Universal and Robust Distributed Network Codes

Tracey Ho†, Sidharth Jaggi‡, Svitlana Vyetrenko†
† California Institute of Technology, ‡ The Chinese University of Hong Kong
†{tho, svitlana}@caltech.edu, ‡jaggi@ie.cuhk.edu.hk

Abstract

Random linear network codes can be designed and implemented in a distributed manner, with low computational complexity. However, these codes are classically implemented [1] over finite fields whose size depends on some global network parameters (size of the network, the number of sinks) that may not be known prior to code design. Also, if new nodes join the entire network code may have to be redesigned.

In this work, we present the *first universal and robust* distributed linear network coding schemes. Our schemes are universal since they are independent of all network parameters. They are robust since if nodes join or leave, the remaining nodes do not need to change their coding operations and the receivers can still decode. They are distributed since nodes need only have topological information about the part of the network upstream of them, which can be naturally streamed as part of the communication protocol.

We present both probabilistic and deterministic schemes that are all asymptotically rate-optimal in the coding block-length, and have guarantees of correctness. Our probabilistic designs are computationally efficient, with order-optimal complexity. Our deterministic designs guarantee zero error decoding, albeit via codes with high computational complexity in general. Our coding schemes are based on network codes over "nested subsets of an infinite field". Instead of choosing coding coefficients from one field at every node as in [1], each node uses linear coding operations over an "effective field-size" that depends on the node's distance from the source node. The analysis of our schemes requires technical tools that may be of independent interest. In particular, we generalize the Schwartz-Zippel lemma [2] by proving a non-uniform version, wherein variables are chosen from sets of possibly different sizes. We also provide a novel robust distributed algorithm to assign unique IDs to network nodes.

I. Introduction

The paradigm of network coding allows each node in a network to process information in a non-trivial manner. As shown in [3], [4], [5], if intermediate nodes simply perform linear operations over some finite field, the resulting *network codes* can be rate-optimal for a large class of communication problems. In particular algorithms that design codes for *multicast* communication problems, wherein each of multiple sinks requires the same information from a source node, have been well-studied computationally efficient deterministic [6], [7] and probabilistic [1], [8] designs of such codes exist.

However, for all current network codes, some information of network parameters is necessary to determine the size of the finite field over which coding is performed. The centralized algorithms in [6], [7] require knowledge of the entire network, and even the decentralized algorithms in [1], [8] require knowledge of the network size and the number of sinks, or at least an upper bound. If these parameters are unavailable, such codes have no guarantees of correctness, hence prior codes are not universal. Moreover, if the network topology dynamically changes, the addition of even one new node might require the entire network code to be updated, hence such codes are also not robust.

In this work we develop the *first universal and robust* distributed linear codes that are independent of all network parameters, and are designed to satisfy a pre-specified tolerance on the error-probability (defined as the probability that the linear transform from the source to some sink is not invertible). The essential idea behind our design is that of using "finite subsets of an infinite field". Linear coding operations are chosen from finite subsets of an appropriate infinite field – in particular we choose $\mathbb{F}_2(z)$ the field of

Parts of this work were in the thesis [9] and presented in [10], and a preliminary version of this work was presented at [16].

rational functions over $\mathbb{F}_2$, *i.e.*, the field whose elements are ratios of binary polynomials. Operations over this field can be implemented via binary filters (convolutional codes) at each node.[1]

As information percolates down the network, each node makes its own estimate of the "effective field size", *i.e.*, the size of the subset of $\mathbb{F}_2(z)$ from which that node should choose its coding operations, so as to meet the guarantee on the pre-specified tolerance on the overall error-probability. Our codes are able to perform this book-keeping despite having access only to information that can be percolated down the network at rates that are asymptotically negligible in the block-length – like standard distributed network codes, our codes are also asymptotically rate-optimal.

Our results are as follows. In Section III we prove a generalization of the Schwartz-Zippel lemma [2] that is useful as a technical tool in some of our code constructions; it may also be of independent interest for other universal algorithms.

In Section IV we present probabilistic universal and robust codes. That is, given any $\epsilon > 0$ and any network, we design codes that guarantee that the linear transform from the source to each sink is invertible with probability at least $1 - \epsilon$. Further, even if the network changes, pre-existing nodes do not need to change their coding operations to preserve the same guarantee of correctness. We present two such codes. The first is independent of network size, but depends on the number of sinks. We present it primarily for exposition, since its presentation is simpler than that of our second code, which is independent of all network parameters. Both constructions base their choices of coding operations on their distance from the source. While the "effective field-size" of our codes (and hence their implementation complexity) are larger than those of the non-universal codes [1], [8], their complexity is still polynomial in network parameters. Also, in Theorem 3 we present a class of networks that demonstrates that our codes have essentially order-optimal computational complexity if universality is required.

In Section V we consider deterministic universal and robust codes. As a technical tool we first discuss a decentralized algorithm to distribute unique IDs to each node in a robust manner – even if the network changes we guarantee that it too can be given an ID that is distinct from all others in the network. Building on this tool, and a novel use of Cantor's classical mapping between $\mathbb{Z}$ and $\mathbb{Z}^n$ for any finite n, we design zero-error decentralized codes that are independent of all network parameters, and robust to changes in network topology. We provide two constructions. Our first construction, also primarily expository, is just for codes of rate 2, and is computationally efficient to design and implement. Our second construction is for arbitrary rate codes. This generalization comes at the cost of exponentially increasing the implementation complexity, compared to our other constructions.

We distinguish between the computational complexity of design and that of implementation. The former refers to the the computational cost of designing the coding operations at each node, and is a one-time cost. The latter corresponds to the computational cost incurred by each node as it implements the pre-designed coding operations, and is a repeated cost for each packet transmitted by that node. All our codes have design complexity that is at most polynomial in the network parameters. Further, most of our designed codes codes have implementation complexity that is also polynomial in network parameters; the only exception is the last of the proposed designs corresponding to the general design for zero-error universal and robust distributed linear codes.

We note that all our algorithms provide guarantees of correctness as long as the source transmits information at a rate no greater than can be supported by the network, *i.e.*, its min-cut. We view the rate-control issue as independent of code design – our codes are independent of the size of the min-cut.

A. Related work

The distributed random linear codes of [1], [8] require field-sizes to scale roughly as $|\mathcal{E}||\mathcal{T}|$. As shown in [11], even with centralized design of network codes, the field size over which coding must be performed as at least $|\mathcal{T}|$.

[1]By such finite subsets we do *not* mean embeddings of, for example, $\mathbb{F}_2$ into $\mathbb{F}_{2^2}$, since the latter is still a finite field. Rather, we consider subsets of an infinite field over which linear network codes can be meaningfully defined. One example, other than the field $\mathbb{F}_2(z)$ we explicitly consider in this paper, is the field of rational numbers $\mathbb{Q}$.

As to universal codes (codes independent of some problem parameters), they have been well-studied in the classical information-theory setting (for instance in source coding [12] and channel coding [13]).

In the network coding setting, however, the literature is much sparser. The work of [5] proposes "robust network codes" that are resilient to network failure patterns. However, the field-size over which coding is performed depends on the number of failure patterns, and hence these codes are not truly universal. Further, the computational complexity of designing such codes is prohibitive. There is also significant work on network coding for packet erasure networks (for instance [14]). Our codes can tolerate all such errors.

The work of [15] examines "decentralized network coding" in which new nodes can join a network without disrupting pre-designed coding operations. Here, too, the field-size choice for the initial design depends on the size of the network. Further, the code designs are for special cases – either for rate-2 codes (analogous to the codes we present in Section V-C) or for networks with only two sink nodes.

II. Notation and definitions

A. Network Model

In this paper, we adopt the single-source multicast network model of [5]. Let the network be represented by a directed acyclicgraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ the set of edges. The graph has a pre-specified source node s and $|\mathcal{T}|$ sink nodes $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$. A directed edge e from node u to node v is said to have *tail* u (denoted $tail(e)$) and *head* v (denoted $head(e)$). The link e is then said to be an *incident outgoing link* of u and an *incident incoming link* of v.

B. Communication Model

The communication goal is for the source to communicate identical information to each sink.

As is standard [5], we assume that each link carries at most one packet of information per time-step. The *packet-length* in bits is denoted by n.

The *network capacity*, denoted by C, is the time-average of the maximum number of packets that can be delivered from the source s to each sink $t \in \mathcal{T}$ simultaneously. It can be also expressed as *the minimum of the min-cut from the source s to each sink t*. The rate R is the average number of *information* packets that the source s generates per time-step, to be delivered to each sink t over the network $\mathcal{G}$. Without loss of generality we assume that $R < C$. Lastly, let c denote the maximum capacity of any single link in the network.

C. Code Model

1) Network code: The *network code* comprises of the encoders at the source and each node inside the network, and the decoders at each of the sinks. In particular we focus on *linear network codes*, *i.e.*, codes where the source node, each internal node, and each sink performs linear combinations of information in packets on incident incoming links to generate packets on incident outgoing links. Specifically we consider the class of *convolutional* linear operations, well-studied in classical coding theory, that we reprise below. The base-field for arithmetic is chosen to be $\mathbb{F}_2$, hence all operations described below are binary.

2) Convolutional network code: Recall that the *z-transform* [17] of any sequence $\{a(i)\}_{i=1}^{n}$ of bits is given by the polynomial $\sum_{i=1}^{n} a(i)z^i$, denoted $A(z)$. Further, recall that the output of the *convolution operation* $a * b$ between two sequences[2] $\{a(i)\}_{i=1}^{n}$ and $\{b(i)\}_{i=1}^{n'}$ is defined as the length $(n+n')$ sequence whose ith term equals $\sum_{j=1}^{i} a(j)b(n'-j+i)$. Lastly, it is well-known that the z-transform of $a * b$ equals $A(z)B(z)$.

Convolutional codes [17] have long been used in point-to-point communication scenarios. The idea of using convolutional codes for network coding (in networks with cycles) was foreshadowed in [3], and

[2]Terms $a(i)$ and $b(j)$ are respectively set to zero for $i \notin \{1, 2, \ldots, n\}$ and $j \notin \{1, 2, \ldots, n'\}$.

made explicit in [5] (who also noted that such an algebraic model for coding operations can help kill two birds with one stone, *i.e.*, it can also help model delays in networks). The work of Erez *et al.* (see for example [18]) gave the first efficient designs for convolutional network codes, *i.e.*, codes over $\mathbb{F}_2(z)$. In our work, $\mathbb{F}_2(z)$ affords the advantage that it allows for coding operations to be chosen from a potentially unbounded set, which helps us circumvent the difficulty that we do not know the network's parameters in advance.

The source's packets are denoted by $X_1, X_2, \dots, X_R$ – each is a length-n bit-vector. The corresponding z-transforms are denoted $X_1(z), X_2(z), \dots, X_R(z)$. Collectively they are represented by the length-R vector of polynomials $\mathbf{X}(z)$. Each edge e carries the packet Y_e, and its z-transform is denoted $Y_e(z)$. Lastly, the z-transform of the packets on incident incoming links to any sink t are denoted by the vector $\mathbf{Y}_t(z)$. We henceforth refer to a sequence and its z-transform interchangeably.

Let u, v and w be three nodes such that there is at least one edge from u to v and at least one edge from v to w. We use a 5-tuple to denote a coding choice for such nodes – specifically, $\beta_{u,i,v,j,w}(z)$ refers to the *local coding coefficient* of the convolutional operation on the information on the ith edge from u to v to the jth edge from v to w. The choices of values of the local coding coefficients $\beta_{u,i,v,j,w}(z)$ are code design parameters whose specifications are the primary objective of subsequent sections. Let e be a specific edge from v to w, and e' denote a dummy variable that ranges over all edges incoming to v (and hence is indexed by the pair (u, i)). Thus the convolutional operation that is performed at node v comprises of taking linear combinations of the information $Y_{e'}$ with the appropriate $\beta_{u,i,v,j,w}(z)$ over all edges e' incident incoming to node v, to generate the information Y_e on the edge e incident outgoing from v. To simplify notation, we henceforth write $\beta_{u,i,v,j,w}(z)$ simply as $\beta_{e,e'}(z)$ with the understanding that (e, e') index the appropriate 5-tuple. Thus the linear transform at each node can be written symbolically as

$$Y_e(z) = \sum_{e'} \beta_{e',e}(z) Y_{e'}(z).$$

Since all the linear operations performed by the network can be represented via operations over polynomials over the binary field, we henceforth consider all arithmetic to be over the *field of rational functions* [5] over $\mathbb{F}_2$, denoted by $\mathbb{F}_2(z)$. The elements of this field are of the form $P(z)/Q(z)$, where both $P(z)$ and $Q(z)$ are binary polynomials. Linear codes over this field have been well-studied in the convolutional coding literature [17].

As in classical distributed network codes [1], the codes in this work are *distributed*, *i.e.*, the choice of a value for $\beta_{u,i,v,j,w}(z)$ at node v can depend only on its local parameters (u, i, v, j, w), and the corresponding parameters of the nodes upstream of node v. Since we consider only directed acyclic networks in this work, this imposes a significant design constraint, since nodes that cannot directly communicate with each other over the network cannot coordinate their coding choices.

One idea of [1] that we too use is the idea of having "short headers" in each transmitted packet. Specifically, each packet (containing n bits) transmitted by the source, also contains the linear transformation induced by the network from the source to that packet – as in [1] these transforms are computed in a distributed manner and percolated down the network along with the payload information at an asymptotically negligible rate. For every $t \in \mathcal{T}$, let T_t be the *network transfer matrix* from s to t – these too can be computed in a distributed manner. Let T be the *overall network transfer matrix* from s to $\mathcal{T}$ formed as $\Pi_{t \in \mathcal{T}} T_t$. Let $|T_t|$ and $|T|$ denote the determinants of T_t and T respectively.

Our codes are either *probabilistic* or *deterministic* depending on whether local coding coefficients are chosen probabilistically or deterministically. The *error probability* is the probability over choices of local coding coefficients that for each source message, at least one sink's reconstruction of at least one possible message from the source is inaccurate. For linear network codes correct coding happens if and only if the transfer matrix from the source to each sink is invertible [5]. Rate R is said to be achievable if for any $\epsilon > 0$ and $\delta > 0$ there exists a coding scheme of block length n with rate $\geq R - \delta$ and error probability $\leq \epsilon$. In particular, we require our deterministic network codes to be *zero-error*, *i.e.*, to have error probability be zero.

III. The Generalized Schwartz-Zippel Lemma

The classical Schwartz-Zippel lemma [2] provides an upper bound on the probability that a polynomial evaluates to zero when variables of it are chosen uniformly at random from a field.

Recall that the *degree d_i of a variable x_i in a polynomial $P(x_1, x_2, \ldots, x_N)$* is the maximal exponent of x_i in its non-zero terms. Further, recall that the *degree d of a polynomial $P(x_1, x_2, \ldots, x_N)$* itself is the maximal value among the sum of the exponents of all its non-zero terms. Note that $d \leq \sum d_i$.

Lemma 1 (Schwartz-Zippel lemma [2])**.** *Let $P(x_1, x_2, \ldots, x_N)$ be a non-zero polynomial of degree $d \geq 0$ over a field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$, and the value of each $x_1, x_2, \ldots, x_N$ be selected independently and uniformly at random from S. Then the probability that the polynomial equals zero is at most $d/|S|$.*

The Schwartz-Zippel lemma is a useful tool in the analysis of random linear network codes (for instance [1]). A random linear network code causes an error if and only if one of the transfer matrices from the source to the destination is singular. This in turn happens if and only if the product of the determinants of these transfer matrices equals zero. But this product of determinants may be viewed as a polynomial whose variables consist of the local coding coefficients at each node. Hence the Schwartz-Zippel lemma provides an upper bound on the probability of error of a random linear network code.

In this work we are interested in a generalization of the Schwartz-Zippel lemma, for polynomials whose variables are chosen from different subsets of $\mathbb{F}$. We prove:

Lemma 2. *Let $P(x_1, x_2, \ldots, x_N)$ be a non-zero polynomial over a field $\mathbb{F}$. For all $i \in \{1, 2, \ldots, N\}$, let S_i be a finite subset of $\mathbb{F}$, the degree of x_i in $P(x_1, x_2, \ldots, x_N)$ be d_i, and the value of each x_i be selected independently and uniformly at random from S_i. Then the probability that the polynomial equals zero is at most $\sum_{i=1}^{N} (d_i/|S_i|)$.*[3]

Proof: Given in Appendix A □

Note 1**:** Neither Lemma 1 nor Lemma 2 put any restriction on the size of the field $\mathbb{F}$, as long as the appropriate subsets from which variables chosen are finite.

Note 2**:** A related but inequivalent generalization of the Schwartz-Zippel lemma was proved in [19]. The utility of this proof is that it allows for the variables comprising the polynomial to be chosen non-uniformly. This is integral to the proof techniques in this work, wherein we choose local coding coefficients over progressively larger sets, depending on how far from the source they are.

IV. Probabilistic designs

In this section we describe two probabilistic designs of universal distributed robust network codes. In particular, given any $\epsilon > 0$, we present schemes such that the overall error probability of the code is at most ϵ.

Our first scheme is independent of the size (number of nodes/edges) of the network, but does requires that the source has *a priori* knowledge of the number of sinks it shall be required to service. Hence we say it is only *weakly universal*. Our purpose in presenting this scheme is primarily expository, since the proof is significantly easier than that of the second scheme – it helps set the stage for the second scheme.

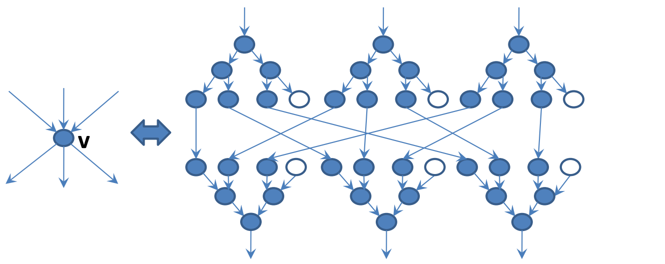
The second scheme is *strongly universal* and is independent of all network parameters, including the number of sinks.

We first describe some useful preprocessing steps relevant for both of our schemes.

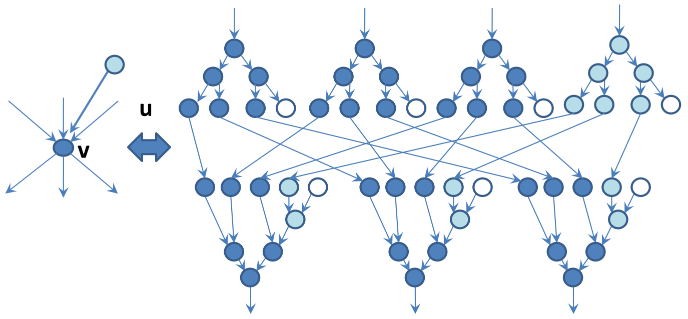
A. Graph transformation

We find it desirable to work over a transformed graph rather than the original graph $(\mathcal{V}, \mathcal{E})$. This transformation can be done locally at each node, and results in a graph with some useful properties. In

[3]This is not the tightest bound possible for all polynomials, but it has some advantages. In particular, the result presented is strong enough to prove the theorems in this work, and has a simple enough form that using it is not unwieldy. In the proof of this lemma in the Appendix we outline how it can be tightened further.



(a) A node v with three incoming and three outgoing links, and its corresponding virtual robust gadget.



(b) Modification to the virtual robust gadget $G(v)$ of node v when a new node w connects to v.

Fig. 1. Construction of a virtual robust gadget for node v.

particular, we use the work of [7] which demonstrates the equivalence between general network coding problems and those over "low-degree" networks where each node has degree at most three. In particular, nodes in the reduced network can be of either one of the two types:

- Nodes with one incident incoming edge and at most two incident outgoing edges (in which case they broadcast the incoming information on incident outgoing edges, and hence are called *broadcasting nodes*, see Figure 2(b)).
- Node with two incident incoming edges and one incident outgoing edge (in which case they code the information on the incident incoming edges to generate information transmitted on the incident outgoing edge, and hence are called *coding nodes*. see Figure 2(a))

This equivalence is useful for our probabilistic algorithms since it allows us to effectively enumerate networks. We change the equivalence relationship of [7] slightly as described below so as to make it robust to nodes joining the original network. That is, in our equivalence relationship, nodes can join the original network while only locally perturbing the "low-degree" network.

The transformed graph $(\mathcal{V}', \mathcal{E}')$ is as constructed follows. For every node $v \in \mathcal{V}$ we construct a *virtual robust gadget* $G(v)$ (see Figure 1(a) for an example[4]).

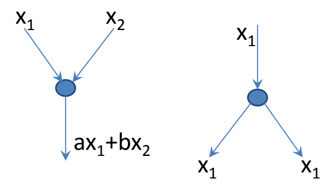
Suppose v has $d_{in}(v)$ incoming links and $d_{out}(v)$ outgoing links. Corresponding each incoming link we construct a binary tree whose root is connected to that incoming link, and which has $d_{out}(v) + 1$ leaves. Similarly, corresponding to each outgoing link we construct an inverted binary tree whose root is connected to that outgoing link, and which has $d_{in}(v) + 1$ leaves. The last leaf node of each binary tree is called a *virtual node*, and the other leaf nodes are called *connection nodes*. We then connect connection nodes so that there is exactly one path from each incoming link to each outgoing link (the connection order does not matter).

Suppose a new link is created in the network – say, a link directed from w to v (see Figure 1(b) for an example). In this case we first create a virtual gadget corresponding to the directed edge (w, v). We then split each virtual node on the inverted binary tree (corresponding to the outgoing links of v) into two by appending a binary tree of depth one to it. We denote the second of the two new leaf nodes as a new virtual node, and the first as a new connection node. The connection nodes on (w, v)'s virtual gadget are then connected to the new connection nodes on each of the outgoing links' virtual gadgets so that there is exactly one path from (w, v) to each link outgoing from v. A corresponding (but inverted) procedure holds if the new link corresponded to a link *outgoing* from v. The removal of a link simply corresponds to removal of the corresponding virtual gadgets on the incoming and outgoing sides, and all links connected to it. Correspondingly, a new node is treated as the corresponding set of new links created in the network. Similarly, a node's departure is treated as the set of corresponding links being removed.

[4]We thank Michael Langberg for providing the template for Figures 1(a) and 1(b)

The virtual nodes in each virtual gadget are what give our transformation robustness, since in case a new node joins or leaves the network, nodes other than the ones directly connecting with the changing node experience no structural changes in their existing virtual gadgets[5].

Henceforth, all algorithms in Section IV shall convert the original graph to the virtual graph above as a pre-processing step, and all computations shall be over this virtual graph. Also, as part of normal communication each node v in the virtual graph $(\mathcal{V}', \mathcal{E}')$ estimates its *depth* $\Delta(v)$, *i.e.*, the length of the shortest path from the source to itself. This can be done by any of a variety of distributed shortest-path algorithms over acyclic graphs, such as the Bellman-Ford algorithm [20].



(a) Coding node. (b) Broadcasting node.

Fig. 2. Coding operations defined on the nodes of $\mathcal{G}'$.

B. Weakly universal design

The essential idea behind our first scheme is as follows. Each node having estimated its depth, it then chooses a subset of $\mathbb{F}_2(z)$ whose size scales exponentially in this depth, from which it pick its coding coefficients uniformly at random. We then show that the probability of error due to information being lost at any depth decays geometrically in the depth, and hence by the union bound the overall probability of error can be controlled so as not to exceed any desired ϵ.

WUP($\epsilon, |\mathcal{T}|$) (Weakly Universal Probabilistic) Code:

- Each coding node v of depth $\Delta(v)$ in the vertex-set $\mathcal{V}'$ of the virtual graph chooses two local coding coefficients corresponding to the two incoming links uniformly at random from the set of polynomials of degree at most $2\Delta(v) + 1 + \log(R|\mathcal{T}|/\epsilon)$.
  The choice of the degree bound above is simply to ease the analysis of Theorem 1. All logarithms are binary. Also, for simplicity of presentation we assume that $\log(R|\mathcal{T}|/\epsilon)$ is an integer – if not, we may round up to the nearest integer with negligible error in our estimate of parameters.

Theorem 1. *For any $\epsilon > 0$,* **WUP($\epsilon, |\mathcal{T}|$)** *has error probability at most ϵ.*

Proof: Recall that $|T_t|$ represents the determinant of the transfer matrix from the source to sink t. As noted in [5], the network code is error-free if and only if the polynomial $\Pi_t|T_t|$ comprising of the product of the $|T_t|$ determinants over all sinks (with the network's local coding coefficients $\beta_{u,i,v,j,w}(z)$ as variables) is non-zero. To evaluate the probability that this is the case given the random assignment of local coding coefficients in **WUP($\epsilon, |\mathcal{T}|$)** , we use Lemma 2. Specifically, each variable x_i in Lemma 2 corresponds to a local coding coefficient. We group the coding coefficients $\beta_{u,i,v,j,w}(z)$ in terms of the depths $\Delta(v)$ of the nodes at which they are used. But there are at most $2^{\Delta}R$ coding nodes at any depth Δ in the virtual graph, since after the transformation in Section IV-A the fast possible growth-rate for the new graph would be if it corresponded to R parallel binary trees – one for each of the source's messages. Hence there are at most $2R2^{\Delta}$ local coding coefficients at that depth. Also, Corollary 1 in [1] shows that the degree of each local coding coefficient in $\Pi_t|T_t|$ is bounded from above by $|\mathcal{T}|$. By construction in **WUP($\epsilon, |\mathcal{T}|$)**

[5]The addition of virtual nodes and the corresponding robust connection procedure is the only substantive difference between our graph transformation and that in [7], though the uses are substantially different.

each virtual node v of depth $\Delta(v)$ in the network chooses local coding coefficients uniformly at random from the set of polynomials of degree at most $2\Delta(v)+1+\log(R|\mathcal{T}|/\epsilon)$. This set is a subset of $\mathbb{F}_2(z)$ of size at most $2^{2R\Delta(v)+1+\log(|\mathcal{T}|/\epsilon)} = 2R|\mathcal{T}|2^{2(\Delta(v)}/\epsilon$. Summing over all possible local coding coefficients at all possible (possibly infinite) depths and substituting the appropriate parameters into Lemma 2, the error probability of the network code is bounded from above by

$$\begin{aligned} \Pr\left(\prod_{t\in\mathcal{T}}|T_t|)=0\right) &\leq \sum_{\Delta=1}^{\infty} 2R2^{\Delta}|\mathcal{T}|\frac{\epsilon}{2R|\mathcal{T}|2^{2\Delta}} \\ &= \sum_{\Delta=1}^{\infty}\frac{\epsilon}{2^{\Delta}} \quad = \quad \epsilon. \qquad \square \end{aligned}$$

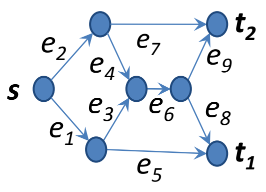
The computational complexity of **WUP($\epsilon, |\mathcal{T}|$)** codes is polynomial in network parameters and $\log(1/\epsilon)$, and the achievable rates approach the network capacity C asymptotically in the block-length. Further, our codes are robust to links joining and leaving. Since the analysis of these properties is very similar to that of the codes in Section IV-C, we delay discussion to the end of that section.

C. *Strongly universal design*

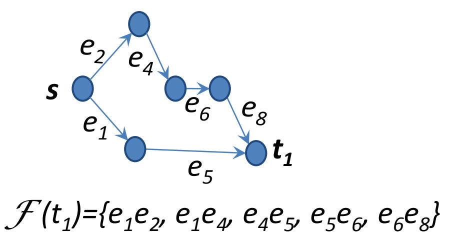
We now present design of probabilistic robust linear network codes that are *strongly universal*, *i.e.*, independent of all network parameters. This obviates the requirement of knowledge of $|\mathcal{T}|$ of the codes in Section IV-B. The idea underlying the construction in this section is as follows. For the purpose of analysis, for each sink we identify a set of edge-disjoint paths, and estimate the probability that the information on these edge-disjoint paths remains invertible as information flows through the network. In particular, for any sink t and any depth Δ in the network we identify the set of edges in these edge-disjoint paths that must contain linearly independent combinations of the source's information. We call such sets of edges *flow-cuts*. It turns out that the number of flow-cuts at any depth is in fact independent of the number of sinks, and further, a bound on this number at each depth can be computed locally. Thus sinks can be classified according flow-cuts. Hence, instead of trying to ensure that the linear transform to each sink is invertible as in Theorem 1, nodes at each depth simply try to ensure that the linear transform to each flow-cut is invertible. To analyze the probability of non-invertibility at each flow-set an alternative to the end-to-end analysis of the probability of error used in [1], [8] is required. Here we use the proof technique of [21], which analyzes the probability that information gets lost from one set of edges in the network to a neighbouring set of edges.

SUP(ϵ) (Strongly Universal Probabilistic) Code

- Each coding node v at depth $\Delta(v)$ in the vertex-set $\mathcal{V}'$ of the virtual graph chooses two local coding coefficients corresponding to the two incoming links uniformly at random from the set of polynomials of degree at most $(R+1)(\Delta(v)+1+\log R)+\Delta(v)+\log(1/\epsilon)-1$.



(a) Butterfly network.



(b) One possible choice of a flow-set $\mathcal{F}(t_1)$ for the butterfly network.

Fig. 3. Illustration of the definition of a flow-cut and a flow-set based on the butterfly network topology. The flow-set $\mathcal{F}(t_1)$ comprises of the successive flow-sets (e_1, e_2), (e_1, e_4), (e_5, e_4), (e_5, e_6), (e_8, e_6).

Recall that by assumption the capacity of the network is at least R. Hence there is a set $\mathcal{P}$ of at least R edge-disjoint paths that go from the source s to each t.

Corresponding to each such set $\mathcal{P}$ of edge-disjoint paths, we define *flow-cuts*. A *flow-cut* $F(t)$ is defined as a set of R edges that have the property that each edge in the flow-cut is from a distinct edge-disjoint path in $\mathcal{P}(t)$. These flow-cuts are useful since we intend to analyze the linear (in)dependence of information flowing through the edges in each flow-cut – if the information on each edge in a flow-cut is linearly independent, then the source's information can be retrieved from that flow-cut. Hence, we only need to inductively prove that no information is lost from one flow-cut to the "next" flow-cut, appropriately defined as below[6].

We define the *depth* $\Delta(F(t))$ *of a flow-cut* $F(t)$ as the maximum depth of the head of any edge in it, *i.e.*, $\Delta(F(t)) = \max_{head(e) \in F(t)} \Delta(e_i)$. Further, we denote a flow-cut of depth Δ by $F(t, \Delta)$.

We then define a *flow-set* $\mathcal{F}(t)$ as an ordered set of flow-cuts with the following properties. In particular, each flow-cut in a flow-set differs from the successive flow-cut in exactly one edge. Specifically, if one flow-cut in $\mathcal{F}(t)$ differs from the next flow-cut in $\mathcal{F}(t)$ in that some edge e is replaced by another e', then it must be the case that e is the edge preceding e' in some path in the set of edge-disjoint paths $\mathcal{P}(t)$. Intuitively, each flow-set $\mathcal{F}(t)$ captures successive snapshots of how information flows from the source to the sink t.

Examples of flow-cuts and flow-sets are provided in Figure 3(b), based on the butterfly network in Figure 3(a).

Let $F(t, \Delta)$ be some flow-cut of depth Δ to sink t, and $F'(t)$ be the flow-cut immediately preceding[7] $F(t, \Delta)$ in flow-set $\mathcal{F}(t)$. Let $T(F(t, \Delta))$ be the linear transform that the network imposes from the source s to the edges in the flow-cut $F(t, \Delta)$, and let $\rho(\Delta)$ be the rank of this transform. Correspondingly, let $T(F'(t))$ be the linear transform from s to $F'(t)$, and let ρ' be the rank of this transform. Then the following lemma gives an upper bound on the probability that choosing local coding coefficients according to the dictates of **WUP(**$\epsilon, |\mathcal{T}|$**)** results in a loss of information in going from $F'(t)$ to $F(t, \Delta)$.

Theorem 2. *For every* $\epsilon > 0$*,* **SUP(**ϵ**)** *has error probability at most* ϵ*.*

Proof: Note that of the two types of nodes in the virtual graph, the broadcasting nodes induce no additional error – if a flow-cut contains R linearly independent packets, and one of the edges in the flow-cut is replaced with another edge at a broadcasting node, the information in the succeeding flow-cut remains unchanged. Thus from now on we focus only on coding nodes.

By construction the structure of the virtual graph $(\mathcal{V}', \mathcal{E}')$ is such that each node can have at most two outgoing edges, and further the source node is replicated R times. Hence the maximum possible number of edges in the virtual graph up to a depth Δ occurs when it comprises of R parallel binary trees. But each binary tree has at most $2(2^{\Delta})$ edges, hence the total number of edges in the virtual graph up to depth Δ is at most $2R(2^{\Delta})$. Also, the total number of flow-sets of depth Δ is at most $\binom{2R(2^{\Delta})}{R}$, which is bounded from above[8] by $exp(R(\Delta + 1 + \log R))$.

We use these bounds to bound from above the number of distinct type of coding choices a coding node at a certain depth faces. All our analysis now focuses on the specific following coding node v. Let its incoming edges be e' and e'', and the outgoing edge be e. Let edge e' belong to a flow-cut $F'(t)$ in flow-set $\mathcal{F}(t)$ going towards sink t, and edge e'' be an arbitrary other edge. Then the outgoing edge e replaces e' in the flow-cut $F'(t)$ to produce flow-cut $F(t)$. Suppose $F(t)$ is of depth Δ. Then by the bounds in the preceding paragraph, the number of ways an arbitrary flow-cut of depth Δ can

[6]Similar intuition was used in the proofs of [6] and [21], where they were called "frontier edge-sets". Note that a flow-cut need not be a cut or a subset of it – for instance, it may include two edges on two edge-disjoint paths, such that one is incoming to a node, and the other is outgoing from it.

[7]Note that the depth of $F'(t)$ might be either Δ or $\Delta - 1$, since two successive flow-cuts differ in exactly one edge, which may or may not be the deepest edge in a flow-cut (if not, then both flow-cuts have the same depth; if so, the depth of the flow-cut can change by at most one).

[8]Since $\binom{a}{b} \leq a^b$. Also, all exponents exp are base 2.

result from the merger of a preceding flow-cut and an arbitrary edge of depth at most Δ is at most $2R(exp(\Delta)) \times exp(R(\Delta + 1 + \log R))$, which equals

$$exp((R+1)(\Delta + 1 + \log R)). \tag{1}$$

Next, we estimate the probability that a coding node "loses information". That is, we bound from above the probability that the number of linearly independent packets on the edges of a flow-cut is less than R even though the immediately preceding flow-cut has R linearly independent packets.

Say $T(F(t,\Delta))$ represents the $R \times R$ matrix whose ith row represents the linear combinations of the source's R messages on the ith link in the flow-cut $F(t,\Delta)$ of depth Δ. Correspondingly, let $T(F'(t))$ represent the matrix representing the linear transform from the source to the flow-cut $F'(t)$ immediately preceding $F(t,\Delta)$ in the flow-set $\mathcal{F}(t)$, and suppose it is of full rank R. Then the message $Y_e(z)$ on edge e in flow-cut $F(t,\Delta)$ may be written as $\beta_{e',e}(z)Y_{e'}(z) + \beta_{e'',e}(z)Y_{e''}(z)$. (Recall that e' and e'' are the edges incoming to v, $Y_{e'}(z)$ and $Y_{e''}(z)$ are the corresponding messages carried by them, and $\beta_{e',e}(z)$ and $\beta_{e'',e}(z)$ represent the local coding coefficients at node v.) But by assumption $T(F'(t))$ is of full-rank, and hence the message $Y_{e''}(z)$ may be written as a linear combination of the messages on the edges in flow-set $\mathcal{F}'(t)$. Thus the message on edge e may be written as

$$\beta_{e',e}(z)Y_{e'}(z) + \sum_{e(i)\in F'(t)} \gamma_{e(i),e}(z)Y_{e''}(z),$$

for some $\gamma_{e(i),e}(z) \in \mathbb{F}_2(z)$. This in turn equals

$$(\beta_{e',e}(z) + \gamma_{e'',e}(z))Y_{e'}(z) + \sum_{e(i)\in F'(t):e(i)\neq e'} \gamma_{e(i),e}(z)Y_{e''}(z).$$

But the information on the links in $\mathcal{F}(t,\Delta)$ other than e is unchanged, and hence the only manner in which the messages on the edges in $\mathcal{F}(t,\Delta)$ are linearly dependent is if the coefficient $(\beta_{e',e}(z) + \gamma_{e'',e}(z))$ equals zero. But by the choice specified in **SUP(ϵ)** the coding coefficients are chosen from the set of polynomials of degree at most $([(R+1)(\Delta+1+\log R) + \Delta + \log(1/\epsilon - 1)])$ – this set is of size $exp\left([(R+1)(\Delta+1+\log R) + \Delta + \log(1/\epsilon)]\right)$. Lemma 1 then implies that the probability that the degree 1 polynomial $(\beta_{e',e}(z)+\gamma_{e'',e}(z))$ equals zero, is at most $exp\left(-\left[(R+1)(\Delta+1+\log R) + \Delta + \log(1/\epsilon)\right]\right)$. Analogously to Theorem 1, taking the union bound over all possible coding operations at depth Δ (for which (1) is an upper bound), and summing over all (possibly infinite) depths Δ gives us that the overall probability of error is at most

$$\sum_{\Delta=1}^{\infty} 2^{(R+1)(\Delta+1+\log R)} \frac{\epsilon}{2^{\Delta+([(R+1)(\Delta+1+\log R)])}}$$
$$= \sum_{\Delta=1}^{\infty} \frac{\epsilon}{2^{\Delta}} \quad = \quad \epsilon. \qquad \square$$

D. Robustness

Due to the robust graph transformation described in Section IV-A, neither the addition nor the deletion of edges or nodes in the network causes problems with our proof. If new nodes are added to the network, the actual depth of some nodes in the network, say v in particular, may decrease. However, we require each node v to use in perpetuity the value of the depth $\Delta(v)$ it estimates in the first round of communication. This ensures that the bound on the number of coding nodes at a particular depth Δ is not violated. Conversely, if nodes leave the network, the actual depth of v may increase. Nonetheless, the bound on the number of coding nodes at a particular depth Δ is still not violated, since the total number of nodes in a network has only decreased. Hence once a coding coefficient is fixed as a polynomial of a particular degree, no matter how many edges or nodes nodes join or leave the network, it does not need to change. This is true for the original coding coefficients in the network, and is also true for coding coefficients corresponding to new edges or nodes inserted into the network.

E. Complexity analysis

The complexity of both **WUP($\epsilon, |\mathcal{T}|$)** and **SUP(ϵ)** scale with the corresponding degree of the polynomials chosen by nodes as coding coefficients.

For **WUP($\epsilon, |\mathcal{T}|$)** , by construction, the degree of the polynomials used as coding coefficients (as noted in [22] is a good proxy for the implementation complexity of codes) scales as the maximum depth of the virtual graph plus $\mathcal{O}(\log(R|\mathcal{T}|/\epsilon))$. But in the virtual graph each original node is replaced with a gadget with depth that is approximately the logarithm of the degree of the node, which in the worst case is at most[9] $c|\mathcal{V}|$. Hence the maximum depth of the virtual graph is at most $\mathcal{O}(\log(c|\mathcal{V}|))$ times the number of vertices in the original graph. Pulling it all together, the complexity of implementation scales as $\mathcal{O}(|\mathcal{V}|\log(c|\mathcal{V}|) + \log(R|\mathcal{T}|/\epsilon))$. Using these bounds, and the bound on the maximum degree of the polynomial used by any node, the end-to-end delay scales as $\mathcal{O}(|\mathcal{V}|^2\log^2(c|\mathcal{V}|)+|\mathcal{V}|\log(c|\mathcal{V}|)\log(R|\mathcal{T}|/\epsilon))$

A similar analysis shows that the complexity of implementation of **SUP(ϵ)** scales as $\mathcal{O}(R|\mathcal{V}|^2\log^2(c|\mathcal{V}|)+|\mathcal{V}|\log^2(c|\mathcal{V}|)\log(1/\epsilon)\})$, and that the redundancy introduced by such codes scales as $\mathcal{O}((R|\mathcal{V}|^3\log^3(c|\mathcal{V}|)+|\mathcal{V}|^2\log^3(c|\mathcal{V}|)\log(1/\epsilon)\}))$.

We now demonstrate that the implementation complexity of **WUP($\epsilon, |\mathcal{T}|$)** and **SUP(ϵ)** is in fact order-optimal by demonstrating a class of networks for why any universal design requires computational complexity which is similar in order of magnitude to that of **WUP($\epsilon, |\mathcal{T}|$)** and **SUP(ϵ)** . Our construction is inspired by that of [11]. Consider any universal ϵ-error linear network code, *i.e.*, any network code that requires that each sink be able to reconstruct the source's information with probability of error at most ϵ, even if the network topology is not known in advance.

Theorem 3. *There exists a class of networks for which the implementation complexity for any ϵ-error universal network code is $\Omega(|\mathcal{E}| - \log(1/\epsilon))$.*

Proof: We construct a single-source multicast network that requires that for any universal code, coding

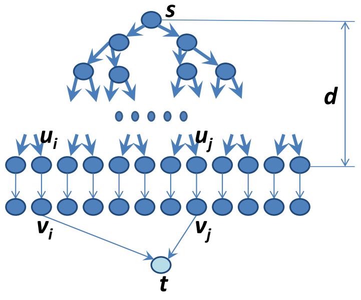


Fig. 4. Example demonstrating that the implementation complexities of **WUP($\epsilon, |\mathcal{T}|$)** and **SUP(ϵ)** are order-optimal.

operations must be chosen from a set that must be exponentially larger than would actually be needed if the topology were known in advance. Consider the "binary-tree-like" network demonstrated in Figure IV-E. The upper part of this graph comprises of a binary tree of depth Δ, with the source s located at the root of the tree, and hence 2^Δ leaf nodes. Each link of this binary tree has capacity 2. Next, each leaf node of this binary tree has a link of unit capacity leaving it to a corresponding *forwarding node*. Finally, we consider two possibilities. Either there are $\binom{2^\Delta}{2}$ sinks, such that each sink is connected to a distinct subset of size two of the set of 2^Δ forwarding nodes, via unit capacity links to each of the two nodes in the subset (this is in the spirit of the *combination networks* examined in [11]). Or, there is only one sink node in the network, which is connected to two of these forwarding nodes (say v_i and v_j) via links of unit capacity each.

As to coding strategies, each node in the binary tree part of the network can forward two linearly independent messages $X_1(z)$ and $X_2(z)$ on each of its outgoing links. Hence at depth Δ each of the 2^Δ

[9]Recall that c denotes the maximum value for any edge's capacity in the network.

leaves of the binary tree have both $X_1(z)$ and $X_2(z)$. Since neither of the two leaf node corresponding to v_i or v_j (say we call them u_i and u_j respectively) is aware of which of the two configurations the network is in, to be universal they must use coding operations that work for both configurations.

First we consider the case of $\epsilon = 0$, *i.e.*, every message must be decoded correctly. Suppose the leaf nodes u_i and u_j choose to transmit the linear combinations $A_1(z)X_1(z)+B_1(z)X_2(z)$ and $A_2(z)X_1(z)+B_2(z)X_2(z)$, or equivalently $X_1(z)+\alpha_1(z)X_2(z)$ and $X_1(z)+\alpha_2(z)X_2(z)$ (by setting $\alpha_i(z) = B_i(z)/A_i(z)$ for $i = 1, 2$). But the messages on each of the forwarding links must be linearly independent (to take into account the eventuality that the network is in the first configuration with $\binom{2^\Delta}{2}$ sinks). Hence there must be at least $2^\Delta - 1$ choices for the $\alpha_i(z)$s (one for each of the leaf nodes, minus one for the case when $A_i(z) = 0$, which can be handled separately.) But if the set of possible coding operations is $\Omega(2^\Delta)$, then its implementation complexity must be at least $\Omega(\Delta)$. But this is $\Omega(|\mathcal{E}|)$. In contrast, if the network happened to be in the configuration with only one sink and this was known in advance, then each of u_i and u_j could simply forward one bit, for an implementation complexity of 1.

The case with ϵ-errors can be similarly analyzed, by allowing sinks to make errors a fraction ϵ of the time. A direct counting argument gives the required result. □

V. Deterministic designs

In this section we describe two deterministic designs of universal distributed robust network codes that are zero-error[10].

Our first scheme is only for codes of rate 2. It is related to a construction of [15], but generalizes it so that the choice of coding operations is independent of the size of the network. We call our scheme the **Rate** 2 **Deterministic Design R2-**D^2 for short. Our purpose in presenting this first scheme is primarily expository, since the proof is significantly easier than that of the second scheme – it helps set the stage for the second scheme.

Our second scheme is for general rates and is independent of all network parameters, including the number of sinks. We call this scheme the **Capacity** 3 **or more, Probability of error** 0 scheme, or **C3-P**0 for short. However, **C3-P**0 is more of an existence result than a practical code since the computational complexity of its implementation is exponential in network parameters.

We first describe some useful preprocessing steps relevant for both of our schemes.

A. Robust distributed unique ID assignment

While the codes in Section IV only required nodes to estimate their depth, the zero-error codes in presented in this section require nodes to obtain a *unique ID*, *i.e.*, an ID that is distinct for each node in the network. Such an ID allows nodes to loosely coordinate coding choices even if they are unable to communicate directly with each other, and thereby ensure that the overall code is “good”. Such IDs might be pre-assigned to nodes (for example via factory stamps, or GPS coordinates, or IP addresses), or be assigned on the fly, as described below.

The task of distributing unique IDs to nodes over a directed graph was considered in [23]. The essential idea of their algorithm is to pretend that the graph is a tree directed from the root to the leaves (if not, extra edges are removed for the ID assignment protocol), and to assign IDs so that the binary expansion of each node’s ID is a prefix to the binary expansion of all nodes downstream from it. This ID distribution can be carried out with communication cost that is asymptotically negligible in the packet length, in conjunction with the normal flow of information through the network, for instance in the header. Here, as in Section IV-A, we need to change the unique ID distribution protocol slightly to make it robust to network changes, so that new nodes are still ensured that IDs assigned to them do not clash with previously assigned IDs. In the same spirit as the robust virtual gadgets in Section IV-A, at each node v

[10]Preliminary versions of the proofs in this section were in the thesis [9].

we reserve a *virtual ID* for the event that a new node might in the future connect to v; if so, this virtual ID is again split into another virtual ID, and an ID that is assigned to the new node.

Precisely, let $\mathcal{I} : \mathcal{V} \to \mathbb{Z}^+$ and $\mathcal{I}_{out} : \mathcal{V} \times \mathcal{V} \to \mathbb{Z}^+$ be the functions defined on the set of network nodes. Our robust unique ID assignment algorithm proceeds as follows:

- Initially the source chooses its own ID $\mathcal{I}(s)$.
- Each node $v \in \mathcal{V}$, waits until it receives at least one message with ID assignment from the nodes upstream. Then its ID $\mathcal{I}(v)$ is set to be equal to $\mathcal{I}_{out}(w, v)$ that it receives first from some adjacent node $w \in \mathcal{V}$ upstream of v.
- Once $\mathcal{I}(v)$ is defined for some $v \in \mathcal{V}$, the node v proceeds as follows. Let n_v be the number of links outgoing from v. Enumerate the links outgoing from v from 1 to n_v. Then, send
$$\mathcal{I}_{out}(v, w_i) = \mathcal{I}(v)2^{\lceil \log_2(n_v+1) \rceil} + i$$
to each node $w_i \in \mathcal{V}$, $i = 1, \dots, n_v$ adjacent to v. Also, each v stores the virtual ID
$$\mathcal{I}_{out}(v, v) = \mathcal{I}(v)2^{\lceil \log_2(n_v+1) \rceil}.$$
- If a new node w joins the network, then $\mathcal{I}(w)$ is chosen to be equal to $\mathcal{I}_{out}(v, v)$ for some v adjacent to w, and the procedure reiterates.

As noted in [23] the worst-case growth rate of the largest node ID with the network size is exponential in $|\mathcal{V}|$, for reasons similar to those outlined in Theorem 3 – nodes might be unable to distinguish between a full binary tree, and a very sparse graph.

B. Cantor labeling

The well-known Cantor diagonal argument [24] makes an unexpected cameo in this work. One version states that the cardinality of the set of integers is the same as that of the set of finite dimensional vectors with integer components, and further gives an effective bijection between the two sets. Further, this bijection guarantees that any vector in $\mathbb{Z}^k$ with maximum component l is mapped to an integer of size $\theta(l^k)$. This mapping is useful since, given a unique ID $\mathcal{I}(v)$ for each node v, we then need to produce unique coding coefficients for each pair of edges i and j such that one is incoming to v and the other is outgoing for v. Prior to code design, the number of such coefficients that each node might need to choose is unknown. However, each $\beta_{u,i,v,j,w}(z)$ coefficient can be labeled by at most the five indices $(\mathcal{I}(u), i, \mathcal{I}(v), j, \mathcal{I}(w))$, each of which is an integer. Hence given a node's unique ID, one can produce unique integral labels for each vector $(\mathcal{I}(u), i, \mathcal{I}(v), j, \mathcal{I}(w))$ that are not too much larger (at most the fifth power) than any of the five parameters in $(\mathcal{I}(u), i, \mathcal{I}(v), j, \mathcal{I}(w))$. This mapping, denoted $K(u, i, v, j, w)$, can then be used to select distinct local coding coefficients as needed in Sections V-C and V-D

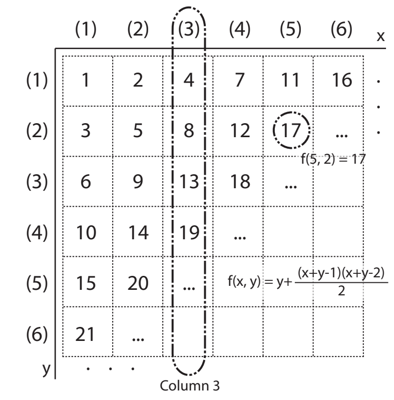


Fig. 5. Example of Cantor-labeling procedure showing the mapping between $\mathbb{Z}$ and $\mathbb{Z}^2$.

C. *Rate 2 zero-error codes*

For the case when the transmission rate equals 2, note that there are essentially just two non-trivial scenarios for each node – either a node receives one linearly independent message on incoming links, or it receives two. In the former case, it can only broadcast incoming information on outgoing links. In the latter case, it can reconstruct the source's information, and thereby can fully control the linear combinations on outgoing links. Our construction for **R2-D**^{2} rests on analysis of these cases.

R2-D^2 (Rate 2 Deterministic Design)

- The source s has two linearly independent messages $X_1(z)$ and $X_2(z)$).
- Depending on its connectivity to the source, on incoming edges each node $v \in \mathcal{V}$ receives either one or two linearly independent combinations of the source messages $(X_1(z), X_2(z))$.
- If a node v receives only one linearly independent message on incoming links, it broadcasts it down all outgoing edges.
- If a node v receives two linearly independent combinations of $(X_1(z), X_2(z))$, this enables it to reconstruct both $X_1(z)$ and $X_2(z)$. For each jth directed edge connecting each pair of nodes v, w (connected possibly by multiple parallel edges), we use the Cantor labeling algorithm in Section V-B to assign a distinct local coding coefficient. In particular, for each 3-tuple $(\mathcal{I}(v), j, \mathcal{I}(w))$ let $K(v, j, w)$ denote the 3-dimensional Cantor mapping. [11] Then the node v then transmits $X_1(z)+\beta_{K(v,j,w)}(z)X_2(z)$ down the jth edge connecting v to w (here $\beta_{K(v,j,w)}(z)$ is chosen to be distinct for each $K(v, j, w)$).

Theorem 4. *For any network $\mathcal{G}$ with min-cut capacity at least 2,* **R2-D**^{2} *succeeds with zero error.*

Proof: For any $v \in \mathcal{V}$ such that the mincut between the source and v is at least 2, there are at least two edge-disjoint paths from the source to v. By the statement of our **R2-D**^{2} algorithm, for any such nodes v and v', the linear combinations of $X_1(z)$ and $X_2(z)$ on all their outgoing links must be distinct, and linearly independent (since the vectors $(1, \beta_{K(v,j,w)}(z))$ and $(1, \beta_{K(v',j',w')}(z))$ are linearly independent if and only if $\beta_{K(v,j,w)}(z)$ and $\beta_{K(v',j',w')}(z)$ are distinct). $\square$

D. *General zero-error codes*

The challenge in extending the results of Section V-C to rates greater than 2 lies in the fact that there might be nodes receiving two or more linearly independent pieces of information, and yet are unable to decode the source messages. In this case, they do not have full control over the messages they are able to send out, and hence the argument of Theorem 4 fails. In this section, we get around this challenge by examining a different invariant of linear convolutional network codes. In particular, we choose coding coefficients in a distributed manner so that the delay of the source messages on *every* path in the network is distinct. This means that the source messages never cancel out at the sinks, and hence can be reconstructed.

C3-P0 (Capacity 3 or more, Probability of error 0) codes

- For each 5-tuple $(\mathcal{I}(u), i, \mathcal{I}(v), j, \mathcal{I}(w))$, let $K(u, i, v, j, w)$ be the 5-dimensional Cantor mapping defined in Section V-B. We define the local coding coefficient $\beta_{u,i,v,j,w}(z)$ as

$$z^{2^{K(u,i,v,j,w)}},$$

i.e., the monomial in z with degree $exp(K(u, i, v, j, w))$ (here the exponent is base 2).

Theorem 5. *For any network $\mathcal{G}$,* **C3-P**0 *succeeds with zero error.*

Proof: Theorem 4 in [1] demonstrates that $|T_t|$, the determinant of the transfer matrix T_t from the source to any sink t, can be written as $\Sigma c_{\mathcal{P}} \Pi_{\mathcal{P}} \beta_{u,i,v,j,w}(z)$, where the product is over all the local coding coefficients on a particular path $\mathcal{P}$ from s to t, $c_{\mathcal{P}}$ is a non-zero constant corresponding to $\mathcal{P}$, and the outer summation

[11] In Section V-B we assume that u and i are also variable, but in this section they are fixed, hence we only need to consider 3-dimensional mappings.

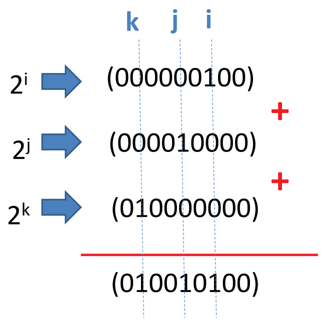


Fig. 6. Graphical representation of the proof of correctness of **C3-P**0 codes.

is over all paths from s to t. Our choice of local coding coefficients along any path in **C3-P**0 implies that $|T_t|$ equals

$$\sum c_{\mathcal{P}} \prod_{\mathcal{P}} z^{exp(K(u,i,v,j,w))} = \sum c_{\mathcal{P}} z^{(\sum_{\mathcal{P}} exp(K(u,i,v,j,w)))}. \tag{2}$$

But by choice, each of the terms $K(u, i, v, j, w)$ is distinct, and hence the binary expansion of $exp(K(u, i, v, j, w))$ has a single 1 in a distinct location. But if two paths in the summation (2) differ, then they must differ in at least one of the local coding coefficients, and therefore the exponent of the power of z along the two paths must differ – hence each path corresponds to a distinct power of z. This implies that as long as there is at least one path from s to each $t \in \mathcal{T}$, each of the corresponding transforms T_t must be invertible. □

E. Complexity Analysis

The complexity of both **R2-D**^{2} and **C3-P**0 scale with the corresponding Cantor labeling and node assignments.

For **R2-D**^{2} the size of the set any node chooses its coding coefficients from scales as the third power of the largest node ID or the largest link-capacity in the network. But as noted in Section V-B, the largest node ID can scale exponentially in $|\mathcal{V}|$. Hence the degree of the polynomials used as coding coefficients scales logarithmically in the size of the sets from which local coding coefficients are chosen, which in turn scales as $\mathcal{O}(\max\{|\mathcal{V}|, \log^3(c)\})$. The corresponding redundancy the network introduces in the codes, arising from the delays introduced by each coding node, then scales as $\mathcal{O}(|\mathcal{V}| \max\{|\mathcal{V}|, \log^3(c)\})$, since each coding node in a path can introduce at most the maximal delay and delays along a path add up.

A similar analysis shows that the complexity of implementation of **C3-P**0 scales as $\mathcal{O}(exp(\max\{|\mathcal{V}|, \log^5(c)\}))$, and that the redundancy introduced by such codes scales as $\mathcal{O}(|\mathcal{V}|exp(\max\{|\mathcal{V}|, \log^5(c)\}))$.

The problem of polynomial identity testing (PIT) [25] examines the question of deterministically determining whether a polynomial with a succinct but non-standard representation (such as the determinant of a matrix of polynomials) identically equals zero. The deterministic complexity of such problems is a long-standing open problem in theoretical computer science. Given this context, we are unable to provide intuition on whether our codes in Section V-D have order-optimal computational complexity – indeed, answering this question in either direction would represent significant progress in resolving the complexity of PIT problems.

VI. Implementation issues

As noted in [22], the complexity of implementation of network codes scales polynomially in the logarithm of the field-size over which operations are performed, or in the case of convolutional network codes, polynomially in the degree of the polynomials used at each node. By this measure, the implementation complexity of the codes in [1], [8] is poly-logarithmic in network parameters, whereas the implementation complexity of the first three of the four codes in this work is polynomial in network parameters. While this is an exponential blow-up, we note that the resulting codes are still computationally tractable, and further, as noted in Section IV-E, such a blow-up is in fact necessary for codes to be universal.

Code	$\|\mathcal{T}\|$ known	$\|\mathcal{E}\|$ known	Error Probability	Implementation Complexity
[1]	yes	yes	ϵ	$\mathcal{O}(polylog(\|\mathcal{E}\|\|\mathcal{T}\|))$
[7]	yes	yes	0	$\mathcal{O}(polylog(\|\mathcal{T}\|))$
WUP($\epsilon, \|\mathcal{T}\|$)	yes	no	ϵ	$\mathcal{O}(poly(\|\mathcal{V}\|\log(c)) + \log(R\|\mathcal{T}\|/\epsilon))$
SUP(ϵ)	no	no	ϵ	$\mathcal{O}(poly(R\|\mathcal{V}\|\log(c)) + \log(1/\epsilon))$
R2-D^2	no	no	0	$\mathcal{O}(\max\{\|\mathcal{V}\|, \log^3(c)\})$
C3-P0	no	no	0	$\mathcal{O}(exp(\max\{\|\mathcal{V}\|, \log^5(c)\}))$

TABLE I
A SUMMARY OF THE PROPERTIES OF OUR CONSTRUCTIONS, AND COMPARISON BETWEEN THEM AND PRIOR NON-UNIVERSAL ALGORITHMS.

While the schemes in this work have been presented in the context of convolutional network coding operations at each node, they also go through for other infinite fields such as $\mathbb{Q}$ – the only requirement is that the field be unbounded in size, and that an infinite subset of it have a succinct representation.

Also, despite presenting all messages at the source and each link as bit-streams of possibly unbounded length, the schemes described in prior sections can also be implemented by packetization, by chopping up the bit-streams into packets of a standard size n.

In our codes the header of each packet contains low-rate control information used by each node to decide on its coding operations. However, by design, the size of this header changes as information flows down the network – the rate of change depends on the network topology, and hence is unpredictable in advance. One challenge in the implementation of our codes is thus to ensure that the intermediate nodes are able to distinguish between header information and payload information. One standard trick for such scenarios is used in Theorem $14.2.3$ of [26] – each bit of the header is doubled, and the final such double-bit is followed by a 01 to signify the end of the header. Since the length of the header is asymptotically negligible in the packet-size, the communication cost of this bit-doubling is still asymptotically negligible.

VII. Discussion

In this work we provide the first rate-optimal network code designs that have guaranteed decodability performance, and yet are independent of all network parameters. While requiring such universality makes us pay a price in the computational complexity and redundancy, (all but one of) our codes are computationally efficient to implement. The analytical tools we derive may well be of independent interest.

References

[1] T. Ho., R. Kötter, M. Médard, D. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” in *IEEE International Symposium on Information Theory (ISIT)*, Yokohama, July 2003, p. 442.

[2] J. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[4] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear network coding,” *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[5] R. Kötter and M. Médard, “Beyond routing: An algebraic approach to network coding,” in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 1, 2002, pp. 122–130.

[6] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, “Polynomial time algorithms for multicast network code construction,” *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, June 2005.

[7] M. Langberg, A. Sprintson, and S. Bruck, “The encoding complexity of network coding,” in *International Symposium on Information Theory*, Sept. 2005.

[8] S. Jaggi, P. A. Chou, and K. Jain, "Low complexity algebraic multicast network codes," in *IEEE International Symposium on Information Theory (ISIT)*, Yokohama, July 2003, p. 368.

[9] S. Jaggi, "Design and analysis of network codes," Dissertation, California Institute of Technology, 2006.

[10] S. Jaggi, T. Ho, and M. Effros, "Zero-error distributed network codes," in *Information Theory and Applications Workshop (unpublished)*, UCSD, San Diego, CA, Jan 2007.

[11] A. Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proceedings of SODA*, 2004.

[12] J. Rissanen, "A universal data compression system," *IEEE Transactions on Information Theory*, vol. 29, no. 5, pp. 656–664, Sep. 1983.

[13] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems.* Akadémiai Kiadó, 1981.

[14] A. F. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros, "Capacity of wireless erasure networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 789–804, Mar. 2006.

[15] C. Fragouli and E. Soljanin, "Decentralized network coding," in *Information Theory Workshop*, San Antonio, TX, USA, 2004, pp. 310–314.

[16] L. Xia, S. Vyetrenko, S. Jaggi, and T. Ho, "Universal and robust distributed network codes," in *Proceeding of the 30th IEEE Conference on Computer Communications (INFOCOM)*, Shanghai, China, 2011.

[17] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[18] E. Erez and M. Feder, "Convolutional network codes," in *IEEE International Symposium on Information Theory*, 2004.

[19] N. Alon, "Combinatorial nullstellensatz," *Combinatorics, Probability and Computing*, vol. 8, pp. 7–29, 1999.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 2nd Edition.* MIT Press and McGraw-Hill, 2001.

[21] S. Jaggi, Y. Cassuto, and M. Effros, "Low complexity encoding for network codes," in *Proc. of the International Symposium on Information Theory*, Seattle, WA, USA, Sep 2006.

[22] S. Jaggi, M. Effros, T. Ho, and M. Médard, "On linear network coding," in *Proceedings of 42nd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2004.

[23] J. Bruck, M. Langberg, and M. Schwartz, "Distributed broadcasting and mapping protocols in directed anonymous networks," in *The proceedings of the Twenty-Sixth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, 2007, pp. 383–383.

[24] G. Cantor, "Eigenschaft des inbegriffes aller reelen algebraischen zahlen," *Crelles Journal*, vol. 77, pp. 258–262, 1874.

[25] M. Agrawal and R. Saptharishi, "Classifying polynomials and identitytesting," *Current Trends in Science, 2009*, 2009.

[26] T. Cover and J. Thomas, *Elements of Information Theory.* John Wiley and Sons, 2nd Edition, 2006.

[27] M. Mitzenmacher and E. Upfal, *Probability and Computing : Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, New York (NY), 2005.

APPENDIX

A. *Proof of Lemma 2*

As in the proof of the original Schwartz-Zippel lemma [2], we proceed by mathematical induction. In the base case when $N = 1$, Lemma 2 is equivalent to the Schwartz-Zippel lemma in one variable. As the inductive hypothesis, suppose that Lemma 2 is true for $N-1$ variables in the polynomial $P(.)$. Now consider the case when the polynomial $P(x_1, x_2, \ldots, x_N)$ has N variables. The polynomial can be rewritten so that

$$P(x_1, x_2, \ldots, x_N) = x_N^{d_N} P_1(x_1, x_2, \ldots, x_{N-1}) + R_1(x_1, x_2, \ldots, x_N) \tag{3}$$

for some polynomials $P_1(.)$ and $R_1(.)$ over the appropriate variables.

The probability that $P(.)$ equals zero can be bounded from above by

$$\begin{aligned}\Pr[P(.)=0] &= \Pr[P(.)=0, P_1(.)=0] + \Pr[P(.)=0, P_1(.)\neq 0]\\ &= \Pr[P_1(.)=0]\Pr[P(.)=0|P_1(.)=0]\\ &\quad+\Pr[P_1(.)\neq 0]\Pr[P(.)=0|P_1(.)\neq 0]\\ &\leq \Pr[P_1(.)=0]+\Pr[P(.)=0|P_1(.)\neq 0] \end{aligned}\tag{4}$$

But by the inductive hypothesis

$$\Pr[P_1(.)=0] \leq \sum_{i=1}^{N-1}\frac{d_i}{|S_i|}. \tag{5}$$

Also, by the Principle of Deferred Decisions [27] the probability $\Pr(P(.)=0)$ is unaffected if the value of x_N is chosen after the values of all the other variables have been fixed. In this case, if $P_1(.)\neq 0$, then $P(.)$ is a polynomial of degree d_N over x_N. By the Schwartz-Zippel lemma

$$\Pr[P(.)=0|P_1(.)\neq 0] \leq \frac{d_N}{|S_N|}. \tag{6}$$

Substituting (5) and (6) into (4) gives the required result. [12] $\square$

[12]One way to strengthen this result is to note that one can optimize over the sequence in which we reduce one variable at a time in (3). For instance, if $P(x_1, x_2) = x_1^2 x_2 + x_2^{100} x_1$, our worst-case bound tells us that $\Pr[P(.) = 0] \leq \frac{2}{|S_1|} + \frac{100}{|S_2|}$. However, if we choose $P_1(x_1) = x_1$, $R(x_1, x_2) = x_1^2 x_2$, then examining (6) tells us that $\Pr[P(.) = 0] \leq \frac{1}{|S_1|} + (\frac{2}{|S_1|} + \frac{1}{|S_2|})$, which might be substantially smaller for some values of $|S_1|/|S_2|$. Nonetheless, in the interest of simplicity, we state and only the worst-case result presented in Lemma 2, since it is a strong enough tool for this work.