Interference Alignment as a Tool in Network Coding as Applied to Distributed Storage

K. V. Rashmi[†], Nihar B. Shah[†], P. Vijay Kumar[†], Kannan Ramchandran[#]

[†] Dept. of ECE, Indian Institute Of Science, Bangalore, India.
Email: {rashmikv, nihar, vijay}@ece.iisc.ernet.in
[#] Dept. of EECS, University of California, Berkeley, USA.
Email: kannanr@eecs.berkeley.edu

***Abstract*—In this paper, we outline an approach to the task of designing network codes in a non-multicast setting. Our approach makes use of the concept of interference alignment. As an example, we consider the distributed storage problem where the data is stored across the network in n nodes and where a data collector can recover the data by connecting to any k of the n nodes and where furthermore, upon failure of a node, a new node can replicate the data stored in the failed node while minimizing the repair bandwidth.**

I. Introduction

We begin with a detailed description of the distributed storage setup. A file of size B is to be stored in a distributed manner across n storage nodes, each having a storage capacity of α units of data (symbols). Each symbol belongs to a finite field $\mathbb{F}_q$ of size q. A data collector (DC) which downloads data stored in any k out of the n nodes should be able to *reconstruct* the entire file. We consider the setting in which each node stores the minimum amount of data required for any DC to reconstruct the data i.e

$$\alpha = B/k \tag{1}$$

When a node fails, the new node replacing the failed node is permitted to download β symbols each from d existing nodes in order to obtain (and store) the same data as the failed node. This is termed as *exact regeneration*. We assume that the data is stored in systematic form in some k out of the n nodes, say nodes $1, \ldots, k$. Our interest is in minimizing the repair bandwidth for exact regeneration of the systematic nodes. For the purposes of this paper, we will refer to this problem as the *distributed storage problem*. Only linear codes will be considered.

The pioneering paper in this area [1] considers a more general setting in which the regenerated node need not be identical to the failed node and goes on to present bounds and achievability proofs for that setting.

In our previous work [2], we considered the problem of exact regeneration, however for the case when a node is allowed to store extra data in order to reduce the repair bandwidth. We also gave an approximately exact regenerating code when nodes have minimum storage and for the parameter set $d = k + 1$. Each of these codes meet the lower bound on the repair bandwidth.

In [3] authors consider the setting of exact regeneration and apply interference alignment to this specific setting. They provide schemes which reduce the dimension of interference, but to values larger than optimal, resulting in a suboptimal scheme.

The problem of obtaining codes to minimize the repair bandwidth is a non-multicast network coding problem, for which very few results are available. In [4] it is shown that determining whether a linear network coding solution exists for an arbitrary network is NP-hard. The insufficiency of linear coding for the non-multicast case is shown in [5].

We make the observation that in this class of networks, the notions of *interference alignment* and *useful information flow* are essential. Using these concepts, we provide a solution to the distributed storage problem which minimizes the repair bandwidth for exact regeneration of the systematic nodes for the case of $d \geq 2k - 1$.

We also show how these concepts can be applied to a larger class of networks as well as provide insight that will aid in code design. It will also be shown to help tighten existing cut-set based upper bounds.

The paper is organized as follows. Section II presents the distributed storage problem as a non-multicast problem. In Section III a set of necessary and sufficient conditions based on interference alignment are provided for a network of non-multicast type to be linearly solvable. In Section IV we apply these conditions to a class of networks - networks with crosslinks and obtain tighter upper bounds on the achievable rates as compared to the traditional cut-set bound. Section V specializes to the case of distributed storage problem. Finally, using the insights obtained here, a code minimizing the repair bandwidth is constructed in Section VI.

II. Distributed storage as a non-multicast problem

We now present the distributed storage problem as an instance of a non-multicast network coding problem in which the graph of the network is directed, delay free and acyclic. The network is viewed as having k source nodes, each corresponding to a systematic node and generating α symbols each per unit time. The non-systematic nodes are simply viewed as downlink nodes. Since it is only possible to download α symbols from a downlink node, this is taken care of in the graph as in [1], by (i) splitting each non-systematic node m into two nodes: m_{in} and m_{out} with an edge of capacity α linking the two with (ii) all

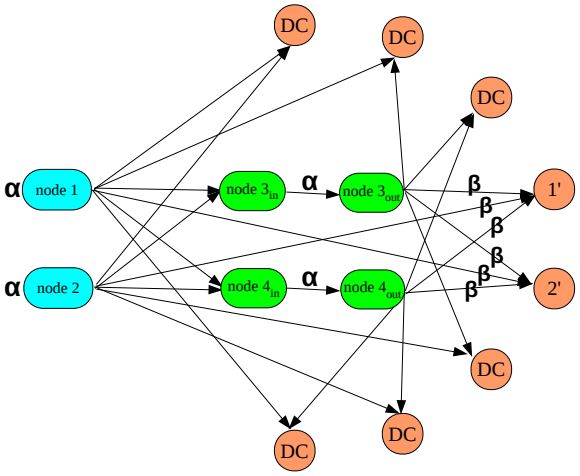

Fig. 1: Complete network for the distributed storage problem for $n = 4$, $k = 2$ and $d = 3$. Unmarked edges have capacity α.

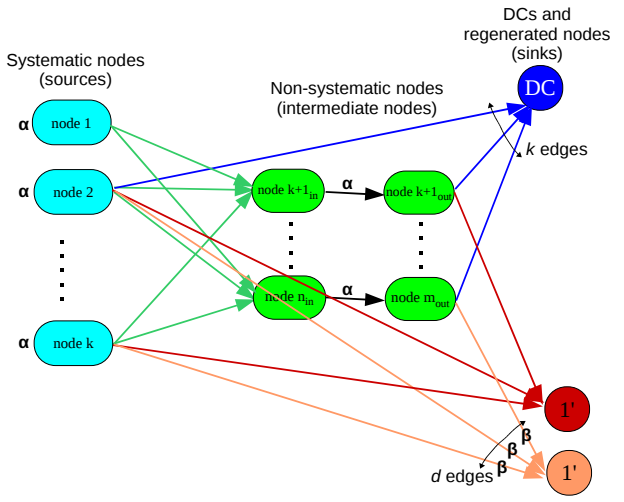

Fig. 2: Part of the multicast network of the general distributed storage problem. Unmarked edges have capacity α.

incoming edges arriving into m_{in} and all outgoing edges emanating from m_{out}. The sinks in the network are of two types. The first type correspond to data collectors which connect to some collection of k nodes in the network for the purposes of data reconstruction. Hence there are $\binom{n}{k}$ sinks of this type. The second type of sinks represents a new node that is attempting to duplicate a failed systematic node. Nodes of this type are assumed to connect to the remaining $k-1$ systematic nodes and any $d-(k-1)$ of the non-systematic nodes. Hence there are $\binom{n-k}{d-k+1}$ sinks of this type. The non-multicast network for the parameter set $n = 4, k = 2, d = 3$ is shown in Figure 1. A part of the network for the general problem is given in Figure 2.

We now introduce some additional notation with respect to this non-multicast network. Figure 2 shows a part of the network, and depicts one of the DCs which connects to some k nodes and two possible new nodes corresponding to failure of the first systematic node connecting to two different sets of d existing nodes.

Let $\underline{f}$ be a vector of length B corresponding to the B symbols produced by the k sources (systematic nodes). Let $\underline{f}^t = [\underline{f}_1 \dots \underline{f}_k]$ where $\underline{f}_l$ is an α length row vector corresponding to the α symbols generated by systematic node l.

Every edge e is associated with a matrix M_e of dimension $C_e \times B$ where C_e is the capacity of that edge. The rows of the matrix M_e are the C_e global kernels associated with C_e symbols flowing along the edge. The actual symbols carried by the edge is $M_e\underline{f}$. Columns $(l-1)\alpha+1$ to $l\alpha$ of M_e for any $l \in \{1, \dots, k\}$ are referred to as the columns corresponding to systematic node l.

Let tail(e) and head(e) be the tail and head vertices of edge e respectively. If the tail(e) of an edge e is a source node i.e. systematic node l ($\in \{1, \dots, k\}$) then in M_e, the columns corresponding to any other systematic node have to be zero. For any other edge e, M_e is a linear combination of matrices $M_{e'} \quad \forall e'$ such that head$(e') =$ tail(e), where the coefficients of linear combination themselves are matrices.

In the next section we will show how it is possible to use the concept of interference alignment to set up a framework for code design in a general network of non-multicast type. We will return in the subsequent sections to the problem of distributed storage.

III. Necessary and Sufficient Conditions for a General Network

The theorems stated in this section give a set of necessary and sufficient conditions that need to be satisfied by any linear coding solution to a general network of non-multicast type.

While the theorems on the one hand are intuitively obvious, they nevertheless provide a new and useful perspective to the problem of code design and this will be illustrated in the subsequent sections. In general, the viewpoint yields heuristics that aid in code construction and sometimes permit tighter upper bounds on achievable rates than the cut-based bounds.

Setting and notation: As mentioned earlier we consider delay free, acyclic, directed graphs. We also assume the networks to be error free. We consider scalar linear network coding solution for these networks. All symbols belong to some finite field $\mathbb{F}_q$. An edge e in the network can carry an integral number of symbols from $\mathbb{F}_q$, and the maximum number of such symbols it can carry at a time is called the capacity of that edge $\mathbf{C}_e$.

There are k independent sources $S_1, \dots, S_k$. Without loss of generality we assume that each sink demands all the information from exactly one source. If a sink demands multiple sources, or a part of a source, then an equivalent network can be constructed by splitting the sink or the source respectively. Also, we assume that there is at least one sink corresponding to every source. A sink is named T_l if it demands source S_l. The source and sink nodes do not have any incoming and outgoing edges respectively. Let R_l be the rate of the source S_l $(l = 1, \dots, k)$. An edge from vertex u to v is represented as $u \to v$.

A *cut* Ω (as illustrated in Figure 3) is a partition of vertices into two sets, called the *source side* and the *sink*

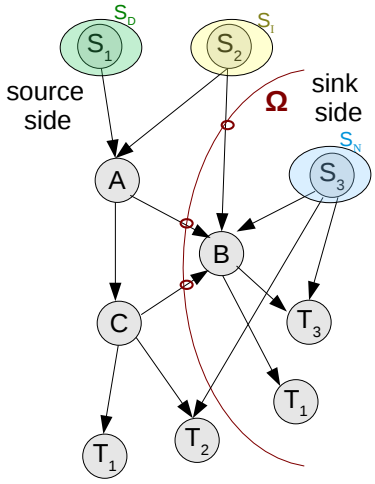

Fig. 3: An illustration of a cut Ω and associated source-sets

side partitions. Edges going across the cut from source side to sink side are said to belong to the cut. The capacity of a cut Ω, denoted by $\mathbf{C}(\Omega)$ is the sum of capacities of all the edges in the cut.

For any cut, the set of sources are divided into three sets: $\mathbf{S}_D(\Omega)$ is the set of *desired* sources, i.e. those which are on the source side of the cut and having at least one of its corresponding sinks on the sink side; $\mathbf{S}_I(\Omega)$ is the set of *interfering* sources, i.e. those which are on the source side of the cut and having none of its corresponding sinks on the sink side; and $\mathbf{S}_N(\Omega)$ is the set of *neutral* sources, i.e. those which are on the sink side of the cut. $\mathbf{R}_D(\Omega)$, $\mathbf{R}_I(\Omega)$ and $\mathbf{R}_N(\Omega)$ are the total rates of the three sets of sources respectively.

Let $\mathbf{R} = \sum_{l=1}^{k} R_l$. As defined in Section II, each edge is associated with a matrix of dimension $\mathbf{C}_e \times \mathbf{R}$.

Consider any cut Ω and an arbitrary set of sources $\mathbf{S}$. The dimension of $\mathbf{S}$ in the cut, denoted as $dim(\mathbf{S}, \Omega)$ is defined as the dimension of the vector space spanned by the rows of M_e $\forall e \in \Omega$ considering only the columns corresponding to the sources in $\mathbf{S}$. Let $\mathcal{E}(\Omega)$ for any cut Ω denote the set of rows of matrices of all the edges in that cut.

Theorem 1 (Useful Information Flow): A necessary condition for a code to achieve the rate tuple $(R_1, \ldots, R_k)$ is that for any cut Ω,

$$dim(\mathbf{S}_D(\Omega), \Omega) \geq \mathbf{R}_D(\Omega) \tag{2}$$

Theorem 2 (Interference Alignment): A necessary condition for a code to achieve the rate tuple $(R_1, \ldots, R_k)$ is that for any cut Ω,

$$dim(\mathbf{S}_I(\Omega), \Omega) \leq \mathbf{C}(\Omega) - \mathbf{R}_D(\Omega) \tag{3}$$

Theorem 3: A necessary and sufficient condition for a code to achieve the rate tuple $(R_1, \ldots, R_k)$ is that for any cut Ω there exists a linear transformation $\mathcal{T}$ on $\mathcal{E}(\Omega)$ and $\mathcal{F} \subseteq \mathcal{E}(\Omega)$ of size $\mathbf{R}_D$ such that

- The dimension of the vector space spanned by the rows of $\mathcal{F}$ considering only the columns corresponding to the sources in $\mathbf{S}_D$ is $\mathbf{R}_D$.
- The vector space spanned by the rows of $\mathcal{F}$ considering only the columns corresponding to the sources in $\mathbf{S}_I$ is linearly dependent to that in $\mathcal{E} - \mathcal{F}$.
- The linear transformation $\mathcal{T}$ on $\mathcal{E}(\Omega)$ results in $\mathcal{F}$ with columns corresponding to the sources in $\mathbf{S}_I$ nulled out and columns corresponding to the sources in $\mathbf{S}_D$ retaining rank $\mathbf{R}_D$.

In the next section, we show how the presence of crosslinks can be used to tighten upper bounds on achievable rates.

IV. Bound for Networks with Crosslinks

Definition 1 (Crosslink): A crosslink is an edge from source S_i to sink T_j where $i \neq j$.

Networks with a large number of crosslinks arise in quite a few applications such as peer-to-peer file sharing where nodes simultaneously upload and download parts of files, in distributed storage etc. A well-known example is the butterfly network, whose modified versions are shown in Figure 4. Here, $\mathbf{S}_2 \rightarrow \mathbf{T}_1$ in Figure 4a and the $\mathbf{S}_1 \rightarrow \mathbf{T}_2$ in Figure 4b are the crosslinks. These crosslinks help to cancel out the interference at the sinks, but do not contribute directly to useful information flow.

Theorem 4 (Upper bound for networks with crosslinks): For any cut Ω and any partition of $\mathbf{S}_D(\Omega)$ into $\mathbf{S}_{D1}(\Omega)$ and $\mathbf{S}_{D2}(\Omega)$, let $\tilde{\mathbf{C}}(\Omega)$ be the total capacity of the cut after removing all the crosslinks, then

$$\mathbf{R}_D(\Omega) \leq \tilde{\mathbf{C}}(\Omega) + \mathbf{C}_{CL}(\mathbf{S}_{D1}(\Omega), \mathbf{T}_{D2}(\Omega)) \tag{4}$$

where $\mathbf{C}_{CL}(\mathbf{S}_{D1}(\Omega), \mathbf{T}_{D2}(\Omega))$ is the sum capacity of all the crosslinks which originate in $\mathbf{S}_{D1}(\Omega)$ and terminate in any sink corresponding to $\mathbf{S}_{D2}(\Omega)$.

Proof: The set of sources $\mathbf{S}_{D1}(\Omega)$ act as interference for sinks $\mathbf{T}_{D2}(\Omega)$. Hence by Theorem 2,

$$dim(\mathbf{S}_{D1}(\Omega), \Omega) \leq \tilde{\mathbf{C}}(\Omega) - \mathbf{R}_{D2}(\Omega) + \mathbf{C}_{CL}(\mathbf{S}_{D1}(\Omega), \mathbf{T}_{D2}(\Omega)) \tag{5}$$

Since $\mathbf{S}_{D1}(\Omega) \subseteq \mathbf{S}_D(\Omega)$, by Theorem 1 we get

$$\mathbf{R}_{D1}(\Omega) \leq dim(\mathbf{S}_{D1}(\Omega), \Omega) \tag{6}$$

Thus from equations (5) and (6),

$$\mathbf{R}_{D1}(\Omega) \leq \tilde{\mathbf{C}}(\Omega) - \mathbf{R}_{D2}(\Omega) + \mathbf{C}_{CL}(\mathbf{S}_{D1}(\Omega), \mathbf{T}_{D2}(\Omega)) \tag{7}$$

This leads to equation (4). ■

Example 1: Consider the butterfly network in Figure 4a. The cut-set bound gives $\mathbf{R}_1 \leq 1$, $\mathbf{R}_2 \leq 1$ and $\mathbf{R}_1 + \mathbf{R}_2 \leq 2$. Choose a cut Ω as shown in the figure. We get $\mathbf{S}_D(\Omega) = \{\mathbf{S}_1, \mathbf{S}_2\}$, $\mathbf{C}(\Omega) = 2$. Choose $\mathbf{S}_{D1}(\Omega) = \{\mathbf{S}_1\}$. Hence, $\mathbf{S}_{D2}(\Omega) = \{\mathbf{S}_2\}$.

$$\mathbf{C}_{CL}(\mathbf{S}_1(\Omega), \mathbf{T}_2(\Omega)) = 0 \tag{8}$$

Removing the crosslink $\mathbf{S}_2 \rightarrow \mathbf{T}_1$, we get

$$\tilde{\mathbf{C}}(\Omega) = 1 \tag{9}$$

Thus from equation (4), we get

$$\mathbf{R}_1 + \mathbf{R}_2 \leq 1 \tag{10}$$

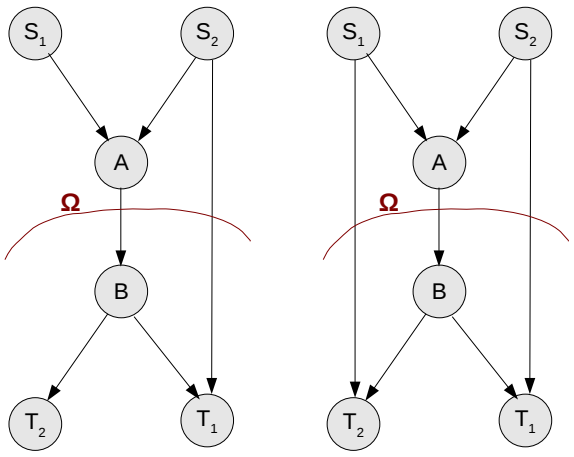

Fig. 4: Two modifications of the butterfly network. Each edge has unit capacity.

Applying this theorem to the remaining cuts and partitions gives $\mathbf{R}_1 \leq 1$ and $\mathbf{R}_2 \leq 1$. Thus, the cut-set bound is tightened in this case.

Example 2: Now consider the butterfly network in Figure 4b. Here, the cut-set bound and the bound given by Theorem 4 coincide to give $\mathbf{R}_1 \leq 1$, $\mathbf{R}_2 \leq 1$ and $\mathbf{R}_1 + \mathbf{R}_2 \leq 2$, which is achievable.

V. Interference Alignment - The crux of the distributed storage solution

In this section we show how the concept of interference alignment introduced in Section III can be put to good use in attacking the distributed storage problem.

Given an optimal construction for $\beta = 1$, optimal constructions for larger values of β can be obtained by partitioning the data into smaller chunks, and encoding them individually using the construction for $\beta = 1$. Hence we consider only $\beta = 1$.

The cut-set bound for the network corresponding to the distributed storage problem gives

$$\beta \geq \frac{\alpha}{d - k + 1} \tag{11}$$

which for $\beta = 1$, yields

$$d \geq \alpha + k - 1 \tag{12}$$

We construct codes achieving this bound for the range of parameters $d \geq 2k - 1$.

Consider the regeneration of a failed systematic node l $(l \in \{1, \ldots, k\})$ using the existing $k-1$ systematic nodes and an arbitrary set of $d - k + 1$ non-systematic nodes $m_1, \ldots, m_{d-k+1}$. The sink l' has d incoming edges each of unit capacity, which constitute the trivial cut across l'(Figure 2). Denote the subspace spanned by the vectors corresponding to these edges by $\mathbf{W}$. $k-1$ of these edges are crosslinks as they come from other systematic nodes and hence cannot carry any *useful information*.

Hence to achieve the lower bound given by equation (12), the remaining set of $d - k + 1 (= \alpha)$ edges (coming from the non-systematic nodes) have to carry all the useful information, and the $k-1$ crosslinks only help to cancel interference. Consider the vectors corresponding to this set of $d - k + 1$ edges. Let $\mathbf{W}_l^{(NS)}$ be the subspace spanned by these vectors considering only the columns corresponding to systematic node l, and $\mathbf{W}_{l^c}^{(NS)}$ be the subspace spanned by these vectors considering only the columns corresponding to all systematic nodes other than l.

Consider the vectors corresponding to the set of the $k-1$ edges from the existing systematic nodes to l'. Let $\mathbf{W}_l^{(C)}$ be the subspace spanned by these vectors considering only the columns corresponding to systematic node l , and let $\mathbf{W}_{l^c}^{(C)}$ be the subspace spanned by these vectors considering only the columns corresponding to all systematic nodes other than l.

The following is an application of Theorem 1.

Lemma 5:

$$dim(\mathbf{W}_l^{(NS)}) = \alpha \tag{13}$$

Proof: Consider the trivial cut Ω across the sink l'. $\mathbf{S}_D(\Omega) = \{S_l\}$, $\mathbf{R}_D(\Omega) = \mathbf{R}_l = \alpha$. Since the $k-1$ edges $r \rightarrow l'$, $r \in \{1, \ldots, k\}$, $r \neq l$ are crosslinks, they have zero components along $\mathbf{S}_l$. Thus

$$dim(\mathbf{W}_l^{(C)}) = 0 \tag{14}$$

Number of edges across this cut which are not crosslinks is $d - (k-1) = \alpha$.

By Theorem 1, we need $dim(\mathbf{S}_l, \Omega) \geq \alpha$. Thus the edges from the non-systematic nodes have to carry linearly independent components along node l. ■

The following is an application of Theorem 2. By projection of $\mathbf{W}$ on a systematic node, we mean the subspace spanned by the vectors corresponding to d edges at sink l' considering only the columns corresponding to that systematic node.

Lemma 6: Projection of $\mathbf{W}$ on any systematic node $r (r \in \{1, \ldots, k\}, \; r \neq l)$ is confined to a subspace of rank atmost one.

Proof: To satisfy Theorem 3, due to equations (13) and (14),

$$\mathbf{W}_{l^c}^{(NS)} \subseteq \mathbf{W}_{l^c}^{(C)} \tag{15}$$

Hence, the projection of $\mathbf{W}_{l^c}^{(NS)}$ on r will also be contained inside the projection of $\mathbf{W}_{l^c}^{(C)}$ on r. Since the vectors carried by the edges $\tilde{r} \rightarrow l'$, for $\tilde{r} = 1, \ldots, k$, $\tilde{r} \neq l$, $\tilde{r} \neq r$ originate at sources other than r, they have zero projections along node r. Thus, dimension of projection of $\mathbf{W}_{l^c}^{(C)}$ on r is atmost 1. Thus, the projection of $\mathbf{W}_{l^c}^{(NS)}$ on r is also confined to at most one dimension. ■

Thus the projections of the vectors corresponding to the edges from the $d - k + 1$ non-systematic nodes to sink l' should be such that their projections on systematic node l must be linearly independent, but on any other systematic must be scalar multiples of each other.

Exact regeneration of a failed systematic node l by connecting to $k-1$ systematic nodes and $d - k + 1$ non-systematic nodes implies that a sink l' connecting to these nodes is able to get its desired sources.

Theorem 7 (Regeneration): A necessary and sufficient

condition for exact regeneration of a failed systematic node l by connecting to the existing $k-1$ systematic nodes and $d-k+1$ non-systematic nodes is that the set of vectors corresponding to the edges from these non-systematic nodes to sink l' satisfy Lemmas 5 and 6.

Proof: Necessity: Follows from rom Lemmas 5 and 6. *Sufficiency*: In the vectors corresponding to edges from non-systematic nodes to sink l', columns corresponding to systematic node r $(r \neq l)$ are confined to one dimension (Lemma 6). This can be nulled out by the vector corresponding to the edge from systematic node r to provide α interference-free vectors. These α interference-free vectors are also linearly independent (Lemma 5). Hence the failed systematic node l can be regenerated exactly. ■

Theorem 8 (Reconstruction): A necessary and sufficient condition for data reconstruction by a DC connecting to k nodes is that the $B \times B$ matrix formed by stacking the $\alpha \times B$ matrices corresponding to the edges from these k nodes to DC is full rank.

Proof: Apply Theorem 1 to the trivial cut across the DC. ■

VI. Code construction

In this section we give a solution to the distributed storage problem for the set of parameters $d \geq 2k-1$. We provide a code construction which achieves the lower bound given in equation (12). Equality in (12) gives $\alpha \geq k$. The code is constructed using the insights obtained in Section V.

Let $M^{(m)}$ $(m = k+1, \ldots, n)$ be the matrix associated with the edge $m_{\text{in}} \rightarrow m_{\text{out}}$. The crux of the solution lies in the construction of these matrices which correspond to the global kernels of the symbols stored in the non-systematic nodes.

Since each source generates α symbols per unit time, and all edges emanating from sources have a capacity of α, we set the matrices associated with all such edges to be I_α. Thus, the matrices corresponding to incoming edges at any non-systematic node m has full rank B. This allows $M^{(m)}$ to take any value as the only restriction of $M^{(m)}$ is that it should be a linear combination of the matrices corresponding to incoming edges at m_{in}.

To satisfy Theorems (7) and (8) we set

$$\mathbf{M}^{(m)}= \begin{bmatrix} \lambda_{1,1}^{(m)}\underline{h}_{1,1}^{(m)} & \lambda_{1,2}^{(m)}\underline{h}_{1,2} & \cdots & \lambda_{1,k}^{(m)}\underline{h}_{1,k} \\ \lambda_{2,1}^{(m)}\underline{h}_{2,1} & \lambda_{2,2}^{(m)}\underline{h}_{2,2}^{(m)} & \cdots & \lambda_{2,k}^{(m)}\underline{h}_{2,k} \\ \vdots & \vdots & & \vdots \\ \lambda_{k,1}^{(m)}\underline{h}_{k,1} & \lambda_{k,2}^{(m)}\underline{h}_{k,2} & \cdots & \lambda_{k,k}^{(m)}\underline{h}_{k,k}^{(m)} \\ \lambda_{k+1,1}^{(m)}\underline{h}_{k+1,1}^{(m)} & \lambda_{k+1,2}^{(m)}\underline{h}_{k+1,2}^{(m)} & \cdots & \lambda_{k+1,k}^{(m)}\underline{h}_{k+1,k}^{(m)} \\ \vdots & \vdots & & \vdots \\ \lambda_{\alpha,1}^{(m)}\underline{h}_{\alpha,1}^{(m)} & \lambda_{\alpha,2}^{(m)}\underline{h}_{\alpha,2}^{(m)} & \cdots & \lambda_{\alpha,k}^{(m)}\underline{h}_{\alpha,k}^{(m)} \end{bmatrix} \quad (16)$$

where $\underline{h}_{i,j}^{(m)}$ is a row vector of length α, and $\lambda_{i,j}^{(m)}$ is a scalar whose values will be determined subsequently.

Regeneration: For regeneration of systematic node l, for all the sinks (new nodes) l' which connect to non-systematic node m, the vector associated with $m_{out} \rightarrow l'$ is set to l^{th} row of $\mathbf{M}^{(m)}$.

Observe that in row l of $\mathbf{M}^{(m)}$, the vectors $\underline{h}_{l,j} \forall j \neq l$ are independent of m. Hence the projections of the set of vectors corresponding to the edges from the $d-k+1$ non-systematic nodes to sink l' on any systematic node $r \neq l$ are just scalar multiples of each other, satisfying Lemma 6. The interference can be cancelled at sink l' by setting the vector corresponding to edge $r \rightarrow l'$ to a vector with $\underline{h}_{l,r}$ as its r^{th} component and $\underline{0}$ elsewhere, $\forall r \in \{1, \ldots, l-1, l+1, \ldots, k\}$.

the edge from systematic node r to sink l' carries the a vector which has $\underline{h}_{l,r}$ as its r^{th} component and $\underline{0}$ elsewhere.

To satisfy Lemma 5, we need

$$det \begin{pmatrix} \lambda_{l,l}^{(m_1)}\underline{h}_{l,l}^{(m_1)} \\ \vdots \\ \lambda_{l,l}^{(m_\alpha)}\underline{h}_{l,l}^{(m_\alpha)} \end{pmatrix} \neq 0 \quad (17)$$

for all systematic nodes l and all combinations of the non-systematic nodes $m_1, \ldots, m_\alpha$. This is a set of polynomials in $\underline{h}$ and λ as variables which are clearly not identically zero.

Reconstruction: For reconstruction, at any DC, the $B \times B$ matrix formed by stacking the k $\alpha \times B$ matrices corresponding to the incoming edges should be full rank. The determinants of these matrices will also be polynomials in $\underline{h}$ and λ as variables, which can be shown to be not identically zero.

Hence, assignment of values to the variables satisfying all the above conditions (polynomials evaluating to non-zero values) can be obtained using the method given by Koetter and Medard [6]. An explicit code for this problem is provided in [7].

References

[1] Y. Wu, A. G. Dimakis and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. Allerton Conference on Control, Computing and Communication*, (Urbana-Champaign, IL), September 2007.

[2] K. V. Rashmi, Nihar B. Shah, P. Vijay Kumar and Kannan Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. Allerton Conf.,* September 2009.

[3] Y. Wu and A. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. ISIT,* July 2009.

[4] A. R. Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proc. Allerton Conf.,* October 2003.

[5] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," in *IEEE Transactions on Information Theory,* vol. 51, no. 8, pp. 2745-2759, Aug. 2005.

[6] Ralf Koetter and Muriel Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking,* v.11 n.5, p.782-795, Oct. 2003.

[7] Nihar B. Shah, Rashmi K. V., P. Vijay Kumar and Kannan Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," to appear in *Proc. of Information Theory Workshop,*, Cairo, January 2010.