$ lies in T_w for some $w \in R_i$. If there is no bad vertex in P, by the definition of P and the discussion above, the inequality in the claim holds.

For the vertices in P, we order them by their distance to the vertex u in $G \setminus F$, i.e. $P = \{v_1, v_2, \ldots, v_s\}$ such that $\mathrm{dist}_{G \setminus F}(u, v_\ell) \leq \mathrm{dist}_{G \setminus F}(u, v_{\ell+1})$, then we shall prove that

$$\Pr\big[v_\ell \text{ is bad} \mid v_j \text{ is good } \forall j < \ell\big] \leq \frac{1}{k - \ell} \tag{1}$$

which will imply that,

$$\Pr\big[\forall v \in P, v \text{ is good}\big] \geq \prod_{\ell=1}^{s} \Pr\big[v_\ell \text{ is good} \mid v_j \text{ is good } \forall j < \ell\big] \geq \prod_{\ell=1}^{s} \frac{k - \ell - 1}{k - \ell} \geq 1 - k^{-\frac{3}{4}}.$$

This will conclude the claim as we discussed in the beginning. The rest is proving Equation (1). Let $d = \mathrm{dist}_{G \setminus F}(u, v_\ell)$. There are two cases.

1. There exists a vertex $v^* \in C'$ adjacent to v_ℓ, and $\mathrm{dist}_{G \setminus F}(v^*, u) < d$. Let w be the sink vertex

such that $v^* \in T_w$. By assumption, $w \notin R_i$ as v^* is good, thereby v_ℓ is not labeled by i, nor in another binary tree $T_{w'}$, which means v_ℓ is good.

2. Otherwise, there exists a vertex $v^* \in B'$ adjacent to v_ℓ, and $\mathrm{dist}_{G\setminus F}(v^*, u) < d$. We only need to consider the case when v_ℓ is a leaf of the binary tree, in which v_ℓ has a label. Let T_{w_j} be the tree contains v^*. If $j \in D$ then the label of v^* must be different from i, since otherwise v_ℓ should be in T_u which contradicts the definition of P. Now $j \notin D$. Among the first ℓ vertices in P, there are at least $k - \ell$ unseen labels, including i. Since the label of v_ℓ is assigned uniformly among these remaining labels as in a random permutation, the probability that v_ℓ receives label i is at most $\frac{1}{k-\ell}$. □

Proof of Lemma 2.5. By Claim 2.6, $k^{-\frac{3}{4}}$ is an upper bound of the probability that each sink u be pruned. Therefore, the expected fraction of the set pruned is $k^{-\frac{3}{4}}$. Then we conclude with Markov's inequality. □

We name this instance $I^* = (G, \mathcal{S}, F)$. Applying Lemma 2.1 to our gap instance I^*, we prove our main lemma.

Lemma 2.7 (Restatement of Lemma 1.5)**.** *Suppose there exists a linear $(q, \delta, q\delta)$-locally decodable code $\mathcal{C} : \mathbf{F}^k \to \mathbf{F}^N$ with perfect smoothness. Then the network coding gap for the instance I^* is at least $\Omega\left(\frac{\delta \log k}{q}\right)$, where the graph has size $n = \Theta(Nk)$.*

Proof. The proof follows the same outline as that of Lemma 2.4. For the network coding throughput, we assume each source message is uniformly distributed over $\mathbf{F}$. Each edge in the network carries a symbol from $\mathbf{F}$, and each sink recovers its target symbol using q symbols from the second layer. Specifically, each vertex in B receives one coordinate of the codeword generated by the locally decodable code, and each sink in the third layer wants to query q such coordinates.

The binary tree T_v rooted at each $v \in B$ propagates the value of the corresponding codeword coordinate toward its leaves. For each sink r, the binary tree T_r aggregates the q codeword symbols received from the second layer and computes the linear decoding function as the data flows upward toward the root. In this way, the network performs the entire decoding process within its structure.

The scale defined in the network coding solution is set by $c^* = \log s$, which is the same as in Lemma 2.4. Under this scaling, the network supports a coding solution that delivers the full message to each sink, and therefore the coding throughput of I_S is at least $a = 1$.

We now estimate the parameters in Lemma 2.1. For each $i \in [N]$, the algorithm constructs a binary tree of size $\Theta(k)$ rooted at $c_i \in B$, contributing $\Theta(Nk)$ vertices. Additionally, each sink is augmented with a binary tree of size $\Theta(q)$. The total number of sinks is $\Theta(\delta Nk/q)$, so the overall graph size is $n = \Theta(Nk)$ and the number of edges is $m = \Theta(n)$, as all vertices have constant degree except for the source S.

The number of sinks that remain after pruning is $r = \Theta(\delta Nk/q)$ by Lemma 2.5. The distance parameter is $b > \frac{1}{4} \log k$, as ensured by the pruning subroutine, and the number of cut edges is $f = N$.

Applying Lemma 2.1, the coding gap is at least:

$$\frac{r}{f+2m/b} = \Omega\left(\min\left\{\frac{r}{f}, \frac{rb}{2m}\right\}\right) = \Omega\left(\min\left\{\frac{\delta k}{q}, \frac{\delta \log k}{q}\right\}\right) = \Omega\left(\frac{\delta \log k}{q}\right).$$

□

3 Upper Bound Network Coding Gap

Given a graph $G = (V, E)$ with capacities c_e for each edge $e \in E$. It will be convenient to define $c_{uv} = c_e$ for all edge $e = (u, v) \in E$ and $c_{uv} = 0$ otherwise. For any set of vertex $U \subseteq V$, we define the cut capacity $C(U, \overline{U}) = \sum_{u \in U, v \in \overline{U}} c_{uv}$. Recall that $S_i = R_i \cup \{s_i\}$ is the set of terminals including source and all sinks for demand i. Then, for any set of vertex U, we define the demand crossing U to be

$$D(U, \bar{U}) := \sum_{\substack{1 \le j \le k \\ 0 < |S_j \cap \bar{U}| < |S_j|}} d_i$$

Define $\Psi = \min_U \frac{C(U,\overline{U})}{D(U,\overline{U})}$ which generalizes the definition of the sparsest cut. It is straightforward to verify that the value of the integral solution to the dual LP in Section B is at most Ψ, since the dual objective achieves Ψ under the assignment $y_e = z_i = 1$ for all edges e and demands i crossing U, and $y_e = z_i = 0$ otherwise. Furthermore, we show that Ψ is an upper bound on the network coding throughput.

Lemma 3.1. *The multi-source multicast network coding throughput is at most* Ψ.

Proof. Let U be the vertex set such that $\Psi = \min_U \frac{C(U,\overline{U})}{D(U,\overline{U})}$. Suppose there are ℓ edges crossing U and $\overline{U}$, denoted by $e_1, \ldots, e_\ell$. Recall the definition in Section A, each edge is splitted into two directed edge and $\Gamma(\vec{e})$ represent the alphabet on edge $\vec{e}$. Let $X_{\vec{e}} \in \Gamma(\vec{e})$ be the symbol transmitted on edge $\vec{e}$. Let

$$X_l = (X_{\overrightarrow{e_1}}, X_{\overrightarrow{e_2}}, \ldots, X_{\overrightarrow{e_\ell}})$$

be all the symbols transmitted on the edges from U to $\overline{U}$, and

$$X_r = (X_{\overleftarrow{e_1}}, X_{\overleftarrow{e_2}}, \ldots, X_{\overleftarrow{e_\ell}})$$

Suppose there are p terminal sets $S_{j_1}, \ldots, S_{j_p}$ that cross U with source node in U, i.e. $s_{j_i} \in U$ and $R_{j_i} \cap \overline{U} \neq \emptyset$, and q terminal sets $S_{j'_1}, \ldots, S_{j'_q}$ that cross U with source node in $\overline{U}$, i.e. $s_{j'_i} \in \overline{U}$ and $R_{j'_i} \cap U \neq \emptyset$. Let $m_j \in M_j$ be the message of the j-th source. Let

$$Y_l = (m_{j_1}, m_{j_2}, \ldots, m_{j_p})$$

and

$$Y_r = (m_{j'_1}, m_{j'_2}, \ldots, m_{j'_q})$$

such that Y_l and Y_r together contains all the messages relevant to this cut $(U, \overline{U})$, and we define Z

to be all the other messages. We want to show that

$$H(X_l) \geq H(Y_l) \quad \text{and} \quad H(X_r) \geq H(Y_r),$$

then the lemma follows by definition of network coding throughput. To see this, suppose by contradiction that the throughput exceeds Ψ: there exists scale constant c^* such that $H(Y_l \cup Y_r) > D(U, \overline{U}) \cdot \Psi \cdot c^*$, we have $H(X_l \cup X_r) > D(U, \overline{U}) \cdot \Psi \cdot c^* = C(U, \overline{U}) \cdot c^*$, which implies that there exists an edge e such that $H(e) > c_e \cdot c^*$, contradicts to the capacity constraint.

We claim that the conditional entropy $H(Y_l \mid X_l, Y_r, Z) = 0$ and $H(Y_r \mid X_r, Y_l, Z) = 0$. Without loss of generality we prove the first one, we need to show that each message m_{j_i} is determined by X_l, Y_r and Z, this is indeed as in the time-expanded graph which is acyclic, the output can be computed once the input is fixed, and once we truncate all the vertices and edges in U which are prior to X_l, every symbol on all the edges could still be computed given X_l, Y_r and Z. By assumption, there exists $r \in R_{j_i}$ such that $r \in \overline{U}$, hence m_{j_i} can be computed for all i, which implies our claim.

Furthermore, Y_l, Y_r and Z are mutually independent. Therefore by the claim we derived and applying chain rule, $I(Y_l; X_l \mid Y_r, Z) = H(Y_l \mid Y_r, Z) - H(Y_l \mid X_l, Y_r, Z) = H(Y_l) - 0 = H(Y_l)$. On the other hand, $I(Y_l; X_l \mid Y_r, Z) \leq H(X_l \mid Y_r, Z) \leq H(X_l)$. Combining together, we have $H(X_l) \geq H(Y_l)$, and same holds for $H(X_r) \geq H(Y_r)$. □

Therefore, the proof of Theorem 1.2 is complete once we have the following lemma, which gives an upper bound on the LP integrality gap. The rest of this section is to prove Lemma 3.2.

Lemma 3.2. *The ratio of the generalized sparsity Ψ over the multi Steiner tree number is at most $O(\log n)$.*

The proof builds upon the results from oblivious routing [Räc08] using the *tree-based flow-sparsifiers*. We first introduce the preliminaries. Then we use the tree-based flow-sparsifiers to lower bound the Steiner tree packing number by a $O(\log n)$-factor of relaxation of the generalized sparsity Ψ.

The generalized tree packing of a graph G consists of a collection of trees T_i and distribution λ_i (i.e. $\sum_i \lambda_i = 1$) such that each tree spans the vertex set V but not necessarily a subgraph of G, along with an embedding of the tree T_i in the graph, i.e. a mapping P_i that maps each edge $(x, y) \in T_i$ to a path in G, which denoted by $P_i(x, y)$. For each $(x, y) \in T_i$, removing (x, y) separates T_i into two parts $(U, V \setminus U)$. We define $C_i(x, y) := C(U, \overline{U})$ as the size of the cut, and define $D_i(x, y) := D(U, \overline{U})$ which captures the weighted demand that crosses the edge (x, y). Now we are ready to define the proposition that is required for the tree packing.

Definition 3.3 (Cut-Tree Packing)**.** It is called an *α-approximate cut-tree packing*, if a collection of trees T_i with a distribution λ_i satisfies the following inequality for every edge $(u, v) \in E$,

$$c_{uv} \geq \frac{1}{\alpha} \sum_i \lambda_i \sum_{(x,y) \in T_i : (u,v) \in P_i(x,y)} C_i(x, y) \ .$$

i.e. the capacity of (u, v) is at least $1/\alpha$ fraction of the sum of weighted cuts associated with the paths using (u, v).

Given the terminal set $S_j := \{s_j\} \cup R_j$, we want to define a multi Steiner tree packing solution based on each tree T_i we constructed for the cut-tree packing. For each tree T_i, there exists a minimal subtree of T_i that connects S_j. The embedding P_i of this subtree corresponds to a graph G_{ij} which is a subgraph of G. After eliminating isolated vertices, G_{ij} is a connected subgraph of G. Let T_{ij} be an arbitrary spanning tree of G_{ij}. We know that T_{ij} spans S_j. We add T_{ij} to the multi Steiner tree packing with weight λ_i.

This gives a multi Steiner tree packing solution that serves one unit demand, as the summation of λ_i equals one. Denote by $\text{cong}_{uv} \in \mathbb{R}_{\geq 0}$ the *congestion* of edge (u, v), which is the total amount of capacity of edge (u, v) used by the packing solution over the capacity of edge (u, v). The congestion ϕ of the packing solution is the maximum congestion over all edges. The multi Steiner tree number is then $\tau = \frac{1}{\phi}$ as it is the maximum fraction demand that could be satisfied while preserving the capacity constraint. For our multi Steiner tree packing solution, the congestion on edge u, v is

$$\text{cong}_{uv} \leq \frac{\sum_i \lambda_i \sum_{(x,y) \in T_i : (u,v) \in P_i(x,y)} D_i(x, y)}{c_{uv}} .$$

Suppose we use α-approximate cut-tree packing for the solution, we can upper bound the congestion,

$$\text{cong}_{uv} \leq \alpha \cdot \frac{\sum_i \lambda_i \sum_{(x,y) \in T_i : (u,v) \in P_i(x,y)} D_i(x, y)}{\sum_i \lambda_i \sum_{(x,y) \in T_i : (u,v) \in P_i(x,y)} C_i(x, y)} \leq \alpha \max_{i,x,y} \frac{D_i(x, y)}{C_i(x, y)} \leq \alpha / \Psi .$$

Therefore, the multi Steiner tree number $\tau = \frac{1}{\phi} \geq \frac{1}{\alpha} \cdot \Psi$. Lemma 3.2 is thereby a consequence of the following lemma. □

Lemma 3.4 ([Räc08, WS11])**.** *There exists an $O(\log n)$-approximate cut-tree packing.*

As discussed earlier, combining Lemma 3.1 and Lemma 3.2 yields Theorem 1.2. □

4 Locally Decodable Codes in Network Coding

In this section, we briefly discuss the limitations encountered when applying existing LDCs to network coding. We then introduce a novel variant of LDCs with a relaxed fault-tolerence property, specifically tailored to construct network coding gaps, so that we believe this variant is easier to construct than standard LDCs while still applicable to network coding.

4.1 Limitations of Existing LDC Constructions

LDCs serve as the primary technical tool in this work and have been extensively studied for decades. Despite significant research, the fundamental trade-off between rate and query complexity remains a major open problem. For constant query complexity $q \geq 3$, the best known lower bound on the codeword length is $\Omega\left(k^{1+2/(q-2)}\right)$ [BHKL24]. In contrast, known upper bounds do not even reach the quasi-polynomial regime when $q = O((\log k)^c)$ for constant $c < 1$. Establishing an upper

bound in this regime would directly imply network coding gaps of $(\log n)^{\Omega(1)}$ via our main lemma, highlighting the particular interest of the logarithmic-query regime.

The best known constructions of LDCs in the polylogarithmic-query regime are Matching Vector (MV) codes and Reed–Muller (RM) codes; their parameters are summarized in Table 1, based on [Yek12]. These codes satisfy perfect smoothness and tolerate a constant fraction of errors; that is, both δ and ϵ are constants. From the table, the rate of Reed–Muller (RM) codes lies in the quasipolynomial regime; however, this is achieved only when the query complexity is super-logarithmic, which is not the desired query regime. Both MV and RM codes have been studied for decades and improving them in either rate or query regime could be very challenging.

Code Type	q	N
MV code	$O(\log k)$	$\exp\left(\exp\left((\log k)^{O\left(\frac{\log\log q}{\log q}\right)}(\log\log k)^{1-\Omega\left(\frac{\log\log q}{\log q}\right)}\log q\right)\right)$
RM or MV codes	$O(\log k \log\log k)$	$k^{O(\log\log k)}$
RM code	$(\log k)^t,\ t>1$	$k^{1+\frac{1}{t-1}+o(1)}$

Table 1: Comparison of parameters for known LDC constructions.

4.2 Robust Distance LDC

In this section, we propose a novel variant of LDCs that is tailored for applications in network coding gaps. Recall from Section 2.1 that the algorithm samples each sink with probability $(\log k)/k$, and consequently discards it with high probability. As a result, the full fault tolerance offered by traditional LDCs is not fully leveraged. This observation motivates us to define a new variant with relaxed fault tolerance requirements, which we believe is easier to construct than standard LDCs.

We begin with some preliminaries. A *q-uniform hypergraph* $G = (V, E)$ is a hypergraph in which every edge $e \in E$ has rank $|e| = q$. A *hypergraph matching* in G is a subset of edges $H' \subseteq E$ such that no two edges overlap; that is, for all $e, e' \in H'$ with $e \neq e'$, we have $e \cap e' = \emptyset$. We introduce the notion of *matching decodability*, a relaxation of local decodability.

Definition 4.1. A code $\mathcal{C} : \mathbf{F}^k \to \mathbf{F}^N$ is called (q, δ)-*matching decodable* if there exist q-uniform hypergraph matchings $H_1, \ldots, H_k$ on vertices indexed by $[N]$, each containing at least δN hyperedges, such that for every $D \in H_i$, there exists a decoding function $f_{i,D} : \mathbf{F}^q \to \mathbf{F}$ satisfying

$$\forall x \in \mathbf{F}^k, \quad x_i = f_{i,D}(\mathcal{C}(x)_v \mid v \in D).$$

i.e., the i-th bit of the message can be decoded using the codewords whose indices lie in D.

This is a variant of the notion known as *normally decodable*, which has been used in proving LDC lower bounds [Yek12].

Let $H = \bigcup_{i=1}^{k} H_i$. For any two hyperedges $e, e' \in H$, define the distance $\mathrm{dist}_H(e, e')$ as the length of the shortest path $e = e_1, e_2, \ldots, e_\ell = e'$ in H connecting e to e', where consecutive hyperedges in

the path must intersect (i.e., $\forall\, 1 \le i < \ell,\ e_i \cap e_{i+1} \neq \emptyset$).

Definition 4.2 (Robust Distance LDC)**.** A (q, δ)-matching decodable code is said to be (q, δ, d)-*robust distance locally decodable* if for each i, $|H_i| \ge \delta N$, and the following distance condition holds:

$$\forall\, 1 \le i \le k,\ \forall\, e, e' \in H_i,\ e \neq e', \quad \mathrm{dist}_H(e, e') \ge d.$$

The advantage of this relaxed variant is highlighted by Proposition 4.3: the required matching size δN can be significantly smaller—specifically, $\delta = \Omega((\log k)/k)$ suffices for our purposes, compared to the stricter requirement $\delta = \omega(1/\log k)$ for traditional LDCs to be useful for our network coding instances. On the other hand, we require the distance d to be at least $\Omega((\log k)^\epsilon)$ for some constant $\epsilon > 0$, to achieve a polylogarithmic coding gap. It is worth noting that, due to purely combinatorial limitations, d can be at most $O(\log k/\log(\delta k))$. Thus, the code has to be sparse to obtain a large distance d. In particular, we show in Proposition 4.4 that standard LDCs can be transformed into robust distance LDCs using the same sampling technique introduced earlier. This demonstrates that robust distance LDCs are a relaxation of standard LDCs.

Proposition 4.3. *If a (q, δ, d)-robust distance locally decodable code $\mathcal{C} : \mathbf{F}^k \to \mathbf{F}^N$ exists, then the multi-source multicast network coding gap is at least $\Omega(\min\{\delta k, d/q\})$ for a graph of size $O(\delta N k)$.*

The proofs for both propositions are deferred to Section C, as it closely follows the argument in Section 2.1.

Proposition 4.4. *Suppose there exists $(q, \delta, q\delta)$-locally decodable code with perfect smoothness, then a $\left(q, \Theta\left(\frac{\delta \log k}{qk}\right), \frac{\log k}{2(\log q + \log\log k)}\right)$-robust distance locally decodable code also exists.*

References

[AC04] Amit Agarwal and Moses Charikar. On the advantage of network coding for improving network throughput. In *Information Theory Workshop*, pages 247–249. IEEE, 2004. 1

[ACLY00] Rudolf Ahlswede, Ning Cai, S-YR Li, and Raymond W Yeung. Network information flow. *IEEE Transactions on information theory*, 46(4):1204–1216, 2000. 1, 2, 23

[AFKL19] Peyman Afshani, Casper Benjamin Freksen, Lior Kamma, and Kasper Green Larsen. Lower bounds for multiplication via network coding. *arXiv preprint arXiv:1902.10935*, 2019. 1, 2

[AHJ+06] Micah Adler, Nicholas JA Harvey, Kamal Jain, Robert D Kleinberg, and April Rasala Lehman. On the capacity of information networks. In *Soda*, volume 6, pages 241–250, 2006. 1

[AR98] Yonatan Aumann and Yuval Rabani. An o(log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998. 2

[BDL13] Abhishek Bhowmick, Zeev Dvir, and Shachar Lovett. New bounds for matching vector families. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 823–832, 2013. 2

[BGS17] Mark Braverman, Sumegha Garg, and Ariel Schvartzman. Coding in undirected graphs is either very helpful or not helpful at all. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, 2017. 1, 3

[BHKL24] Arpon Basu, Jun-Ting Hsieh, Pravesh K Kothari, and Andrew D Lin. Improved lower bounds for all odd-query locally decodable codes. *arXiv preprint arXiv:2411.14361*, 2024. 17

[BKL11] Anna Blasiak, Robert Kleinberg, and Eyal Lubetzky. Lexicographic products and the power of non-linear network coding. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 609–618. IEEE, 2011. 3

[BLMR98] John W Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of the ACM SIGCOMM*, pages 56–67, 1998. 3

[BTV17] Alexander Barg, Itzhak Tamo, and Serge Vlăduţ. Locally recoverable codes on algebraic curves. *IEEE Transactions on Information Theory*, 63(8):4928–4939, 2017. 3

[CCMG18] Alejandro Cohen, Asaf Cohen, Muriel Medard, and Omer Gurewitz. Secure multi-source multicast. *IEEE Transactions on Communications*, 67(1):708–723, 2018. 2

[CM15] Viveck R Cadambe and Arya Mazumdar. Bounds on the size of locally recoverable codes. *IEEE transactions on information theory*, 61(11):5787–5794, 2015. 3

[CMST21] Han Cai, Ying Miao, Moshe Schwartz, and Xiaohu Tang. A construction of maximally recoverable codes with order-optimal field size. *IEEE Transactions on Information Theory*, 68(1):204–212, 2021. 3

[Cov99] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999. 2

[DGY11] Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM Journal on Computing*, 40(4):1154–1178, 2011. 2, 6, 27

[DH13] Zeev Dvir and Guangda Hu. Matching-vector families and ldcs over large modulo. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 513–526. Springer, 2013. 2

[Efr09] Klim Efremenko. 3-query locally decodable codes of subexponential length. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 39–44, 2009. 2, 5, 26

[FHLS19] Alireza Farhadi, MohammadTaghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi. Lower bounds for external memory integer sorting via network coding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 997–1008, 2019. 1

[FS+07] Christina Fragouli, Emina Soljanin, et al. Network coding fundamentals. *Foundations and Trends® in Networking*, 2(1):1–133, 2007. 1

[GXY19] Venkatesan Guruswami, Chaoping Xing, and Chen Yuan. How long can optimal locally repairable codes be? *IEEE Transactions on Information Theory*, 65(6):3662–3670, 2019. 3

[HKL04] Nicholas J Harvey, Robert D Kleinberg, and April Rasala Lehman. Comparing network coding with multicommodity flow for the k-pairs communication problem. 2004. 1

[HKL06] Nicholas JA Harvey, Robert Kleinberg, and April Rasala Lehman. On the capacity of information networks. *IEEE Transactions on Information Theory*, 52(6):2345–2364, 2006. 1, 2, 23

[HL08] Tracey Ho and Desmond Lun. *Network Coding: An Introduction*. Cambridge University Press, Cambridge, UK, 2008. 1

[HMK+06] Tracey Ho, Muriel Médard, Ralf Koetter, David R Karger, Michelle Effros, Jun Shi, and Ben Leong. A random linear network coding approach to multicast. *IEEE Transactions on information theory*, 52(10):4413–4430, 2006. 1

[HPY22] Yael Hitron, Merav Parter, and Eylon Yogev. Broadcast congest algorithms against eavesdroppers. In *36th International Symposium on Distributed Computing, DISC 2022*, page 27. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2022. 3

[HSX+12] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. Erasure coding in windows azure storage. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 15–26, 2012. 3

[HWZ20] Bernhard Haeupler, David Wajc, and Goran Zuzic. Network coding gaps for completion times of multiple unicasts. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 494–505. IEEE, 2020. 1, 3

[JSC+05] Sidharth Jaggi, Peter Sanders, Philip A Chou, Michelle Effros, Sebastian Egner, Kamal Jain, and Ludo MGM Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE transactions on information theory*, 51(6):1973–1982, 2005. 1

[JVY06] Kamal Jain, Vijay V Vazirani, and Gideon Yuval. On the capacity of multiple unicast sessions in undirected graphs. *IEEE Transactions on Information Theory*, 52(6):2805–2809, 2006. 1

[KM03] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *IEEE/ACM transactions on networking*, 11(5):782–795, 2003. 2

[KS06] Gerhard Kramer and Serap A Savari. Edge-cut bounds on network coding rates. *Journal of Network and Systems Management*, 14:49–67, 2006. 1

[KSCM23] Saurabh Kadekodi, Shashwat Silas, David Clausen, and Arif Merchant. Practical design considerations for wide locally recoverable codes (lrcs). *ACM Transactions on Storage*, 19(4):1–26, 2023. 3

[LKEGC11] Sung Hoon Lim, Young-Han Kim, Abbas El Gamal, and Sae-Young Chung. Noisy network coding. *IEEE Transactions on Information Theory*, 57(5):3132–3152, 2011. 2

[LL04a] Zongpeng Li and Baochun Li. Network coding in undirected networks. CISS, 2004. 1

[LL04b] Zongpeng Li and Baochun Li. Network coding: The case of multiple unicast sessions. In *Allerton Conference on Communications*, volume 16. Citeseer, 2004. 1

[LLR95] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995. 2

[LM09] Michael Langberg and Muriel Médard. On the multiple unicast network coding, conjecture. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 222–227. IEEE, 2009. 1, 2

[Lov14] Shachar Lovett. Linear codes cannot approximate the network capacity within any constant factor. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, page 141, 2014. 3

[Lub02] Michael Luby. Lt codes. *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 271–280, 2002. 3

[LYC03] S-YR Li, Raymond W Yeung, and Ning Cai. Linear network coding. *IEEE transactions on information theory*, 49(2):371–381, 2003. 1

[Mic19] Giacomo Micheli. Constructions of locally recoverable codes which are optimal. *IEEE transactions on information theory*, 66(1):167–175, 2019. 3

[MPK19] Umberto Martínez-Peñas and Frank R Kschischang. Universal and dynamic locally repairable codes with maximal recoverability via sum-rank codes. *IEEE Transactions on Information Theory*, 65(12):7790–7805, 2019. 3

[OS81] Haruko Okamura and Paul D Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75–81, 1981. 1

[Par23] Merav Parter. Secure computation meets distributed universal optimality. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2336–2368. IEEE, 2023. 3

[PD14] Dimitris S Papailiopoulos and Alexandros G Dimakis. Locally repairable codes. *IEEE Transactions on Information Theory*, 60(10):5843–5855, 2014. 3

[Räc08] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 255–264, 2008. 2, 6, 16, 17

[Sho06] Amin Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006. 3

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009. 28

[TB14] Itzhak Tamo and Alexander Barg. A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory*, 60(8):4661–4676, 2014. 3

[TGC16] Satyajit Thakor, Alex Grant, and Terence Chan. Cut-set bounds on network information flow. *IEEE Transactions on Information Theory*, 62(4):1850–1865, 2016. 2

[Tre04] Luca Trevisan. Some applications of coding theory in computational complexity. *arXiv preprint cs/0409044*, 2004. 5

[WS11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011. 17

[XLWH14] Tang Xiahou, Zongpeng Li, Chuan Wu, and Jiaqing Huang. A geometric perspective to multiple-unicast network coding. *IEEE Transactions on Information Theory*, 60(5):2884–2895, 2014. 1

[Yek12] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends® in Theoretical Computer Science*, 6(3):139–255, 2012. 2, 6, 18

[YLCZ06] R. W. Yeung, S.-Y. Li, N. Cai, and Z. Zhang. *Network Coding Theory*. Foundations and Trends® in Communications and Information Theory (Now Publishers). Now Publishers, Boston, MA, USA, 2006. Monograph. 1

[YLLW17] Xunrui Yin, Zongpeng Li, Yaduo Liu, and Xin Wang. A reduction approach to the multiple-unicast conjecture in network coding. *IEEE Transactions on Information Theory*, 64(6):4530–4539, 2017. 1

A Definition of Network Coding via Time-Expanded Graph

In this section, we define the general network coding solution via the time-expanded graph construction, following prior work [ACLY00, HKL06]. For each undirected edge $e \in E$, we treat it as a pair of directed edges, $\vec{e}$ and $\overleftarrow{e}$ (i.e., one in each direction). We will define their respective capacities shortly, ensuring that their sum equals the original capacity of e. From this point forward, we treat all edges as directed.

Definition A.1 (Time-Expanded Graph G^*)**.** Given a directed graph $G = (V, E)$, the *time-expanded graph* $G^* = (V^*, E^*)$ is a directed acyclic graph constructed as follows:

- The vertex set is $V^* = V \times \mathbb{Z}$.
- For each edge $e = (u, v) \in E$ and each time step $t \in \mathbb{Z}$, add a directed edge $e_t = (u^{(t-1)}, v^{(t)})$ to E^*.
- For each vertex $v \in V$ and each $t \in \mathbb{Z}$, add a *memory edge* $(v^{(t-1)}, v^{(t)})$ to E^*.

We also model sources and sinks as special edges. For each source s_i, we introduce two auxiliary vertices σ_i and σ_i', and add edges (σ_i, σ_i') and $(\sigma_i', s_i^{(t)})$ for all $t \in \mathbb{Z}$, each with infinite capacity. Similarly, for each sink $r \in R_i$, we add two auxiliary vertices $\gamma_{r,i}$ and $\gamma_{r,i}'$, and edges $(\gamma_{r,i}, \gamma_{r,i}')$ and $(r^{(t)}, \gamma_{r,i})$ for all $t \in \mathbb{Z}$, also with infinite capacity.

The alphabet on the edges (σ_i, σ_i') and $(\gamma_{r,i}, \gamma_{r,i}')$ is M_i. We say that the network coding solution satisfies *correctness* if, for each sink $r \in R_i$, the symbol transmitted on $(\gamma_{r,i}, \gamma_{r,i}')$, denoted $X_{(\gamma_{r,i},\gamma_{r,i}')}$, is equal to the message m_i sent by the corresponding source, i.e., $X_{(\gamma_{r,i},\gamma_{r,i}')} = X_{(\sigma_i,\sigma_i')} = m_i$.

We now define the capacity constraints in the time-expanded graph. All memory edges have infinite capacity. For each directed edge $\vec{e}$, let the overall alphabet be $\Gamma(\vec{e}) = \prod_{t\in\mathbb{Z}} \Gamma(\vec{e}^{(t)})$, and let $X_{\vec{e}} \in \Gamma(\vec{e})$ denote the collection of symbols transmitted along all time steps of $\vec{e}$. The network coding solution respects the capacity constraint if: $H(\vec{e}) + H(\overleftarrow{e}) \le c_e$, where by $H(\vec{e})$ we denote the entropy of the symbol $X_{\vec{e}}$ sended along $\vec{e}$.

B Linear Programs for Multi Steiner Tree Packing

The primal LP is as follows. Recall that for each demand i, the terminal set is $S_i = \{s_i\} \cup R_i$. Each tree T in the Steiner tree packing for the i-th demand must span all nodes in S_i. The combined packing across all demands must respect the edge capacity constraints.

$$
\begin{aligned}
\text{maximize} \quad & \tau \\
\text{subject to} \quad & \sum_{T: S_i \subseteq V(T)} x_{i,T} \ge \tau \cdot d_i && \forall\, 1 \le i \le k \\
& \sum_{\substack{i,T \\ e \in T}} x_{i,T} \le c_e && \forall\, e \in E \\
& x_{i,T} \ge 0 && \forall\, i, T
\end{aligned}
$$

The dual of this linear program is as follows:

$$
\begin{aligned}
\text{minimize} \quad & \frac{\sum_{e\in E} c_e y_e}{\sum_i d_i z_i} \\
\text{subject to} \quad & \sum_{e \in T} y_e \ge z_i && \forall\, i, T, \text{s.t. } S_i \subseteq V(T) \\
& z_i \ge 0 && \forall\, 1 \le i \le k \\
& y_e \ge 0 && \forall\, e \in E
\end{aligned}
$$

In Section 3, we mentioned that the generalized sparsity Ψ defined there serves as an upper bound on the integer program of the dual under the integral constraints $z_i, y_e \in \{0, 1\}$ for all $i \in [k]$ and $e \in E$. In fact, one can verify that these two quantities differ by at most a constant factor of two, although this observation is not used in our proof.

C Missing Proofs of Section 4.2

C.1 Proof of Proposition 4.3

The gap instance $(G, \mathcal{S}, F)$ is generated via the Algorithm 3, where $G = (A, B, C, E)$ is a tripartite graph. All demands and edge capacities are units.

Algorithm 3 Gap Instance via Robust Distance LDCs

1: Set $A \leftarrow \{S\}$ and $s_i \leftarrow S$ for all $i \in [k]$.
2: Set $B \leftarrow \{w_i\}_{i=1}^{N}$.
3: Set $F \leftarrow \{(S, w_i) \mid i \in [N]\}$ and initialize $E \leftarrow F$.
4: Initialize $R_i \leftarrow \emptyset$ for all $i \in [k]$.
5: **for each** $i = 1, 2, \ldots, k$ **do**
6: **for each** $D \in H_i$ **do**
7: Add new vertex v to R_i.
8: Add edges (w_j, v) to E for each $j \in D$.
9: **end for**
10: **end for**
11: Set $C \leftarrow \bigcup_{i=1}^{k} R_i$.

The proof of network coding throughput is at least $a = 1$ is the same as Lemma 2.4.

For the other parameters in Lemma 2.1 for the gap instance, the size of the third layer dominates the number of vertices, and hence $n = \Theta(\delta N k)$ by the robust distance LDC, as we assume $\delta = \Omega(1/k)$. The number of edges $m = \Theta(nq)$, and the number of sinks $r = \Theta(n)$. The distance b is at least $\Omega(d)$, as for any $i \in [k]$ and any pair of decoding hyperedges in H_i, their distance is at least d by Definition 4.2, and any path from a vertex $u \in R_i$ to another $v \in R_i$ in $G \setminus F$ in the gap instance maps to a path in the hypergraph H_i, such that the consecutive hyperedges in the path intersect. Therefore, the distance parameter is guaranteed by the robust distance LDC. The number of cut edges is $f = N$. Upon Lemma 2.1, the coding gap is at least

$$\frac{a \cdot r}{f + 2m/b} = \Omega\left(\min\left\{\frac{r}{f}, \frac{r}{2m/b}\right\}\right) = \Omega\left(\min\left\{\delta k, d/q\right\}\right).$$

□

C.2 Proof of Proposition 4.4

The proof closely follows the argument in Section 2.1, which we now reinterpret its implication in the language of coding theory. We first construct the tripartite graph as in Algorithm 1, with the only modification lying in the pruning subroutine: as we no longer assume $q = o(\log k)$, we modify the pruning condition for a sink vertex at line 16 in Algorithm 1 to

$$\mathrm{dist}_{G \setminus F}(u, v) \leq \frac{\log k}{\log q + \log\log k}.$$

With this new condition, the arguments in Lemma 2.3 continue to hold under the updated parameters.

For each sink $v \in R_i$, let D denote the coordinates of the codeword corresponding to the neighbors of v in the second layer B. We add a hyperedge containing the coordinates in D to the hypergraph matching H_i. We now show the third parameter of the robust distance LDC holds. The distance between any two hyperedges in H is at least half the distance between the corresponding sinks u and v in the tripartite graph $G \setminus F$. Therefore, for any $i \in [k]$ and $e, e' \in H_i$, the distance of them in H is then lower bounded by $\frac{1}{2} \cdot \frac{\log k}{\log q + \log\log k}$.

The remaining is to lower bound the size of each matching H_i. At sampling subroutine, in total at least $\delta N/q$ hyperedges are generated and each is sampled to keep with probability $(\log k)/k$. The pruning subroutine then delete subconstant fraction of the matching. Therefore, the size $|H_i| = \Omega(\frac{\delta N \log k}{q \log\log k})$, implies the second parameter of the robust distance LDC code.

D Re-Analysis of Matching Vector Codes

The main purpose of this section is to prove Lemma 1.6.

Definition D.1 (Matching Vector Family)**.** Let $S \subseteq \mathbb{Z}_m \setminus \{0\}$. The families of vertors $\mathcal{U} = \{u_i\}_{i=1}^n$ and $\mathcal{V} = \{v_i\}_{i=1}^n$ of vectors in $\mathbb{Z}_m^h$ is said to be *S-matching family* if the following conditions hold:

1. $\langle u_i, v_i \rangle = 0$ for every $i \in [n]$.
2. $\langle u_i, v_j \rangle \in S$ for every $i \neq j$.

We will show how to use matching vector families to construct LDCs. We start with some preliminaries.

Fact D.2 ([Efr09])**.** *For every odd m there exists $t \leq m$ such that $m \mid 2^t - 1$. For the finite field $\mathbf{F}_{2^t} = GF(2^t)$, there exists $g \in \mathbf{F}_{2^t}$ that generates a subgroup of size m, i.e. $g^m = 1$ and $g^i \neq 1$ for $1 \leq i < m$.*

Definition D.3. Let g be the generator of $\mathbf{F}$. A polynomial $P \in \mathbf{F}[x]$ is called an *S-decoding polynomial* if the following conditions hold:

- $\forall i \in S, P(g^i) = 0$,
- $P(g^0) = P(1) = 1$.

Fact D.4 ([Efr09])**.** *For any S such that $0 \notin S$ there exists an S-decoding polynomial P of degree $|S|$.*

Lemma D.5 ([Efr09])**.** *Let $\mathcal{U}, \mathcal{V}$ be a family of S-matching vectors in $\mathbb{Z}_m^h$, $|\mathcal{U}| = |\mathcal{V}| = k, |S| = s$. For $1 \leq t < m$ such that $m \mid 2^t - 1$, there exists a linear code $C : \mathbf{F}_{2^t}^k \to \mathbf{F}_{2^t}^{m^h}$ that is $(s+1, \delta, (s+1)\delta)$-locally decodable for all δ, and is correct on uncorrupted codewords.*

Proof. We specify the encoding and decoding procedures. We simply denote the field $\mathbf{F}_{2^t}$ as $\mathbf{F}$.

- **Encoding:** Let $e_i \in \mathbf{F}^k$ be the i-th unit vector. We shall define $C(e_i) : \mathbf{F}^k \to \mathbf{F}^{m^h}$ for all i, and take the encoding function $C(\sum x_i e_i) := \sum x_i C(e_i)$ where the addition and multiplication are coordinate-wise. The encoding of e_i is defined by a complete evaluation of a function $f_i : (\mathbb{Z}_m)^h \to \mathbf{F}$, which is defined as

$$f_i(z) = g^{\langle u_i, z\rangle}, \text{ and } C_i(e_i) = (f_i(z))_{z \in (\mathbb{Z}_m)^h} .$$

- **Decoding:** The input to the decoder is a (corrupted) codeword y and an index i. Write the S-decoding polynomial in Fact D.4 as $P(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_s z^s$. The decoder performs the following:

  1. Pick $w \in \mathbb{Z}_m^h$ uniformly at random.
  2. Query the codeword at $y(w), y(w + v_i), y(w + 2v_i), \ldots, y(w + s \cdot v_i)$.
  3. Output $g^{-\langle u_i, w\rangle}(a_0 y(w) + a_1 y(w + v_i) + a_2 y(w + 2v_i) + \ldots + a_s y(w + s \cdot v_i))$.

We show that the code is correct on uncorrupted codewords and smoothness, implying the lemma by Fact 1.4. The smoothness is due to the uniform random of the point w.

To show correctness on uncorrupted codewords, for any i and w,

$$a_0 y(w) + a_1 y(w + v_i) + a_2 y(w + 2v_i) + \ldots + a_s y(w + s \cdot v_i) = \sum_{\ell=0}^{s} a_\ell \sum_{j=1}^{k} x_j g^{\langle u_j, w + \ell \cdot v_i\rangle} .$$

Rearranging terms, we have

$$\begin{aligned}
\sum_{\ell=0}^{s} \sum_{j=1}^{k} x_j g^{\langle u_j, w + \ell \cdot v_i\rangle} &= \sum_{j=1}^{k} g^{\langle u_j, w\rangle} x_j \sum_{\ell=0}^{s} a_\ell \left(g^{\langle u_j, v_i\rangle}\right)^\ell \\
&= \sum_{j=1}^{k} g^{\langle u_j, w\rangle} x_j P\left(g^{\langle u_j, v_i\rangle}\right) \\
&= g^{\langle u_i, w\rangle} x_i .
\end{aligned}$$

□

Definition D.6. Let $m = \prod_{i=1}^{t} p_i$ be a product of distinct primes. The *canonical set* in $\mathbb{Z}_m$ is the set of all non-zero s such that for every $i \in [t]$, $s \in \{0, 1\} \bmod p_i$.

Lemma D.7 ([DGY11])**.** *Let $m = \prod_{i=1}^{t} p_i$ be a product of distinct primes. Let w be a positive integer. Let $\{e_i\}, i \in [t]$ be integers such that for all i, we have $p_i^{e_i} > w^{1/t}$. Let $d = \max_i p_i^{e_i}$ and $h \geq w$ be arbitrary. Let S be the canonical set; then there exists an $\binom{h}{w}$-sized family of S-matching vectors in $\mathbb{Z}_m^n$, where $n = \binom{h}{\leq d}$.*

Proof of Lemma 1.6. The proof follows by setting appropriate parameters in Lemma D.7 and applying Lemma D.5.

1. By a strengthened version of Bertrand's postulate (Theorem 5.8 in [Sho09]), there exists a constant c such that the interval $[(c/2)t \ln t, ct \ln t]$ contains at least t distinct odd primes $p_1, \ldots, p_t$.
2. Let $m = \prod_{i \in [t]} p_i$. This implies $m = t^{\Theta(t)}$.
3. The size of canonical set is $s = |S| = 2^t - 1$.
4. Define b as the smallest positive integer such that $m \mid 2^b - 1$. Then $b \leq m - 1$.
5. Assume w is a multiple of t, and set $k = w^{w/t}$. Then it follows that $w = \Theta\left(\frac{t \log k}{\log \log k}\right)$.
6. Let $d = \max_i p_i^{e_i} = O(w^{1/t} \cdot t \ln t)$.
7. Set $h = c' \cdot w^{1+1/t}$, where c' is a sufficiently large constant to ensure $h \geq d$.
8. Observe that
$$\binom{h}{w} \geq \left(\frac{h}{w}\right)^w \geq k.$$
9. Also, note that
$$\binom{h}{\leq d} \leq d \cdot \left(\frac{eh}{d}\right)^d.$$
10. Then the total code length is
$$N = m^{\binom{h}{\leq d}} \leq \exp\exp\left(t \ln t \cdot w^{1/t} \ln w\right) \leq \exp\exp\left(O\left((\log k)^{1/t} (\log \log k)^{1-1/t} \cdot t \ln t\right)\right).$$

Finally, the assumption in (5) $k = w^{w/t}$ can be made without loss of generality. If k does not exactly have this form, we can pad the message with zeros to obtain a message of length k' that satisfies this structure. This padding incurs at most a quadratic blowup, which does not affect the asymptotic bounds.

□