"Real" and "Complex" Network Codes: Promises and Challenges

S. Shintre, S. Katti, S. Jaggi,
B. K. Dey, D. Katabi, and M. Médard

Abstract — **As an alternative to the algebraic network codes prevalent in the literature, we consider *Arithmetic* Network Codes (henceforth abbreviated as ANCs), i.e., codes in which interior nodes perform finite precision arithmetic over the real or complex fields. We suggest two applications where using such codes can be advantageous. First, we demonstrate that the multi-resolution behaviour of ANCs potentially outperforms that of algebraic network codes. Second, the interfering and fading nature of wireless channels naturally results in complex linear combinations of transmissions, analogous to ANCs. We then characterize the multicast rates achievable by ANCs, and demonstrate that for high precision arithmetic these are equivalent to those obtained by algebraic network codes. We show the connection between the performance of ANCs and the numerical conditioning of network transform matrices. Using this, we obtain upper and lower bounds on the number of significant bits required to perform the finite precision arithmetic in terms of the network parameters. We compare this with simulation results for randomized and deterministic design of ANCs.**

I. Introduction

The paradigm of *network coding* [1], i.e., coding at all nodes of a network rather than just at the edges, has been intensively studied recently. By now it is known how to construct and implement information-theoretically optimal robust network codes for a variety of problems in a distributed and efficient manner (see [2], [3], [4] for a sampling of the extensive literature). Perhaps the most compelling reason for their popularity is that such performance can usually be achieved by *linear* operations at each node. Using linear operations means that both design and implementation can be handled via relatively simple and well-understood principles.

In most network code designs, all the linear operations are over a finite field. However, for sources or networks better described by real (or complex) arithmetic, as in some real world applications, it might be advantageous to consider network codes that are linear over real (or complex) fields. We motivate ANCs via two "real world" communication problems – multi-resolution multicast, and wireless multicast. In the first case, each receiver desires a graceful tradeoff in distortion performance versus throughput. Since the distortion measure corresponds to the real field, ANCs are better suited. In the second case, the interference and fading properties of the wireless medium are naturally modeled via complex addition and multiplication respectively.

However, the main thrust of *this* work is to characterize the *differences* between finite field network codes (henceforth abbreviated as FFNCs) and ANCs for large networks. For random distributed FFNCs over large finite fields, with high probability, each sink can decode the source's information. At first glance it would seem that performing finite precision arithmetic with a large number of significant bits/digits should result in ANCs with similar properties. However, the results in this work paints a subtler picture. It turns out that the finite-precision artifacts can dominate the performance of ANCs. Repeatedly applying finite-precision linear transforms can make the network transform *ill-conditioned*, i.e., numerically indistinguishable from a singular matrix. Such ill-conditioning can scale quickly with network size.

We first sketch the two aforementioned applications of ANCs, both of which will be explored further in [5].

A. Multi-resolution multicast

Most of the network coding literature focuses on *multicast* problems, i.e., when each receiver demands all the information at each sources. This scenario is quite well understood; there exist low-complexity distributed codes that achieve capacity [3]. However, for more general information flow problems, linear codes may not achieve capacity [6]. Even for linear codes, it is NP-hard to construct capacity-achieving codes [7]. However, approximation algorithms exist for some special cases [8].

A natural generalization to the multicast problem is to consider *multi-resolution multicast*, i.e., allowing different receivers to have different download capacities. Suppose each sink wishes to reproduce with minimum distortion a sequence of real numbers from the source. For the usual implementation of FFNCs, even if a receiver's capacity is only marginally less than the source's rate, the minimum mean-square error of the receiver's estimate can still be high. The problem arises since the $\mathbb{R}$-vector space in which the source's data is naturally embedded is quite different from the $\mathbb{F}_q$-vector space over which arithmetic is performed by the network's nodes. Hence the resulting distortion can be much higher. Instead, building on powerful results from the field of *compressed sensing* [9], [10], ANCs that can be designed and implemented in a low

[0]S. Shintre and B. K. Dey are with the Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India, 400 076, email: {bikash,saurabh}@ee.iitb.ac.in .
S. Jaggi is with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong, email: jaggi@ie.cuhk.edu.hk .
S. Katti, D. Katabi, and M. Médard are with CSAIL/LIDS, Massachusetts Institute of Technology, Cambridge, MA 02139, email: skatti,dk,medard}@mit.edu .
The first and third author were supported by a CUHK Direct grant, CERG Grant No. 412207, and the MS-CU-JL grant.

978-1-4244-1689-9/08/$25.00 ©2008 IEEE

complexity and distributed manner are shown [5] to obtain a graceful tradeoff between throughput and distortion.

B. Wireless network coding

Wireless networks are significantly different from wired networks for several reasons – their broadcast nature, and the presence of interference, noise, and fading. Current network codes usually only exploit the broadcast property [11]. Most code designs for wireless networks (e.g. [12],[13]) assume noiseless links, and either no interference due to careful scheduling or that interfering packets are dropped. These assumptions essentially transform the wireless network coding problem into something like a wired network coding problem.

However, the brute-force approach of eliminating interference and fading ignores the inherent opportunities. For many multiple access channels it is well-known that significantly higher rates can be achieved than those achievable via time-sharing point-to-point communications [14]. Also, since wireless networks naturally change amplitudes and phases of transmissions, and add together interfering signals, the medium itself generates linear combination of the transmissions. This has been noted in some previous works (e.g. [15],[16] and [17]). However, in most such cases the network codes are still assumed to be linear over some $\mathbb{F}_q$ rather than over $\mathbb{C}$. For this to be a realistic assumption, the modulation scheme needs to be well-matched to the finite field. However, many such models are not robust to fading, synchronization, and noise. For ANCs all these issues are much more naturally addressed via well-studied signal processing techniques.

Consider the example illustrated in Fig. 1. The two sources S_1 and S_2 want to communicate two real numbers r_1 and r_2 respectively to each other over a wireless channel. However, they are not in each others transmission range and so they have to communicate via a common repeater R. With only routing, this requires four time slots as shown in Fig. 1(a). With FFNCs, the repeater can broadcast $r_1 + r_2$ to both S_1 and S_2 in a single time slot and thus requirw only three time slots (Fig. 1(b)). However, with appropriate power control and synchronization, if S_1 and S_2 simultaneously transmit r_1 and r_2 respectively, then R receives $r_1 + r_2$. R can then broadcast $r_1 + r_2$ to both S_1 and S_2 in just two time slots (Fig. 1(c)).

II. Challenges

The above scenarios, and the work outlined in [5] indicate the promising possibilities of ANCs. With ANCs the accuracy of the estimate of the source's data depends on the numerical stability of the system of linear equations corresponding to the network transform. In this work, we thus focus on characterizing the numerical stability of such systems of linear equations.

For noiseless wired multicast problems, the design of FFNCs is well understood. Before tackling harder problems, it is thus natural to consider design of ANCs for these problems. Analogous to convolutional network codes [18], if each node is allowed to perform arbitrary precision arithmetic, it is possible to design computationally tractable network codes that asymptotically achieve capacity. However, from a practical

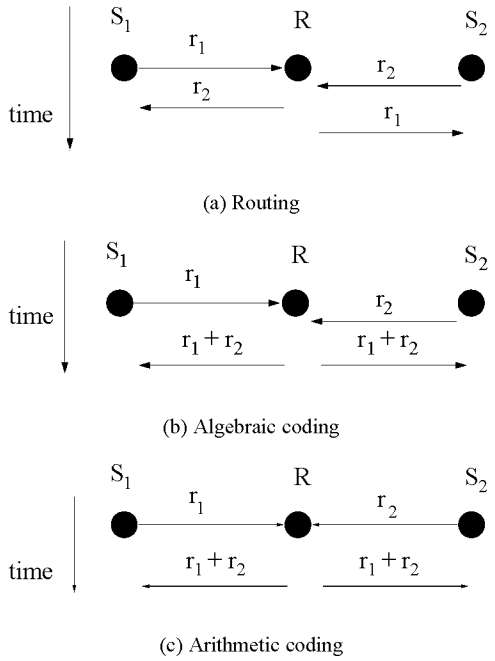

Figure 1: Different types of coding in wireless networks

standpoint quantization is necessary since the standard implementations of real arithmetic in digital computers handle finite precision floating-point calculations.

At first sight, it seems that ANCs should still have the same behaviour as FFNCs. After all, constructions of network codes over finite fields (for instance, see [19], [3]) work by maintaining the linear independence of information flowing in the network. The larger the size of finite field, the higher the probability of having an invertible network transform. Intuitively, therefore, the performance of ANCs with m significant bits per arithmetic operation should be comparable to the performance of FFNCs over $\mathbb{F}_{2^m}$. We show, surprisingly, that this is not true. For FFNCs, only the rank of the network transform matters – all network transforms with the same rank can transmit the same amount of information. However, for ANCs, since the linear transforms defining the network operations are over a normed vector space and finite precision arithmetic is performed, the *conditioning* of the network transform also matters. The *condition number* $\kappa(A)$ of a matrix A is a well-studied quantity in numerical analysis. Its inverse $\kappa(A)^{-1}$ measures how "close" the matrix A is to being singular. As shown in Section V, the amount of information that can flow through the network depends on the conditioning of the linear transforms at each node.

III. Example

We illustrate our point with an example. Consider the line network shown in Figure 2 with one source, one sink, and k serial

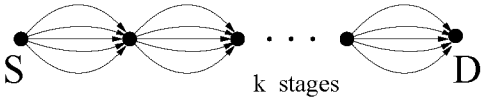

Figure 2: Line network with k hops

relays. There are C links between each consecutive pair of nodes. We consider a simple code where each node performs the *same* linear transform M. This is for ease of exposition – similar results can be obtained even if different operations are performed at each node. We consider one FFNC, and two kinds of ANCs.

First, we examine the performance of an FFNC. The first node chooses a random $C \times C$ matrix M with coefficients chosen uniformly i.i.d. from $\mathbb{F}_q$. The probability that M is singular can be shown to be at most $1 - C/q$ [20]. Let q equal 2^m. Then the probability that after k hops the linear transform is not invertible is at most $C2^{-m}$, which is independent of k.

Second, suppose the coefficients of M are chosen from $\mathbb{R}$. Each node applies the transform M with m-bit precision. Let the eigenvalues of largest and smallest magnitude of M be λ_l and λ_s respectively. Then with probability 1, $|\lambda_l| \neq |\lambda_s|$. After k hops, the ratio of the magnitudes of the largest and the smallest eigenvalues of the overall transform matrix is $|\lambda_l/\lambda_s|^k$. If $|\lambda_l/\lambda_s|^k > 2^m$, then the overall transform matrix is computationally singular. Note that this happens even though the rounding off is done only at the receiver. This seems to imply that quantization is the source of the problem, and that the overall performance must necessarily degrade exponentially with network size. However, the last part of this example shows that this degradation is significantly less for well-conditioned matrices.

As a third case, suppose M is a random unitary matrix. Therefore, all its eigenvalues are of unit magnitude. If quantization is done only at the receiver then, unlike the second case, there is no loss of rank in the overall transform. However, let us now consider the effect of rounding off after transformation at each repeater. Every addition or multiplication of m-bit precision incurs a normalized (w.r.t. the magnitude of the result) quantization noise of at most 2^{-m}. Every component of the output vector is obtained after C multiplications and $(C-1)$ additions. So, after the transformation and rounding off at the first repeater, every component of the message vector is corrupted by a quantization error of at most $2C2^{-m}$. This error accumulates at each stage, and after k stages, the error in each component is at most $2kC2^{-m}$. So, the mean square error (MSE) after k stages is at most $2kC^{3/2}2^{-m}$. The MSE grows only linearly with the size of the network, i.e., the loss of precision (in bits) grows logarithmically with the size of the network.

There are several interesting issues raised by the example above. First, at a high level, in large networks the behaviour of ANCs is intriguingly different from that of FFNCs. Second, the effect of quantization depends on how "well-behaved" the underlying linear transforms are – unitary transforms show much better performance than general transforms do. This leads to the third point, which is the difficulty of actually using unitary transforms as ANCs. For one, a node may have more outgoing links than incoming links. The local linear transform matrix must have more rows than columns, and therefore cannot be unitary. Further, what matters is the linear transform across cutsets of the network, and this is a global property of the code – using unitary matrices at individual nodes cannot guarantee global unitarity, especially if one wishes to design the ANC in a distributed manner. Even if one knew the entire network, it may not even be possible to design such a code. For an example of networks for which such issues exist, see the combination networks in Theorem 2.

IV. Model and Definitions

To simplify exposition, we consider only the problems where a single source s wishes to communicate a message exactly to a single destination t over an acyclic wired network, or to multicast it to a set $\mathcal{T}$ of receivers. Our notation is derived for the unicast case – for multicast problems we add a superscipt t specifying the sink node. However, using standard reductions our techniques generalize to general multicast communication networks (multiple sinks, delays at nodes and edges, cycles, wireless networks) over which random linear network codes are employed. The special case of unicast, as considered here, may also refer to a particular source-destination pair in a more general network. The major difference with FFNCs is that all operations involve finite precision arithmetic over real or complex fields, rather than over finite fields. We focus on real arithmetic, though everything can be directly translated to complex arithmetic as well.

Network Model: The network is modeled as a graph with vertices $\mathcal{V}$ and edge-set $\mathcal{E}$. There is a source, Alice, and a destination, Bob, who communicate over a wired network. Each transmission corresponds to an edge directed from the transmitting node to a receiver node. Each transmission communicates noiselessly a single m-bit vector.

Source: Alice has a C-length vector $\vec{\mathbf{X}}$ of quantized real numbers that she wishes to deliver to Bob over the network, where each real number in Alice's collection is a m-bit floating point number. Here C corresponds to Alice's *rate*. The results of all arithmetic operations henceforth are rounded off to m-bit floating point numbers as well.

Encoders and network: Alice takes linear combinations of the X_i to generate her transmissions. Each edge can carry a single m-bit real number. As the information traverses the network, the internal nodes also apply linear transforms to received real numbers to generate the m-bit real numbers they transmit. In particular, for each edge e leaving a node v, the real number $Y(e)$ traversing e is generated via the dot-product $\left\langle \beta(\vec{\mathbf{e}}), \mathbf{Y}(\vec{\mathbf{v}}) \right\rangle$. Here $\beta(\vec{\mathbf{e}})$ (called the *local coding vector for* e) is a vector over $\mathbb{R}$. Its components are real numbers $\beta(e', e)$, for all edges e' that are incoming to v. $\mathbf{Y}(\vec{\mathbf{v}})$ is the vector of all the real numbers that the node v receives. Also, let the *global coding vector for edge* e, denoted $\gamma(e) = [\gamma_1, \ldots, \gamma_C]^T$, be the C-length real vector that corresponds to the linear transformation from the source to e. That is, e carries the real number $\gamma_1 X_1 + \cdots + \gamma_C X_C$.

Decoder: The destination Bob organizes the received real numbers into a vector $\vec{\mathbf{Y}}$, from which he attempts to reconstruct Alice's information $\vec{\mathbf{X}}$ exactly.

Network and Code parameters: We define the following concepts. A *cut* S of the network is defined as a subset of

vertices containing Alice but not containing Bob. The value of a cut is defined as the sum capacity of all the links from S to its complement. The *min cut*, denoted by C, is the minimum of all the cuts in the network. The rate, R, is the number of information bits in a batch amortized by the precision m of the real numbers. The rate R is said to be *achievable* if for a sufficiently large m, there exists a m-bit precision network code where Bob recovers the source message exactly.

Network transform: To aid exposition, let us temporarily assume that each node can perform arbitrary precision arithmetic and transmit the result, rather than transmitting the rounded-off value. Since each transmission by an internal node is a linear combination of its incoming transmissions, the effect of the linear combinations at each network node can be written as follows. In general, for two cuts $S' \subseteq S$, the equation

$$\mathbf{Y}(\vec{\mathbf{S}}_{S'}) = \mathbf{T}_{S',S}\mathbf{Y}(\vec{\mathbf{S}}_S). \tag{1}$$

describes the linear relationship between the information in the two cuts $S_{S'}$ and S_S. In particular, $\mathbf{T}_{\phi,\mathcal{Y}}$ describes the *network transform* between Alice and Bob. We assume that Bob knows $\mathbf{T}_{\phi,\mathcal{Y}}$ (as in [3], its value can be computed and transmitted in a distributed manner).

A. The Condition Number of a Matrix

Consider the transform $\vec{\mathbf{y}} = \mathbf{A}\vec{\mathbf{x}}$, where $\mathbf{A}$ is a square matrix. If $\mathbf{A}$ is invertible and the computation $\mathbf{A}\vec{\mathbf{x}}$ is carried out exactly, it is possible to recover $\vec{\mathbf{x}}$ as $\mathbf{A}^{-1}\vec{\mathbf{y}}$ with no distortion. However, from the rounded-off value of $\vec{\mathbf{y}}$ it is possible only to get an estimate $\hat{\vec{\mathbf{x}}}$ of $\vec{\mathbf{x}}$. If the rounding-off error in $\vec{\mathbf{y}}$ is denoted by $\vec{\Delta\mathbf{y}}$ and the corresponding error $\vec{\mathbf{x}} - \hat{\vec{\mathbf{x}}}$ in the estimate $\hat{\vec{\mathbf{x}}}$ is denoted by $\vec{\Delta\mathbf{x}}$, then one would like to minimize the normalized error $\vec{\Delta\mathbf{x}}/\vec{\mathbf{x}}$. (In particular, $\log_2(\vec{\Delta\mathbf{x}}/\vec{\mathbf{x}})$ is the loss in floating point precision for fixed normalized error $\vec{\Delta\mathbf{y}}/\vec{\mathbf{y}}$ in the output.) In other words, the "goodness" of the transform $\mathbf{A}$ is indicated by the maximum value of

$$\frac{||\vec{\Delta\mathbf{x}}||/||\vec{\mathbf{x}}||}{||\vec{\Delta\mathbf{y}}||/||\vec{\mathbf{y}}||} = \frac{||\mathbf{A}^{-1}\vec{\Delta\mathbf{y}}||/||\mathbf{A}^{-1}\vec{\mathbf{y}}||}{||\vec{\Delta\mathbf{y}}||/||\vec{\mathbf{y}}||}. \tag{2}$$

This maximum value can be readily seen to be equal to the product of the corresponding matrix operator norms of $\mathbf{A}$ and its inverse [21]. The operator norm of $\mathbf{A}$ is defined as $||\mathbf{A}|| = \sup\left\{||\mathbf{A}\vec{v}||/||\vec{v}|| : \vec{v} \neq \vec{0}\right\}$. The operator norm used in (2) is chosen to match the corresponding vector norm in the left hand side. The quantity $||\mathbf{A}||\,||\mathbf{A}^{-1}||$ is known as the *condition number* of $\mathbf{A}$ and is denoted by $\kappa(\mathbf{A})$. Let a_{ij} and $\hat{a}_{ij}$ respectively be the entries of $\mathbf{A}$ and $\mathbf{A}^{-1}$. In general, the condition number of a matrix can be defined with respect to any norm. For the *row-sum (resp. column-sum) operator norm*, the corresponding codition number, denoted $\kappa_\infty(\mathbf{A})$, equals $(\max_i |\sum a_{ij}|)(\max_i \sum |\hat{a}_{ij}|)$ (resp. $(\max_j |\sum a_{ij}|)(\max_j \sum |\hat{a}_{ij}|)$). For the operator norm induced by the l_2 vector norm, the condition number is known to be same as $\sigma_{max}(\mathbf{A})/\sigma_{min}(\mathbf{A})$, where $\sigma_{max}(\mathbf{A})$ and $\sigma_{min}(\mathbf{A})$ are respectively the maximum and the minimum eigen values of $\mathbf{A}$. The following properties of condition numbers will be useful.

- $\kappa(\mathbf{A}) \geq 1$ and $\kappa(\mathbf{A}) = 1$ for unitary matrices.
- $\kappa(\mathbf{AB}) \leq \kappa(\mathbf{A})\kappa(\mathbf{B})$ for any two matrices $\mathbf{A}, \mathbf{B}$.
- From (2), it is clear that the condition number gives the maximum normalized error in $\vec{\mathbf{x}}$ per unit normalized error in $\vec{\mathbf{y}}$. So, $\log_2(\kappa(\mathbf{A}))$ is the loss in precision in the estimate $\hat{\vec{\mathbf{x}}}$ due to rounding-off of $\vec{\mathbf{y}}$.

V. Results

We have two sets of results – the first two results give upper bounds to the rates achievable by ANCs, and the next two give lower bounds on the same.

First, we give a general cutset bound on the best rate achievable by a given ANC over a given network. Let S be any cut of the network. We define *mincut up to* S, denoted C_S, as the smallest cut that is a subset of S, i.e., $\min_{S':S'\subseteq S} S(S')$. Let $\mathbf{T}_\mathbf{S}$ denote the (infinite precision) linear transform from s to the edges in S.

Theorem 1 *An upper bound to the rate achievable by a given ANC is*

$$\min_{s:S \text{ is a cut}} C_S - \frac{1}{m}\log_2(\kappa(\mathbf{T}_\mathbf{S})). \tag{3}$$

Proof: The mincut up to S is an upper bound on the amount of information that can flow out of the cut S. As discussed in Section IV.A, $\log_2(\kappa(\mathbf{T}_\mathbf{S}))$ is the number of significant bits lost by performing a linear transform using finite-precision arithmetic – normalizing this quantity with the number of significant bits m gives us the required result. □

Theorem 1 gives a performance bound for a particular ANC – we now give a lower bound on the condition number of *any* ANC for a particular class of *combination networks*. Let $\mathcal{G}_{k,C} = (\mathcal{V}, \mathcal{E})$ have vertices $\mathcal{V} = \{s\} \cup U \cup \mathcal{T}$ where $U = \{1, \ldots, k\}$, $\mathcal{T} = \{t_W \mid W \subseteq U, |W| = C\}$, and edges $\mathcal{E} = \{(s,u) \mid u \in U\} \cup \{(u, t_W) \mid t_W \in \mathcal{T}, u \in W\}$. That is, the source s constitutes the first layer, the k nodes in U constitute the second layer, and the $\binom{k}{C}$ nodes described by $\mathcal{T}$ constitute the third; each node in $\mathcal{T}$ is connected by unit capacity links to a distinct C-element subset W of U. Figs. 3 shows an example of such a graph. Combination networks have been

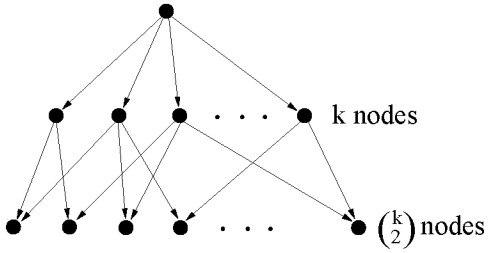

Figure 3: Cobination network $\mathcal{G}_{k,2}$

useful in proving lower bounds for a variety of network coding problems (see, for example, [7]).

Theorem 2 *For any ANC that asymptotically achieves the capacity of* $\mathcal{G}_{k,2}$*, the minimum value of* $\max_t \kappa(\mathbf{T}_{\mathbf{s},\mathbf{t}})$ *is* $2/(\sin(\pi/k))$.

Proof: Let the length-2 global coding vector on the i^{th} edge leaving the source, $\gamma(e_i)$, be written as $(r_i \cos(\theta_i)\, r_i \sin(\theta_i))$. For any sink t, the linear transform between s and t is a 2×2 matrix

$$\mathbf{T_{s,t}} = \begin{bmatrix} r_i \cos(\theta_i) & r_i \sin(\theta_i) \\ r_j \cos(\theta_j) & r_j \sin(\theta_j) \end{bmatrix}$$

for some corresponding i and j in $\{1, \ldots, k\}$

The condition number $\kappa_F(\mathbf{T_{s,t}})$ corresponding to the Frobenius norm of the matrix can be directly computed as

$$\kappa_F((\mathbf{T_{s,t}})) = \frac{r_i^2 + r_j^2}{\sqrt{2} r_i r_j \sin(\theta_j - \theta_i)}.$$

The minimum of this quantity over the variables r_i and r_j occurs when $r_i = r_j$ for all $i, j \in \{1, \ldots, k\}$. Since $\sin(\pi + \theta) = \sin(\theta)$, the minimum of the term $\max_{i,j} 1/|\sin(\theta_j - \theta_i)|$ occurs when the each $\theta_{i+1} - \theta_i = \pi/k$ for each $i \in \{1, \ldots, k\}$. □

Since the ratio $\sin(\theta)/\theta$ tends to 1 for θ tending to 0, and $k = \Theta(\sqrt{|\mathcal{V}|})$ for combination networks, Theorems 3 and 2 provide an asymptotic lower bound of $0.5 \log(|\mathcal{V}|)$ on the loss of precision of ANCs.

To complement Theorems 3 and 2 we present a lower bound on the rate achievable via a particular ANC, and a matching algorithm for designing ANCs. Our bound and algorithm are motivated by the classical deterministic network coding algorithm from [2]. To this end, we reproduce here the notion of *frontier edge sets* and *frontier edge set matrices* defined in [2]. This is done in two stages. In the first stage, a flow algorithm is run to find, for each sink $t \in \mathcal{T}$, a *flow to sink t*, i.e., a set $F^t = \{P_j(s,t)\}_{j=1}^{t}$ of C edge-disjoint paths $P_j(s,t)$ from the source s to sink t. Only the edges in the union of these flows over all sinks, denoted $F^{\mathcal{T}}$, are considered in the second stage.

The second stage visits each edge in turn and designs the linear code employed on that edge. The order for visiting the edges is chosen so that the encoding for edge e is designed after the encodings for all edges leading to e. In particular the edges in $F^{\mathcal{T}}$ are numbered from 1 to $|F^{\mathcal{T}}|$. This is possible since by assumption the network is directed and acyclic. Also, a *step-counter a* which keeps track of the stage of our code design, is initialized to 1. At stage a, our algorithm designs the local coding vector $\beta(\vec{\mathbf{e}})$ for edge e_a. At each step a, our algorithm inductively defines the *frontier edge set for sink t at step a* as an ordered subset $\mathcal{F}_a^t$ containing C edges in the following manner. For each $t \in \mathcal{T}$, $\mathcal{F}_1^t$ equals the first edges on the paths $P_j(s,t)$. The j^{th} edge in $\mathcal{F}_a^t$ is the edge $e_{a'}$ in $P_j(s,t)$ with the largest value of a' less than or equal to a. The edges in each $\mathcal{F}_a^t$ thus form a subset of size C of a cut from s to t. The *frontier edge set matrix for sink t at step a* is the $C \times C$ matrix $\mathbf{T_a}$ composed of the global coding vectors for the edges in $\mathcal{F}_a^t$. 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 $t \in \mathcal{T}$ the frontier edge set $\mathcal{F}_{|F^{\mathcal{T}}|}^t$ consists only of edges e that terminate at t. We denote the $C \times C$ linear transform from $\mathcal{F}_a^t$ to $\mathcal{F}_{a+1}^t$ by $\mathbf{T^t_{a,a+1}}$,

Theorem 3 *A lower bound to the rate achievable by a given ANC is*

$$C - \max_{t \in \mathcal{T}} \frac{1}{m} \sum_{a=0}^{|F^{\mathcal{T}}|-1} \lceil \log_2(\kappa(\mathbf{T^t_{a,a+1}})) \rceil \qquad (4)$$

Proof: As shown in Section IV.A, the number of significant bits lost in applying a linear transform $\mathbf{T}$ is at most $\lceil \log_2(\kappa(\mathbf{T})) \rceil$. Thus the overall loss in rate, normalized w.r.t. the number of significant bits, is as outlined in (4). □

Note: The bound in Theorem 3 may depend significantly on the order chosen on the edges in F – one way to improve it is to maximize it over all such orderings.

Deterministic ANC algorithm: The goal in designing the local coding vector $\beta(\vec{\mathbf{e}})$ for each e is to ensure that for each t and a, the condition number of the $C \times C$ linear transform from $\mathcal{F}_a^t$ to $\mathcal{F}_{a+1}^t$, denoted by $\mathbf{T^t_{a,a+1}}$, is as small as possible. We outline this code design choice below.

Designing $\beta(\vec{\mathbf{e}})$s: By our construction above, $\mathcal{F}_a^t$ and $\mathcal{F}_{a+1}^t$ differ in at most one edge (e_a, if it occurs in one of the paths towards t, replaces one of its immediate predecessors). Therefore for each a and t, the linear transform $\mathbf{T^t_{a,a+1}}$ acts as the identity transform on at least $C-1$ distinct unit vectors. That is, each matrix $\mathbf{T^t_{a,a+1}}$ has the form $\begin{bmatrix} I_{C-1} & \vec{\mathbf{0}} \\ & \vec{\mathbf{v_a^t}} \end{bmatrix}$. Here $\vec{\mathbf{v_a^t}}$ is the vector that represents $\gamma(e_a)$, the global coding vector on e_a, in terms of the basis corresponding to the rows of $\mathbf{T_a^t}$. Since the global coding vector on e_a is obtained by taking the linear combination $\beta(e_a)$ of the global coding vectors on all the edges immediately preceding e_a, therefore $\gamma(a)$ depends linearly on $\beta(e_a)$. At each step a, our algorithm chooses $\beta(e_a)$ so as to minimize the largest of the condition numbers of $\{\mathbf{T^t_{a,a+1}}\}_{t \in \mathcal{T}}$. The goal is thus to perform the optimization $\min_{\beta(e_a)} \max_t \kappa(\mathbf{T^t_{a,a+1}})$. To simplify the computation, the operator norm we use to define $\kappa(\mathbf{T^t_{a,a+1}})$ is the row-norm (induced by the L_∞ norm on $\mathbb{R}^C$). Since the inverse of a matrix of the form $\mathbf{T^t_{a,a+1}}$ also has the same form, therefore the condition number can be computed with low computational overhead. □

Note: While our algorithm has the benefits of being computationally tractable, and also matching the bounds given in Theorem 3, it does not yield a performance bound independent of the ANC used. This is one of our current areas of interest.

VI. Simulation Results

In this section, we present some MATLAB simulation results to illustrage that random arithmetic coding produces overall transform matrices with very large condition number. We consider the line network in Fig. 2 with $C = 2$ links from any node to the next, and the network obtained by stacking k butterfly networks as shown in Fig. 4. In either case, each edge carries a vector of length n. Then for the line network, the transformation at each node is taken to be a $nC \times nC$ matrix with components chosen i.i.d. uniformly distributed in $[0, 1]$. The matrices at different nodes are also chosen i.i.d. The overall transform matrix from the source to the destination is computed by multiplying all the matrices. Fig. 5 shows the condition number, averaged over 100 different experiments, of the

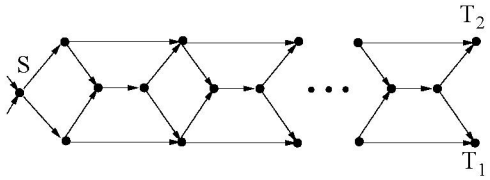

Figure 4: Stacked butterfly network

overall transform matrix as a function of k. The figure shows a log_2 plot for each $n = 1, 2, 3, 4$. As expected, the condition number of the overall transform matrix with random coding is seen to increase exponentially with the number of hops.

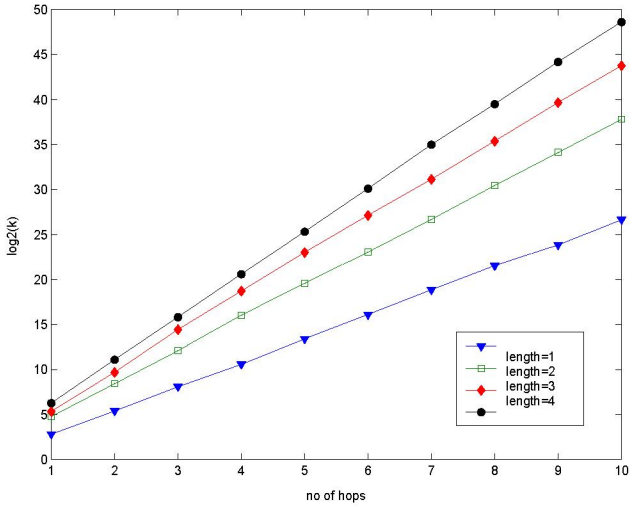

Figure 5: Condition number of the overall transform for the line network

For the stacked butterfly network, random coding vectors are chosen for each node and the over all transform matrix is computed for one of the destinations. For this network also, the behaviour of the condition number of the overall transform matrix is seen to be growing exponentially with k. The saturation at the top of curves is due to finite precision numerical computation in MATLAB.

The results suggest that with random arithmetic coding, the loss of precision of the source data increases linearly with the network size.

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] 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.

[3] T. Ho., R. Koetter, 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)*, page 442, Yokohama, July 2003.

[4] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, and M. Effros. Resilient network coding in the presence of byzantine adversaries. In *INFOCOM*, Anchorage, AK, 2007.

[5] S. Katti, S. Shintre, D. Katabi, M. Médard, and S. Jaggi. Compressed sensing via linear network coding. To be presented at the Allerton conference (Invited talk), 2007.

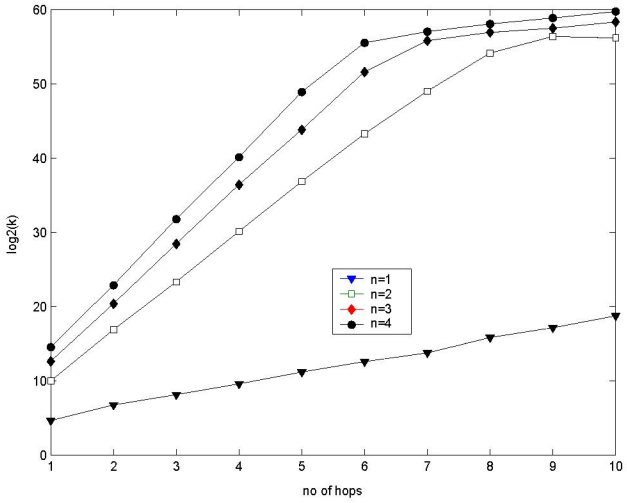

Figure 6: Condition number of the overall transform for stacked butterfly network

[6] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow. *Junge Wissenschaft*, 51(8):2745–2759, 2005.

[7] A. Rasala Lehman and E. Lehman. Complexity classification of network information flow problems. In *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, 2003.

[8] T. Ho and H. Viswanathan. Dynamic algorithms for multicast with intra-session network coding. submitted to IEEE Transactions on Information Theory, 2007.

[9] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, April 2006.

[10] E. Cands, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. on Information Theory*, 52(2):489–509, February 2006.

[11] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, page 243, 2006.

[12] M. Mdard R. Koetter D. R. Karger T. Ho E. Ahmed D. S. Lun, N. Ratnakar and F. Zhao. Minimum-cost multicast over coded packet networks. *IEEE Trans. Inform. Theory*, 52(6):2608–2623, June 2006.

[13] A. F. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros. Capacity of wireless erasure networks. *IEEE Transactions on Information Theory*, 52(3):789–804, 2006.

[14] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.

[15] S. C. Liew S. L. Zhang, Y. Zhu and K. B. Letaief. Joint network coding and channel decoding design for wireless network. In *IEEE Wireless Communications and Network Conference (WCNC)*, March 2007.

[16] S. N. Diggavi S. Avestimehr and D. N. C. Tse. A deterministic model for wireless relay networks and its capacity. In *IEEE Information Theory Workshop (ITW)*, 2007.

[17] S. Bhadra, P. Gupta, and S. Shakkottai. On network coding for interference networks. In *ISIT*, 2006.

[18] E. Erez and M. Feder. Convolutional network codes. In *ISIT*, June 2004.

[19] R. Koetter and M. Médard. Beyond routing: An algebraic approach to network coding. *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM)*, 1:122–130, 2002.

[20] C. Cooper. On the distribution of rank of a random matrix over a finite field. *Random Structures and Algorithms*, 17(3-4):197–212, 2000.

[21] Erwin Kreyszig. *Advancd Engineering Mathematics*. John Wiley & sons, 9^{th} edition.