Low Complexity Encoding for Network Codes

Sidharth Jaggi[1]
Laboratory of Information and Decision Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
Email: jaggi@mit.edu

Yuval Cassuto[2], Michelle Effros
Department of Electrical Engineering
California Institute of Technology
Pasadena, CA 91125, USA
Email: {ycassuto,effros}@caltech.edu

***Abstract*— In this paper we consider the per-node run-time complexity of network multicast codes. We show that the randomized algebraic network code design algorithms described extensively in the literature result in codes that on average require a number of operations that scales quadratically with the block-length m of the codes. We then propose an alternative type of linear network code whose complexity scales linearly in m and still enjoys the attractive properties of random algebraic network codes. We also show that these codes are optimal in the sense that any rate-optimal linear network code must have at least a linear scaling in run-time complexity.**

I. Introduction

In the *multicast* problem, a specified set of sink nodes must reproduce without error all of the information generated at the network's source node. The work of [1] provides a tight characterization of the rate-region under the assumption that nodes in the network are allowed to perform arbitrary operations on the values from incoming edges to generate messages on outgoing edges. The results of [10] draw further interest by proving that linear codes are sufficient to achieve capacity, and [7] gives an elegant (though exponential-time) code construction. Linear codes are attractive for their simplicity on two levels. First, linear codes have low run-time complexity; by definition, the complexity of the circuit required to generate a single output bit at any node is at most linear in the effective block-length over which the code operates. This allows for the possibility of practical implementation. Second, considerable work has already been done in the design and analysis of linear codes, which provides a rich set of theoretical and practical tools to build on.

Work by [5] and especially [3] demonstrates that even if individual nodes choose their own coding operations independently and uniformly at random, with high probability the resulting code achieves the network multicast capacity. Work by [6] demonstrates a polynomial-time construction of network codes for the multicasting problem. These results demonstrate the existence of codes that are not just fast at run-time, but are also tractable to design. The network codes described in [3] are particularly attractive due to the distributed nature of their design, which means that such codes can be used with minimal oversight.

Other work aimed at simplifying network codes includes [8], which demonstrates code constructions that bound the number of nodes that have to perform non-trivial coding operations, and [2], which provides codes that minimize the number of such nodes. A similar encoding complexity investigation for the special case of line networks with erasures, appears in [12].

In our work, we focus on minimizing the run-time complexity at individual nodes. We first show that the randomized design of a block-length m code suggested in [3] requires on average $\mathcal{O}(m^2)$ arithmetic operations to generate a single length-m bit-vector on any outgoing edge. (In the asymptotic analysis m grows to infinity and network parameters such as the edge and vertex counts are regarded as constants.) We demonstrate that any linear network code that achieves the network multicast capacity has a run-time complexity of implementation that scales at least linearly with m. In that context, we propose a distributed randomized design of network multicast codes called *permute-and-add network codes*, with run-time coding complexity $\mathcal{O}(m)$ at source and internal nodes. Our central result is a proof that the given codes achieve multicast capacity with probability that tends to 1 with growing m, memory requirements that grow linearly in m, and asymptotically negligible rate-loss. Thus permute-and-add network codes achieve essentially the same performance as algebraic network codes while meeting the complexity lower bound. The decoding complexity at receiver nodes remains unchanged from earlier algorithms ($\mathcal{O}(m^2)$).

II. Definitions

A. Network Model

We use essentially the same network model as in [7]. Let $\mathcal{V}$ be a set of vertices, and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}\times\mathbb{Z}$ be a set of unit-capacity directed edges, where $e = (v, v', i) \in \mathcal{E}$ denotes the ith edge from v to v'. The tuple $(\mathcal{V}, \mathcal{E})$ defines a directed graph $\mathcal{G}$.

For a node $v \in \mathcal{V}$, let $\Gamma_O(v)$ denote the set of the edges (v, v', i) outgoing from v and $\Gamma_I(v)$ denote the set of edges (v'', v, i) entering v. An edge $e = (v, v', i)$ has *tail* $v = v_t(e)$, and *head* $v' = v_h(e)$.

An ordered set $\{u_1, i_1, u_2, i_2, \ldots, i_{n-1}, u_n\}$ is a *path* $P(u_1, u_n)$ *from* u_1 *to* u_n *in* $\mathcal{G}$ if $(u_j, u_{j+1}, i_j) \in \mathcal{E}$ for all $j \in \{1, \ldots, n-1\}$. Two paths P and P' are *edge-disjoint* if they do not share edges in common. A path $P(u, v)$ is a *cycle* if $u = v$. A graph $\mathcal{G}$ is *acyclic* if it contains no cycles. We here restrict our attention to the case of acyclic networks; our results can be generalized to networks with cycles using delays as proposed in [1], and demonstrated, for instance, in [4].

[1]Supported by AFOSR FA9550-06-1-0155
[2]Supported by the Lee Center for Advanced Networking and by NSF grant ANI-0322475

For any $S \subseteq \mathcal{V}$, a *cut* $\mathrm{cut}(S) \subseteq \mathcal{E}$ is the set of all edges $(v, v', i) \in \mathcal{E}$ such that $v \in S$, $v' \in \mathcal{V} \setminus S$. The *value of* $\mathrm{cut}(S)$, written $|\mathrm{cut}(S)|$ equals the size of $\mathrm{cut}(S)$. A *min-cut from* v *to* v' $\mathrm{mincut}(v, v')$ is any $\mathrm{cut}(S)$ of minimal size such that $v \in S$ and $v' \notin S$. The *value of* $\mathrm{mincut}(v, v')$, $|\mathrm{mincut}(v, v')|$ equals the size of $\mathrm{mincut}(v, v')$.

Vertex-set $\mathcal{V}$ contains a pre-specified *source vertex* s and a pre-specified set $\mathcal{T}$ of *sink vertices*. Time is discrete and indexed by non-negative integers. Let $\mathcal{X} = \{0, 1\}$ and let a code design parameter m specify the *block-length*. Block-length m and *rate* R are assumed to be such that both m and the product mR are integers. Over coding interval i, s generates i.i.d. Bernoulli-$(1/2)$ random bits at a rate mR, $\{X_{i,j}\}_{j=1}^{mR} \in \mathcal{X}^{mR}$. These need to be reproduced exactly at each sink node $t \in \mathcal{T}$. The source s possesses a *source encoder*, and each $t \in \mathcal{T}$ possesses a *sink decoder*. Every other node in $\mathcal{V}$ possesses an *internal encoder*. A *multicast network code* $\mathcal{C}$ is defined by its source encoders, internal encoders, and decoders at receiver nodes.

B. Network Codes

We now define *block-linear network codes*. The network $\mathcal{G}$, block-length m and rate R are design parameters for the network code $\mathcal{C}(\mathcal{G}, m, R)$, henceforth denoted simply by $\mathcal{C}$. We block source bits into an mR-dimensional vector over $\mathbb{F}_2$. In particular, $(X_{i,1}, X_{i,2}, \ldots, X_{i,mR})^T$ is defined as $X_i \in (\mathbb{F}_2)^{mR}$. Since our code is time-invariant, we henceforth drop all subscripts indexing time intervals from our notation; thus X_i is denoted by X.

Let Y^e be the length-m vector transmitted across edge e, defined inductively as follows.

The source encoder comprises a collection of functions $\{f^e\}_{e \in \Gamma_O(s)}$. For each $e \in \Gamma_O(s)$, $f^e : (\mathbb{F}_2)^{mR} \to (\mathbb{F}_2)^m$ is a linear map over $\mathbb{F}_2$ from X to Y^e. In particular the source encoder for any edge e generates Y^e as $[\beta^e]X$, where $[\beta^e]$ is an $m \times mR$ matrix over $\mathbb{F}_2$. (In our notation all matrices are enclosed in square braces [.].)

For each $e \notin \Gamma_O(s)$ the internal encoder $f^e : (\mathbb{F}_2)^{m|\Gamma_I(v_t(e))|} \to (\mathbb{F}_2)^m$ is a linear map over $\mathbb{F}_2$ taking messages $\{Y^{e'}\}_{e' \in \Gamma_I(v_t(e))}$ on edges e' incoming to the tail of e to the message-vector Y^e. In particular the internal encoder for any edge e generates Y^e as $\Sigma_{e' \in \Gamma_I(v_t(e))}[\beta^{e',e}]Y^{e'}$, where each $[\beta^{e',e}]$ is an $m \times m$ matrix over $\mathbb{F}_2$.

The encoders inductively define the *global coding matrix* for each edge $e \in \mathcal{E}$, i.e., the linear transformation of the information from the source to the bit-vectors that traverse edge e. Thus each edge $e \in \mathcal{E}$ carries the length-m vector $Y^e = [\beta^e]X$, where $[\beta^e]$ is the $m \times mR$ global coding matrix.

For each $t \in \mathcal{T}$, the sink decoder $h^t : (\mathbb{F}_2)^{m|\Gamma_I(t)|} \to (\mathbb{F}_2)^{mR}$ is a linear map over $\mathbb{F}_2$ that takes the collection $Y^t = (Y^e)_{e \in \Gamma_I(t)}$ of received channel outputs to a reconstruction $\hat{X}^t$ of message X. In particular, each decoder at sink t generates $\hat{X}^t$ as $\Sigma_{e' \in \Gamma_I(t)}[\beta^{e'}]Y^{e'}$, where each $[\beta^{e'}]$ is an $mR \times m$ matrix over $\mathbb{F}_2$.

A network code $\mathcal{C}$ achieves rate R if for each transmitted message X and each sink $t \in \mathcal{T}$, $\hat{X}^t = X$. The work of [1] demonstrates that the maximal achievable rate for any network code, the network *multicast capacity*, is $C = \min_{t \in \mathcal{T}} |\mathrm{mincut}(s, t)|$.

The block network codes given here differ from the algebraic network codes introduced in [7]. In an algebraic network code, each X and Y^e is viewed as an element of the finite-field $\mathbb{F}_q$, with $q = 2^m$. The encoding and decoding functions at each node are linear functions over $\mathbb{F}_q$, so that the length-m symbol on any edge e is of the form $\Sigma_{e' \in \Gamma_I(v_t(e))}\beta^{e',e}Y^{e'}$, where each $\beta^{e',e}$ is an element of $\mathbb{F}_q$. Algebraic linear network codes are a special case of block network codes since for every finite field $\mathbb{F}_{2^m}$, a set of 2^m matrices of size $m \times m$ (called *representative matrices*) can be found such that multiplication in $\mathbb{F}_{2^m}$ is equivalent to a matrix-vector product over $\mathbb{F}_2$ [11]. In our new *permute-and-add linear network codes*, each $[\beta^{e',e}]$ is an $m \times m$ *permutation matrix*, i.e., a binary matrix with one 1 in each column and each row, and zeros everywhere else. For such network codes, the operation $\Sigma_{e' \in \Gamma_I(v_t(e))}[\beta^{e',e}]Y^{e'}$ is equivalent to permuting each length-m vector $Y^{e'}$ of bits incoming to $v_t(e)$ and then summing the permuted vectors over $\mathbb{F}_2$ to generate the length-m vector Y^e transmitted on edge e.

III. Bounds on encoding complexity

In algebraic linear network codes, the encoding for every edge is done via a summation of finite field products. In this section we show that these finite field products, when viewed as matrix-vector products, require representative matrices that are dense. This is true regardless of the particular set of matrices one may choose for these products. Hence algebraic codes may be undesirable when the network nodes are computationally limited. We now give a proof for fields of characteristic 2.[1] The node $v = v_h(e')$ multiplies $Y^{e'}$ by a finite field constant β and adds the result, a symbol $(\beta Y^{e'}) \in \mathbb{F}_q$, to the corresponding values from all $e \in \Gamma_I(v)$. We avoid discussion of "good" ways to implement finite field multiplications since implementation choices may be hardware dependent. We assume only that the transformation from $Y^{e'}$ to $(\beta Y^{e'})$ is done by a matrix vector multiplication: $(\beta Y^{e'}) = [M_\beta](Y_1^{e'}, \ldots, Y_m^{e'})^T$, where $[M_\beta]$ is an $m \times m$ matrix over $\mathbb{F}_2$ created by mapping each of the 2^m elements of $\mathbb{F}_q$ to a unique $m \times m$ matrix over $\mathbb{F}_2$. We show that every set of representative matrices has to be "dense" (have $\Omega(m^2)$ non-zero elements) on average. Let α be a primitive element in $\mathbb{F}_q$ and let $\{\mathbf{0}, \boldsymbol{\alpha}^{\mathbf{0}}, \ldots, \boldsymbol{\alpha}^{\boldsymbol{q-2}}\}$ be the q vectors from $(\mathbb{F}_2)^m$ that represent the elements of $\mathbb{F}_q$, with respect to some basis over $\mathbb{F}_2$ (symbols in bold-face are used to distinguish the vector representations from the abstract representations of the field elements).

Theorem 1: If $\{[M_0], [M_{\alpha^0}], \ldots, [M_{\alpha^{q-2}}]\}$ are $m \times m$ matrices over $\mathbb{F}_2$ such that $[M_{\alpha^i}]\boldsymbol{\alpha^j} = \boldsymbol{\alpha^{i+j}}$, and if $D([M])$ is the number of nonzero elements in matrix $[M]$, then

$$\frac{1}{q}[D([M_0]) + \sum_{j=0}^{q-2} D([M_{\alpha^j}])] = \frac{m^2}{2}.$$

[1] While this proof is sufficient for our purposes, the proof extends directly to the general case of $\mathbb{F}_q$, for every prime power q

Proof: For any element $\alpha^i \in \mathbb{F}_q$ and for each element $\alpha^j \in \mathbb{F}_q$, there exists a distinct matrix $[M_{\alpha^{j-i}}]$ such that $[M_{\alpha^{j-i}}]\boldsymbol{\alpha^i} = \boldsymbol{\alpha^j}$. Let α^i be the element represented by the vector with a one in location $l(i)$ and zeros elsewhere. Then the $l(i)$th column of the matrix $[M_{\alpha^{j-i}}]$ must have a Hamming weight (number of nonzero locations, denoted $w_H(\cdot)$) that equals $w_H(\boldsymbol{\alpha^j})$. Averaging $w_H(\boldsymbol{\alpha^j})$ over all j and over $\mathbf{0}$, gives $m/2$. Thus, the average weight of the $l(i)$th column equals $m/2$. Repeating the argument for all $l(i) \in \{1, 2, \ldots, m\}$ and summing the column weights gives the desired result. □

This theorem allows us to bound from below the number of matrices that are dense in any representation.

Corollary 2: At least half of the elements in $\mathbb{F}_q$ have representative matrices with at least $m^2/2$ non-zero elements.

We now show a simple network that provides a lower bound on the number of operations required per block by rate-optimal linear network codes.

Theorem 3: For each C and $m \in \mathbb{Z}$, there exists a network $\mathcal{G}_{C,m}$ with $\mathrm{mincut}(s,t) = C$ such that for any network code $\mathcal{C}$ with at most $m' < m$ arithmetic operations per block, the maximal achievable multicast rate is $\frac{m'}{m}C$.

Proof: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the network with $\mathcal{V} = \{s, u, t\}$ and $\mathcal{E} = \{e, e'\}$ such that $e = (s, u, 1)$ and $e' = (u, t, 1)$. The min-cut from the source to the receiver equals 1. This network can be easily extended to any min-cut C. By assumption the $m \times m$ matrix $[\beta^{e,e'}]$ has at most m' non-zero elements, which implies that its rank is at most m'. Therefore the maximal achievable rate equals $\frac{m'}{m}C$. □

IV. Permute-and-add codes

We design the permute-and-add network code $\mathcal{C}_\pi$ as follows. Let network $\mathcal{G}$ have multicast capacity C. Assume without loss of generality that $|\Gamma_O(s)| = C$; if $|\Gamma_O(s)| > C$, we add a super-source node s' and connect it to s with C parallel edges. Let $\sigma_{\pi,m}$ be the uniform distribution on the set of all $m \times m$ permutation matrices. For all internal encoders we choose $[\beta^{e',e}]$ i.i.d. from $\sigma_{\pi,m}$. The source encoders f^e for $e \in \Gamma_O(s)$ are aggregated in a single $mC \times mR$ matrix $[A]$. Matrix $[A]$ is the product $[G][A']$, where $[A']$ is a random $mC \times mR$ binary matrix and $[G]$ is the invertible $mC \times mC$ matrix that performs Gaussian elimination on the rows of $[A']$.

The following lemma about random permutation matrices is useful in proving our main result.

Lemma 4: For $[\beta]$ chosen from $\sigma_{\pi,m}$, arbitrary $m \times m$ matrix $[L]$, and any $\epsilon > 0$, the probability that the rank of $[L] + [\beta]$ is less than $m(1-\epsilon)$ is at most $2^{-m\epsilon + \log(m+1)}$.

Proof: For a fixed, length-m vector V, the probability over $\sigma_{\pi,m}$ that V is in the null-space of $[L] + [\beta]$ is

$$\begin{aligned}
&\mathbf{Pr}_{\sigma_{\pi,m}}[([L]+[\beta])V = 0] \\
&= \mathbf{Pr}_{\sigma_{\pi,m}}[V' = [\beta]V] \\
&= \begin{cases} \frac{1}{\binom{m}{w_H(V)}}, & w_H(V') = w_H(V) \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

where $V' = [L]V$. The last equality follows because $[\beta]$ acts as a random permutation on V and therefore $V' = [\beta]V$ can only occur if V and V' have the same Hamming weight and then only if the random permutation maps the non-zero locations of V to the non-zero locations of V'. Now let V be chosen uniformly at random from σ_m, the uniform distribution on length-m binary vectors. Then the probability over σ_m and $\sigma_{\pi,m}$ that V is in the null-space of $[L] + [\beta]$ is

$$\begin{aligned}
&\mathbf{Pr}_{[\sigma_{\pi,m},\sigma_m]}[([L]+[\beta])V = 0] \\
&\leq \frac{1}{2^m}\sum_V \frac{1}{\binom{m}{w_H(V)}} \\
&= \frac{1}{2^m}\sum_{w=0}^{m}\binom{m}{w}\frac{1}{\binom{m}{w}} \qquad (1) \\
&= \frac{m+1}{2^m}.
\end{aligned}$$

Equation 1 results from the partitioning of the set of all length-m vectors V into $m+1$ weight classes. Since both the permutation matrix $[\beta]$ and the vector V are drawn uniformly from their corresponding domains (of size $m!$ and 2^m respectively), the number of $(V, [\beta])$ pairs satisfying $V' = [\beta]V$ is bounded from above by

$$m!2^m\frac{m+1}{2^m} = (m+1)!$$

Thus the number of $[\beta]$ matrices for which $V' = [\beta]V$ for at least $2^{m\epsilon}$ distinct vectors V is no greater than $(m+1)!/2^{m\epsilon}$. Since there are $m!$ possible $[\beta]$ matrices in total, and $\sigma_{\pi,m}$ is uniform by assumption, the desired probability is bounded by $[(m+1)!/2^{m\epsilon}]/m! = (m+1)/2^{m\epsilon} = 2^{-m\epsilon+\log(m+1)}$. □

Let ϵ_m, a parameter in the design of $\mathcal{C}_\pi$, be any function of m such that $\lim_{m\to\infty}\epsilon_m = 0$. Define $\epsilon'_m = \epsilon_m - \frac{\log(m+1)}{m}$ and require $\lim_{m\to\infty} m\epsilon'_m = \infty$. We are now in a position to state and prove our main result.

Theorem 5: For any $\epsilon_m > 0$, network code $\mathcal{C}_\pi$ achieves rate $R = C - (|\mathcal{E}|+1)\epsilon_m$ with probability greater than $1 - 2^{-m\epsilon'_m + \log[|\mathcal{T}|(|\mathcal{E}|+C)]}$.

Proof: We first present a high-level outline of the proof. We need to show that the transfer matrix from the source to any sink is invertible with high probability. To prove this, we use first the directed, acyclic nature of $\mathcal{G}$ to define a partial order on specific cut-sets of $\mathcal{G}$, and then analyze the transfer matrices between vectors carried by successive cut-sets (for now ignoring the source encoder). We show that, with high probability, the rank of the linear transformation between any two successive cut-sets is nearly full, i.e., that almost all of the information carried by edges in one cut-set is retrievable from edges in the succeeding cut-set. We use the union bound to bound, by a function decaying in m, the probability that any linear map between successive cut-sets results in a rank-loss that is not asymptotically negligible. We then note that the composition of linear maps that are all almost full-rank results in a linear map that is also almost full-rank. Therefore, the transfer-function to each receiver is, with high probability, almost full-rank. Lastly, we show that the span of vectors injected into the network by a random (and sparse) source

encoding function does not intersect the null-space of the transfer function to any receiver.

We now define a number of variables and formalize the above outline. Let the *flow* $F^t = \cup_{j=1}^{C}\{e \in \mathcal{E} : e \text{ is on path } P_j(s,t)\}$ be the edges in a set of C edge-disjoint paths from s to any sink t. The *network flow* $F^{\mathcal{T}}$ *between* s *and* $\mathcal{T}$ is the union of the edges in individual flows, i.e., $F^{\mathcal{T}} = \cup_{t\in\mathcal{T}} F^t$. We do not assume that the network nodes know the flows or any other information about the topology of the network (and indeed the random scheme works without such knowledge) but for the purpose of analysis we only consider information flowing on edges in $F^{\mathcal{T}}$, rather than possibly all edges in $\mathcal{E}$. The edges in $F^{\mathcal{T}}$ are numbered from 1 to $|F^{\mathcal{T}}|$ in a manner consistent with the partial order on $e \in \mathcal{E}$ (i.e., for any edges e' and e, $v_h(e') = v_t(e)$ implies $e' < e$.) Let a be a step-counter. Counter a, which keeps track of the stage of our inductive proof-checking algorithm, is initialized to 0. In step $a > 0$, our proof-checking algorithm examines edge $e(a)$, which is the ath edge of $F^{\mathcal{T}}$ according to the partial order on $\mathcal{E}$.

A frontier edge set for sink t at step a is an ordered subset $\mathcal{F}^{t,a} \subseteq F^t$ containing C edges and inductively defined as follows. For each $t \in \mathcal{T}$, $\mathcal{F}^{t,0}$ is the set containing the first edge on each path $P_j(s,t)$; thus $\mathcal{F}^{t,0} = \Gamma_O(s)$. The *frontier edge set matrix* for sink t at step a is the $mC \times mC$ transfer matrix $[B^{t,a}]$ from the vectors carried by the C edges in $\Gamma_O(s)$ to the vectors carried by the C edges of $\mathcal{F}^{t,a}$. In step a, the proof-checking procedure checks for each $t \in \mathcal{T}$, $j \in \{1,\ldots,C\}$ whether the jth element of $\mathcal{F}^{t,a-1}$ is the immediate predecessor of $e(a)$ in $P_j(s,t)$. If so, this edge is replaced by $e(a)$ to obtain the updated frontier edge set $\mathcal{F}^{t,a}$. This inductive definition of frontier edge-sets is aimed at reducing complexity in our bounds; it ensures that the proof-checker needs to examine at most $|\mathcal{E}||\mathcal{T}|$ cut-sets instead of possibly all $2^{|\mathcal{V}|}$ cut-sets.

Our proof-checking procedure then calculates, for each $t \in \mathcal{T}$, the probability $E^{t,a}$ that $[B^{t,a}]$ has rank at least $m(C - a\epsilon_m)$.

The step-counter a is then incremented by 1 and this procedure repeats until $a = |F^{\mathcal{T}}|$. After the above procedure terminates, for each $t \in \mathcal{T}$, frontier edge set $\mathcal{F}^{t,|F^{\mathcal{T}}|}$ consists only of edges e such that $v_h(e) = t$.

A lower bound for $E^{t,a}$ can be obtained as follows. Assume, for notational convenience, that $e(a+1)$ replaces $e(a)$ in the transition from $\mathcal{F}^{t,a}$ to $\mathcal{F}^{t,a+1}$. As an inductive hypothesis, with probability $1 - a \cdot 2^{-m\epsilon+\log(m+1)}$, $[B^{t,a}]$ has rank at least $m(C - a\epsilon_m)$. Let $[\hat{B}^{t,a}]$ be any matrix consisting of a subset of rows of $[B^{t,a}]$ that forms a basis for the row space of $[B^{t,a}]$. (Denote the dimension of $[\hat{B}^{t,a}]$ by $K \leq mC$.) Since each edge carries a vector of block length m, m rows of $[B^{t,a}]$ correspond to any edge $e \in \mathcal{F}^{t,a}$. with the reduction of $[B^{t,a}]$ to $[\hat{B}^{t,a}]$, some of those rows may be discarded. We use $[\hat{B}^{t,a}_{e(a)}]$ to describe the retained $\hat{m} \leq m$ rows for edge $e(a)$ and $[\hat{B}^{t,a}_{\mathcal{F}\setminus e(a)}]$ to describe the remainder of $[\hat{B}^{t,a}]$. Our goal is to show that with high probability the matrix $[B^{t,a+1}]$ has rank at least $K - m\epsilon_m$. For that purpose, we assume that the rows of $[B^{t,a+1}]$ are only dependent on vectors from $[\hat{B}^{t,a}]$. This is not true in general because the vector carried by edge $e(a+1)$ may be linearly dependent on vectors outside the span of $[\hat{B}^{t,a}]$, carried on edges $e \notin \mathcal{F}^{t,a}$. However, such linear dependencies can only increase the rank of $[B^{t,a+1}]$ so we assume the worst case where they do not exist. Thus,

$$\text{rank}([B^{t,a+1}]) \geq \text{rank}([\hat{L}^{t,a}] \cdot [\hat{B}^{t,a}]),$$

where $[\hat{L}^{t,a}]$ is a $K \times K$ matrix. Since $\mathcal{F}^{t,a}$ and $\mathcal{F}^{t,a+1}$ differ in exactly one edge ($e(a)$ is replaced with $e(a+1)$), rearranging rows and columns, $[\hat{L}^{t,a}]$ can be written as

$$[\hat{L}^{t,a}] = \begin{bmatrix} I & 0 \\ \left[\hat{L}^{t,a}_1\right] & \left[\hat{L}^{t,a}_2\right] + \left[\hat{\beta}^{e(a),e(a+1)}\right] \end{bmatrix}.$$

The top blocks of $[\hat{L}^{t,a}]$ (namely the order $K - \hat{m}$ identity matrix and the $(K - \hat{m}) \times \hat{m}$ zero matrix) represent the linear transformation of the vectors of $\mathcal{F}^{t,a} \setminus e(a)$ (which, by the definition of the inductive step, are unchanged). The bottom blocks represent the linear dependencies between $[\hat{B}^{t,a}]$ and the $\hat{m}$ vectors of $[B^{t,a+1}]$ carried on $e(a+1)$. Here $[\hat{\beta}^{e(a),e(a+1)}]$ refers to the part of $[\beta^{e(a),e(a+1)}]$ corresponding to the basis vectors that appear in $[\hat{B}^{t,a}_{e(a)}]$. It is therefore a minor of the random permutation matrix $[\beta^{e(a),e(a+1)}]$ and is itself a random permutation matrix (of size $\hat{m} \times \hat{m}$). The matrix $[\hat{L}^{t,a}_1]$ corresponds to the linear combinations of vectors from edges in $\mathcal{F}^{t,a} \setminus e(a)$, that contribute to the vectors on $e(a+1)$.

We must now show that the probability that the rank of $[B^{t,a+1}]$ is smaller than the rank of $[B^{t,a}]$ by more than $m\epsilon_m$ is *at most* $2^{-m\epsilon_m+\log(m+1)}$. In that case, adding this probability to the failure probability in the induction hypothesis proves the induction step. Our next step is therefore analyzing the rank of $[\hat{L}^{t,a}]$, as the nullity of $[\hat{L}^{t,a}]$ is an upper-bound on the difference between the ranks of the transformations $[B^{t,a}]$ and $[B^{t,a+1}]$. Gaussian elimination on $[\hat{L}^{t,a}]$ results in a matrix of the form

$$\begin{bmatrix} I & 0 \\ 0 & [\hat{L}^{t,a}_2] + [\hat{\beta}^{e(a),e(a+1)}] \end{bmatrix}.$$

Therefore the nullity of $[\hat{L}^{t,a}]$ equals the nullity of $[\hat{L}^{t,a}_2] + [\hat{\beta}^{e(a),e(a+1)}]$. By Lemma 4, the nullity of $[\hat{L}^{t,a}_2] + [\hat{\beta}^{e(a),e(a+1)}]$ is at most $m\epsilon_m$ with probability at least $1 - 2^{-m\epsilon_m+\log(\hat{m}+1)} \geq 1 - 2^{-m\epsilon_m+\log(m+1)}$, as required for the induction proof.

The above shows that the overall linear transformation from the source to any receiver is, with high probability, close to full-rank. To show that a multicast of rate R is achievable, we need only show that, with high probability, the image of the source encoder $[A] : (\mathbb{F}_2)^{mR} \to (\mathbb{F}_2)^{mC}$ does not intersect the null-space of $[B^{t,|F^{\mathcal{T}}|}]$ for any $t \in \mathcal{T}$. In addition, $[A]$ is required to have rank mR with high probability. Define $n = mC$, $k = mR$ and $r = \dim \mathsf{Ker}([B^{t,|F^{\mathcal{T}}|}])$, where $\mathsf{Ker}(\cdot)$ denotes the null-space of a transformation. Let $\mathbf{Pr}[n,k,r]$ be the probability that a matrix $[A']$, selected uniformly at random from all $n \times k$ binary matrices, has rank k and an image space that does not intersect an arbitrary fixed subspace of dimension

r. Then

$$\begin{aligned}\mathbf{Pr}[n,k,r] &= \frac{(2^n-2^r)(2^n-2^{r+1})\cdots(2^n-2^{r+k-1})}{2^{kn}} \\ &= \left(1-\frac{2^r}{2^n}\right)\left(1-\frac{2^{r+1}}{2^n}\right)\cdots\left(1-\frac{2^{r+k-1}}{2^n}\right) \\ &\geq \left(1-2^{-n+r+k-1}\right)^k\end{aligned}$$

The numerator on the right hand side of the first equality is the number of ways to select k length n binary vectors such that their span is a vector space of dimension k outside the span of a fixed dimension r vector space. The denominator is the total number of ways to select k such vectors and since we assume uniform distribution, the probability equals this ratio. Given $n = mC$, $k = mR$, and $r \leq m|\mathcal{E}|\epsilon_m$, we get

$$\mathbf{Pr}[mC, mR, r] \geq \left(1-2^{-m\epsilon_m}\right)^{mR} \geq 1 - mR \cdot 2^{-m\epsilon_m}$$

The last inequality holds for sufficiently large m. Taking the union bound of all of the events that cause the code to fail we get

$$\mathbf{Pr}(\text{fail}) \leq |\mathcal{T}||\mathcal{E}| \cdot 2^{-m\epsilon_m+\log(m+1)} + |\mathcal{T}|mR \cdot 2^{-m\epsilon_m}$$

the first summand is the union bound for the probability that any sink in $\mathcal{T}$ has rank loss greater than $m|\mathcal{E}|\epsilon_m$. The second summand is the union bound for the probability that the image space of $[A']$ intersects any of the spaces $\mathsf{Ker}([B^{t,|F^{\mathcal{T}}|}])$. The above bound readily leads to

$$\mathbf{Pr}(\text{fail}) \leq 2^{-m\epsilon_m+\log(m+1)+\log[|\mathcal{T}|(|\mathcal{E}|+C)]}.$$

□

Matrix $[A']$ used in the proof is chosen uniformly from all $mC \times mR$ binary matrices, most of which are dense (have portion of nonzero elements close to half). However, the source encoder $[A]$ that is obtained by Gaussian elimination on the rows of $[A']$ allows encoding functions that sum $\mathcal{O}(m)$ information bits per encoder output length-m bit-vector. This sparsifying transformation does not affect the rank of the overall transformation between the source vector to the vectors received by the sink nodes, and thus the bound on $\mathbf{Pr}(\text{fail})$ applies for $[A]$ as well. This $\mathcal{O}(m)$ sparsity of the source encoder is the same as the sparsity exhibited by internal encoders.

Theorem 5 proves that random distributed design of permute-and-add network codes fails with probability that is vanishing with growing m. If we permit a small amount of feedback from each t to s, then each sink t can tell source s the matrix $[B^{t,|F^{\mathcal{T}}|}]$. The expected number of code-design attempts required to guarantee that the permute-and-add code works is at most 2 whenever m is large enough for $\mathbf{Pr}(\text{fail})$ to be less than $1/2$.

V. Future Research and Conclusions

We concentrate on a particular notion of complexity of network codes; the number of arithmetic operations required per node to implement a rate-optimal multicast network code. We show that existing network code designs result in a quadratic growth of complexity with respect to the block-length. We define a new class of linear network codes called permute-and-add network codes which exhibit a per-node complexity that is linear in the block-length. We show that this complexity is order-optimal for network multicast codes.

Permute-and-add codes can be generalized to multicast networks with multiple sources. They can also be used in frameworks that limit the number of nodes that need to perform non-trivial network coding [8],[2]. It would be interesting to see whether permute-and-add codes can also be used in the framework of [9], i.e., whether we can apply an identical permutation matrix for each coding operation to achieve rate-optimal network multicast codes.

There are many opportunities for future work to extend our results. The random code design we propose is only guaranteed to result in codes that are asymptotically rate-optimal with high probability. It is possible that mimicking techniques used in [6] could enable (centralized) design of finite block-length codes that are guaranteed to achieve the network multicast capacity. In another direction, permute-and-add codes still result in dense decoding matrices, and therefore have a decoding complexity comparable to that of algebraic network codes; it would be interesting to see if similar ideas can be used to design block network codes with low decoding complexity.

VI. Acknowledgments

We thank Jehoshua Bruck and Alexander Sprintson for useful discussions and comments.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan. Minimal network coding for multicast. In *IEEE International Symposium on Information Theory (ISIT)*, Adelaide, Australia, Sept 4-9 2005.

[3] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Yokohama, Japan, June 29–July 4 2003.

[4] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger. On randomized network coding. In *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2003.

[5] S. Jaggi, P. A. Chou, and K. Jain. Low complexity algebraic multicast network codes. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, page 368, Yokohama, July 2003.

[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*, 51(6):1973–1982, June 2005.

[7] R. Koetter 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 (INFOCOMM)*, volume 1, pages 122–130, 2002.

[8] M. Langberg, A. Sprintson, and J. Bruck. The encoding complexity of network coding. In *IEEE International Symposium on Information Theory (ISIT)*, pages 1987–1991, Sept 4-9 2005.

[9] A. H. Lee and M. Médard. Simplified random network codes for multicast networks. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Adelaide, Australia, Sept 4-9 2005.

[10] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.

[11] R. Lidl and H. Niederreiter. *Finite Fields*. Cambridge University Press, UK, 1997.

[12] P. Pakzad, C. Fragouli, A. Shokrollahi Coding schemes for line networks In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Adelaide, Australia, Sept 4-9 2005.