Introduction to Information and coding theory

Lecture 10

Lecturer: Haim Permuter *Scribe: Maxim Lvov*

I. Multiple Source Network

In previous lectures, we've seen that for a network, with one source-one destination, the capacity (which is the maximum transmission rate) is given by the Max-Flow Min-Cut theorem. If we had multiple destinations (but only one source), the capacity was given by the formula

$$C = \min_{i\in\mathcal{D}} \min_{\substack{S:\\ 1\in S, i\in S^c}} \left\{ \sum_{j\in S, k\in S^c} C_{jk} \right\} \tag{1}$$

where C is the capacity of the network, $\mathcal{D}$ is the set of all destinations, S is a subset of nodes containing the source, and C_{jk} is the capacity of the edge between the nodes j and k.

Now we consider the case, with multiple sources, where each source sends information with a rate R_j (j is the number of source).

Definition 1 (Multiple Source - Multiple Destination Network) A multiple sources - multiple destinations network contains:

- $\mathcal{V}$ - A set of nodes, where each node receives and transmits messages.
- $\mathcal{E}$ - Set of edges between the nodes. An edge between the nodes i and j is denoted by (i,j)
- $\mathcal{C}$ - Set of capacities of each edge.
- $\mathcal{X}_{(i,j)} = \{0,1\}^{C_{(i,j)}}$ - The alphabet sent at the edge (i,j).
- $\mathcal{S}$ - Set of all source nodes.
- $\mathcal{D}$ - Set of all destination nodes.

Every node $i \in \mathcal{V}$ can send $C_{(i,j)}$ bits on the error-free link (i,j) in a unit time, and these bits are functions of:

- The bits received at the input links to node (i)
- The bits node (i) generates itself (if this is a source node).

Whenever a destination d node wants to decode a message M sent by a source s, the decoding is based on both the bits received at the input links of d, and on the bits generated at the node d itself (only if d is a source node).

We also denote by:

- $\mathcal{I}(i) =$ set of all edges entering node i.

- $\mathcal{O}(i)$ = set of all edges getting out from node i.

You can see an example of such a network in Fig.I. The capacity of each edge is written near it. Each

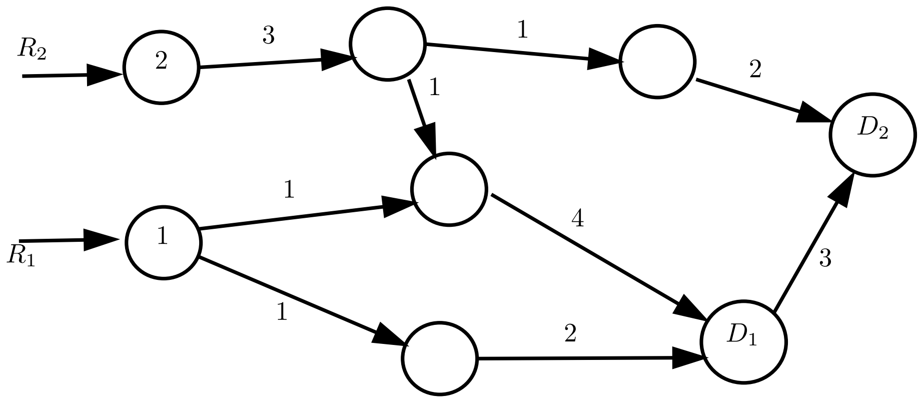

Fig. 1. Diagram that shows a network with 2 sources and 2 destinations.

source generates a sequence of bits it wants to transmit to some set of destination nodes. Note that not necessarily every destination needs to decode the messages from all sources. For example, in the network shown in Fig. I, the destination D_1 may want the bits generated only at the source 2, while destination D_2 may want only the bits generated at the source 1. In case of a multiple source-multiple destination network, there is no longer a "capacity" number C, but rather a *capacity region* which we shall explain now.

Definition 2 (Network Code) For a given multiple source - multiple destination network with $|\mathcal{S}|$ sources and $|\mathcal{D}|$ destinations, a network code $(n, 2^{nR_1}, ..., 2^{nR_{|\mathcal{S}|}})$ consists of:

- $|\mathcal{E}|$ encoding functions $f_{(i,j)} : \left\{\mathcal{X}_{(k,i)}|(k,i) \in \mathcal{I}(i)\right\}^n \times \left\{1, ..., 2^{nR_i}\right\} \to \mathcal{X}^n_{(i,j)}$ where R_i is the rate of the generated bits at node i ($R_i = 0$ if $i \notin S$)
- $|\mathcal{D}| \cdot |\mathcal{S}|$ decoding functions $g_{d,s} : \left\{\mathcal{X}_{(i,d)}|(i,d) \in \mathcal{I}(d)\right\}^n \times \left\{1, ..., 2^{nR_d}\right\} \to \left\{1, ..., 2^{nR_s}\right\}$ for every $d \in \mathcal{D}$ and $s \in \mathcal{S}$.

Definition 3 (Error event) For a destination $d \in \mathcal{D}$ and a source $s \in \mathcal{S}$ we define the probability $P^n_{e,(d,s)}$ as the probability that d hasn't recovered correctly the message sent by the source s:

$$P^n_{e,(d,s)} = \Pr\{g_{d,s} \neq M_s\} \tag{2}$$

Definition 4 (Achievable Rate) Given a multiple source-multiple destinations network, where the rate of each source is R_j (j is the number of source), then the n-tuple $(R_1, .., R_k)$ is called achievable if there exists a sequence of codes $(n, 2^{nR_1}, ..., 2^{nR_k})$, such that the probability of error for each destination $P^n_{e,(d,s)}$

goes to zero when $n \to \infty$, for all $s \in \mathcal{S}$. Note: when destination d doesn't want the message sent by the source s, we don't need the above limit to hold, that is, we don't need $P^n_{e,(d,s)} \to 0$.

It is easy to see, that if the n-tuple $(R_1, .., R_k)$ is achievable, then the n-tuple $(R_1^*, .., R_k^*)$ is also achievable, where $R_j^* \leq R_j$, but not vise-versa! We can define the *capacity region* as the closure of the set of all achievable n-tuples $(R_1, .., R_k)$.

Definition 5 (Capacity Region) Capacity Region is the set of all n-tuples $(R_1^*, .., R_k^*)$, such that for every $\epsilon > 0$ there exists an achievable n-tuple $(R_1, .., R_k)$ such that

$$\|(R_1^*, .., R_k^*) - (R_1, .., R_k)\| < \epsilon \tag{3}$$

Where $\| \cdot \|$ is some norm defined on $\mathbb{R}^k$

In the case of single source, the capacity region was the set $[0, C]$, where C was the supremum over all of the achievable rates, but in the case of multiple sources, the capacity region is a high dimensional set. For example, in the case of 2 sources, the capacity region can always be described as a convex set in $\mathbb{R}^2$, as shown in Fig. 2.

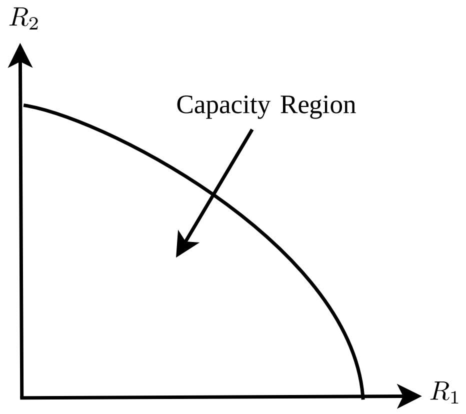

Fig. 2. Achievable rates and Capacity Region.

We will now limit us, to the case where there is only one destination, that should get all of the messages sent by all sources, and there are two sources in the network (for more sources, the problem and the solution is easily extended). We want to find the capacity region in that case. For that, we have the next theorem.

Theorem 1 (Capacity Region for two sources) For a given ”two sources - single destination network”, the capacity region is the set of all pairs (R_1, R_2) that satisfy the next equations:

$$R_1 + R_2 \leq C_{sum} \triangleq \min_S \left\{ \sum_{j \in S, k \in S^c} C_{jk} | 1, 2 \in S, N \in S^c \right\} \tag{4}$$

$$R_1 \leq C_1 \triangleq \min_S \left\{ \sum_{j \in S, k \in S^c} C_{jk} | 1 \in S, N \in S^c \right\} \tag{5}$$

$$R_2 \leq C_2 \triangleq \min_S \left\{ \sum_{j \in S, k \in S^c} C_{jk} | 2 \in S, N \in S^c \right\} \tag{6}$$

where N is the destination node, and $1, 2$ are the source nodes.

Proof: We'll first prove the converse, i.e., equations $(4) - (6)$ must be satisfied if the pair (R_1, R_2) is achievable. The proof is by contradiction: suppose first that the pair (R_1, R_2) is achievable and (4) isn't satisfied. Then look at another network, shown in Fig.3 (this is a single source-single destination network). This network is created, by taking the old network we had (with two sources), and adding a ”zero source”

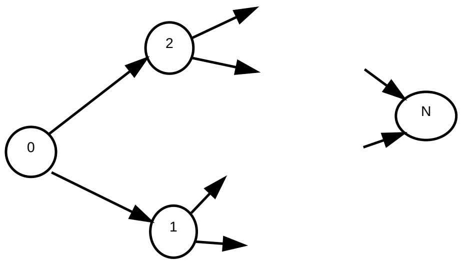

Fig. 3. A new network, created from the old one, by adding the ”zero” source to the left

to the left. The zero source have two edges going from it, to nodes 1 and 2. Each such edge have infinite capacity. The rest of the network, is as before. By the Max-Flow Min-Cut theorem, the capacity of the new network, is given by:

$$C = \min_S \left\{ \sum_{j \in S, k \in S^c} C_{jk} | 0 \in S, N \in S^c \right\} \tag{7}$$

$$\overset{(a)}{=} \min_S \left\{ \sum_{j \in S, k \in S^c} C_{jk} | 0, 1, 2 \in S, N \in S^c \right\} \tag{8}$$

$$\overset{(b)}{=} \min_S \left\{ \sum_{j \in S, k \in S^c} C_{jk} | 1, 2 \in S, N \in S^c \right\} \tag{9}$$

$$= C_{sum} \tag{10}$$

Where:

- (a) follows from the fact, that in order to achieve the minimum, both nodes 1 and 2 should be included in S (otherwise the sum will be infinite, since then at least one of the new edges with infinite capacities will be in the cut between S and S^c).
- (b) follows from the fact, that if $1, 2 \in S$ and $0 \in S^c$ then the sum will again be infinite (since both of the new edges with infinite capacities will be in the cut), and therefore the minimum in (9) is achieved only when $0 \in \mathcal{S}$.

On the other hand, since (R_1, R_2) is achievable in the original network (by assumption), we can construct the next code (in the new network) with a rate $R = R_1 + R_2 > C_{sum}$ by sending R_1 bits per second on the (0,1) edge, and R_2 bps on the (0,2) edge. The rest of the network coding is left as in the original case (with two sources). This way, we've constructed a code in the new network (with one source) that allows transmission with a rate higher than the capacity, which is impossible. Therefore, the assumption that (4) isn't satisfied was wrong. The other equations (5) and (6) if we assume that the sources dont share their messages, since we can look at the problem as a one-source one-destination network, with one of the sources that acts as a real source and the other source acts as an ordinary node. In that case, we know from (1) that (5) and (6) must hold.

Achievability: To show achievability, we first assume that (R_1, R_2) satisfy the above inequalities $(4)-(6)$, and we construct a code $(n, 2^{nR_1}, 2^{nR_2})$ for the network. First of all, we expand the original network to the one given in Fig.3, but this time, the capacities $C_{(0,1)}, C_{(0,2)}$ are

$$C_{(0,1)} = R_1, C_{(0,2)} = R_2 \tag{11}$$

By the Max-Flow Min-Cut theorem, the capacity of this network given by the

$$C = \min_S \left\{ \sum_{j \in S, k \in S^c} C_{jk} | 0 \in S, N \in S^c \right\} \tag{12}$$

$$\overset{(a)}{=} \left\{ \sum_{j \in S, k \in S^c} C_{jk} | S = \{0\} \right\} \tag{13}$$

$$= R_1 + R_2 \tag{14}$$

We explain why (a) is true: if S contained both 1 and 2 (and maybe any other nodes), then the sum in (12) wouldn't be below C_{sum} (by definition of C_{sum}), and therefore it wouldn't be below $R_1 + R_2$. If S contained only one of the them (suppose $0, 1 \in S$ and $2 \in S^c$) then the sum in (12) would be:

$$\sum_{j \in S, k \in S^c} C_{jk} \overset{(a)}{=} R_2 + \sum_{0 \neq j \in S, k \in S^c} C_{jk} \geq \tag{15}$$

$$\overset{(b)}{\geq} R_2 + R_1 \tag{16}$$

where:

- (a) follows from the fact that the only edge connecting the source 0 to S^c is the edge (0,2).
- (b) follows from equation (5) and the fact that $1 \in S$.

Since the capacity of the new one-source one-destination network is $R_1 + R_2$, there exists a code that allows the source 0 to send information with that rate. The only possible way for this code to work, is if the source 0 sends R_1 bps on the edge $(0, 1)$ and R_2 bps on the edge $(0, 2)$.
Therefore, we can take the new network, omit the 0 source, and provide R_1 bps to node 1, and R_2 bps to node 2, and the rest of the code is left as in the new network. In that way, the destination N will receive both massages without error.

Therefore (R_1, R_2) is achievable. ■

II. Linear Network Coding Algorithm

In the previous lesson, we've seen that in a network with a single source and multiple destinations, where all links capacities equal 1, we can use linear network coding to reach the capacity of the network. We recall that in linear network coding, each node t receives $|\mathcal{I}(t)|$ messages $(m_1, .., m_{|}\mathcal{I}(t)|)$, and sends on its outputs linear combinations of these messages. Every message m_j is considered as a scalar from the finite field F_{2^n} represented by n bits. An example for such network is given in Fig. 4. If we assume that only one bit can be sent on every link in a unit time, then it will take n time units to send a scalar on every link.

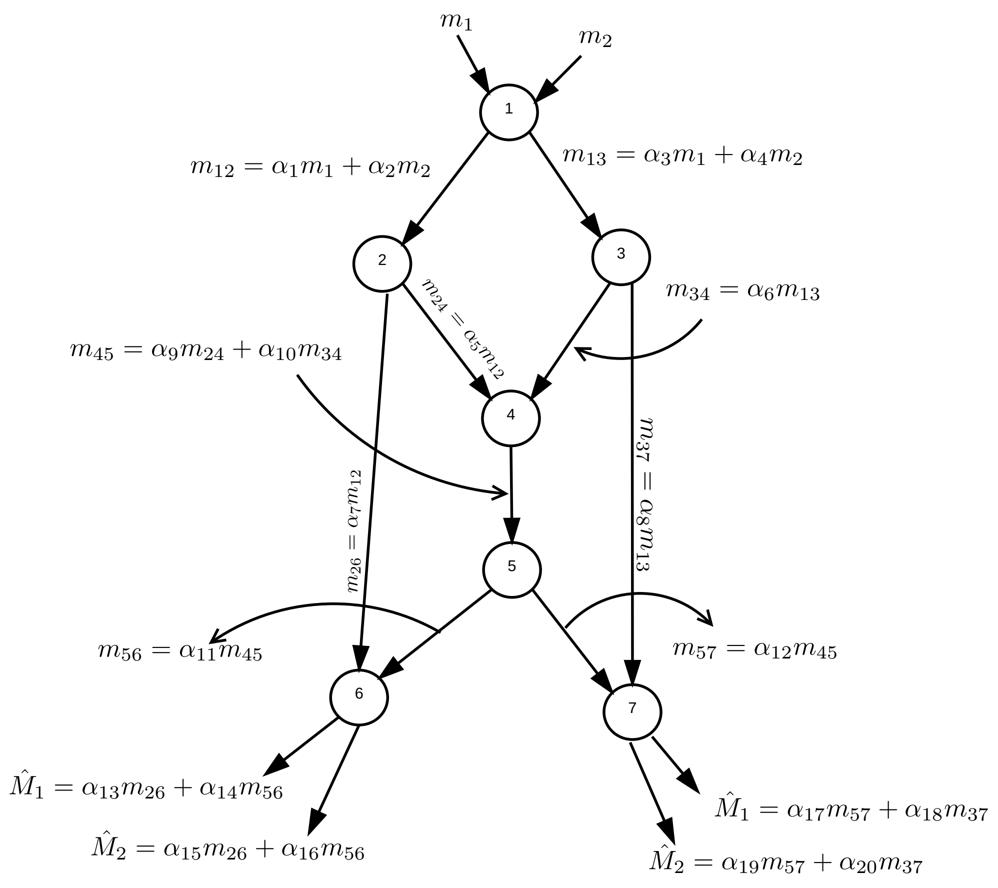
m_1
m_2
1
$m_{12} = \alpha_1 m_1 + \alpha_2 m_2$
$m_{13} = \alpha_3 m_1 + \alpha_4 m_2$
2
3
$m_{24} = \alpha_5 m_{12}$
$m_{34} = \alpha_6 m_{13}$
$m_{45} = \alpha_9 m_{24} + \alpha_{10} m_{34}$
4
$m_{26} = \alpha_7 m_{12}$
$m_{37} = \alpha_8 m_{13}$
5
$m_{56} = \alpha_{11} m_{45}$
$m_{57} = \alpha_{12} m_{45}$
6
7
$\hat{M}_1 = \alpha_{13} m_{26} + \alpha_{14} m_{56}$
$\hat{M}_2 = \alpha_{15} m_{26} + \alpha_{16} m_{56}$
$\hat{M}_1 = \alpha_{17} m_{57} + \alpha_{18} m_{37}$
$\hat{M}_2 = \alpha_{19} m_{57} + \alpha_{20} m_{37}$

Fig. 4. Linear network coding. Every output of each node is a linear combination of its inputs

For now, we'll limit ourself to deal with networks that don't contain cycles. A cycle, is a situation, when the edges in the network form a directed loop, as shown in Fig.5. In such networks, it is always possible to write the network from top to bottom, as in Fig.4, where every edge points down. Moreover, the messages received by the destinations are linear combinations of the messages sent by the sources. More precisely,

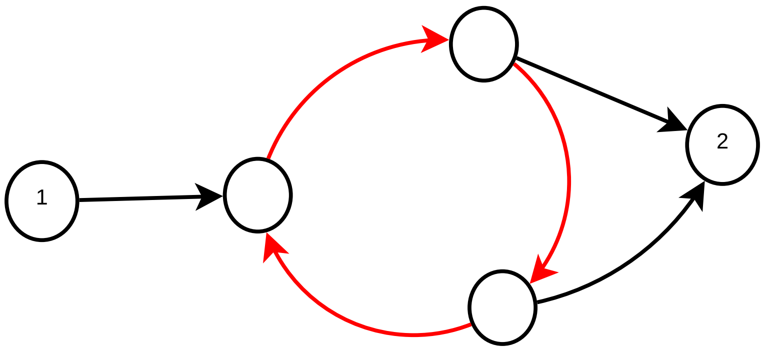

Fig. 5. A cycle in the network is shown in red.

if we write the messages sent by the sources in a column vector $(M_1, M_2, ..., M_R)^T$, and the messages received by destination $d \in \mathcal{D}$ in a column vector $(\hat{M}_1^d, \hat{M}_2^d, ..., \hat{M}_R^d)^T$, then the following relation holds:

$$\begin{pmatrix} \hat{M}_1^d \\ \hat{M}_2^d \\ .. \\ \hat{M}_R^d \end{pmatrix} = G_d(\alpha_1, .., \alpha_q) \begin{pmatrix} M_1 \\ M_2 \\ .. \\ M_R \end{pmatrix} \tag{17}$$

The matrix $G_d(\alpha_1, .., \alpha_q)$ is called a *global encoding kernel matrix*, and it depends on the coefficients $\{\alpha_i\}$ and on the network structure. We want that G_d will be the identity matrix, or at least, an invertible matrix, so the destination will be able to decode the messages $(M_1, M_2, ..., M_R)^T$. We've shown in the previous lesson, that there exists an n, and scalars $\{a_i\}$ such that the destinations can decode the messages without error, or equivalently, that the matrices G_d are invertible. We show now two algorithms for finding these scalars. The first algorithm is based on choosing random coefficients, and with high probability, to have a good network (the destinations will be able to decode the messages). The second algorithm will be a deterministic one.

A. The First Algorithm (Random)

As mentioned before, we choose a relatively large n, and then choose the coefficients $\{a_i\}$ randomly from the set $\{0, .., 2^n - 1\}$ independently of each other, with uniform distribution. If n is chosen large enough (comparing to $|\mathcal{E}|$), the probability to have invertible global encoding kernel matrices is almost one.

Example 1 (Global encoding kernel matrices) We find the global encoding kernel matrices for destination 7 in the network given in for the network given in Fig. 4:

$$\begin{pmatrix}\hat{M}_1\\ \hat{M}_2\end{pmatrix} = \begin{pmatrix}\alpha_{17} & \alpha_{18}\\ \alpha_{19} & \alpha_{20}\end{pmatrix}\begin{pmatrix}m_{57}\\ m_{37}\end{pmatrix} \tag{18}$$

$$\begin{pmatrix}m_{57}\\ m_{37}\end{pmatrix} = \begin{pmatrix}\alpha_{12} & 0\\ 0 & \alpha_{8}\end{pmatrix}\begin{pmatrix}m_{45}\\ m_{13}\end{pmatrix} \tag{19}$$

$$\begin{pmatrix}m_{45}\\ m_{13}\end{pmatrix} = \begin{pmatrix}\alpha_{9} & \alpha_{10} & 0 & 0\\ 0 & 0 & \alpha_3 & \alpha_4\end{pmatrix}\begin{pmatrix}m_{24}\\ m_{34}\\ m_1\\ m_2\end{pmatrix} \tag{20}$$

$$\begin{pmatrix}m_{24}\\ m_{34}\\ m_1\\ m_2\end{pmatrix} = \begin{pmatrix}\alpha_5 & 0 & 0 & 0\\ 0 & \alpha_6 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}m_{12}\\ m_{13}\\ m_1\\ m_2\end{pmatrix} \tag{21}$$

$$\begin{pmatrix}m_{12}\\ m_{13}\\ m_1\\ m_2\end{pmatrix} = \begin{pmatrix}\alpha_1 & \alpha_2\\ \alpha_3 & \alpha_4\\ 1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}m_1\\ m_2\end{pmatrix} \tag{22}$$

The global encoding kernel matrix is:

$$G_7 = \begin{pmatrix}\alpha_{17} & \alpha_{18}\\ \alpha_{19} & \alpha_{20}\end{pmatrix}\begin{pmatrix}\alpha_{12} & 0\\ 0 & \alpha_{8}\end{pmatrix}\begin{pmatrix}\alpha_{9} & \alpha_{10} & 0 & 0\\ 0 & 0 & \alpha_3 & \alpha_4\end{pmatrix}\begin{pmatrix}\alpha_5 & 0 & 0 & 0\\ 0 & \alpha_6 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\alpha_1 & \alpha_2\\ \alpha_3 & \alpha_4\\ 1 & 0\\ 0 & 1\end{pmatrix} \tag{23}$$

$$= \begin{pmatrix}\alpha_{17} & \alpha_{18}\\ \alpha_{19} & \alpha_{20}\end{pmatrix}\begin{pmatrix}\alpha_{12} & 0\\ 0 & \alpha_{8}\end{pmatrix}\begin{pmatrix}\alpha_9\alpha_1\alpha_5+\alpha_{10}\alpha_3\alpha_6 & \alpha_9\alpha_2\alpha_5+\alpha_{10}\alpha_4\alpha_6\\ \alpha_3 & \alpha_4\end{pmatrix} \tag{24}$$

$$= \begin{pmatrix}\alpha_1\alpha_5\alpha_9\alpha_{12}\alpha_{17}+\alpha_3\alpha_6\alpha_{10}\alpha_{12}\alpha_{17}+\alpha_3\alpha_8\alpha_{18} & \alpha_2\alpha_5\alpha_9\alpha_{12}\alpha_{17}+\alpha_4\alpha_6\alpha_{10}\alpha_{12}\alpha_{17}+\alpha_4\alpha_8\alpha_{18}\\ \alpha_1\alpha_5\alpha_9\alpha_{12}\alpha_{19}+\alpha_3\alpha_6\alpha_{10}\alpha_{12}\alpha_{19}+\alpha_3\alpha_8\alpha_{20} & \alpha_2\alpha_5\alpha_9\alpha_{12}\alpha_{19}+\alpha_4\alpha_6\alpha_{10}\alpha_{12}\alpha_{19}+\alpha_4\alpha_8\alpha_{20}\end{pmatrix} \tag{25}$$

To check if G_7 is invertible, simply take the determinant of it, and check if its zero.

Theorem 2 (Probability of error for random network coding) In a one source - multiple destinations network with the coefficients $\{\alpha_1, ..., \alpha_q\}$ chosen randomly and independently with uniform distribution, the probability that at least one global encoding kernel matrix G_d won't be invertible goes to zero as n tends to infinity, provided that the rate $R \leq C$.

Proof: By the construction of the code, it can be seen that the entries in G_d are polynomials in $(\alpha_1, .., \alpha_q)$. To check if the matrix G_d is invertible, we look at the determinant of G_d. This determinant is a polynomial $f(\alpha_1, ..., \alpha_q)$ with coefficients from the set $\{0, 1\}$. Note, that we treat $(\alpha_1, ..., \alpha_q)$ as variables, since they are not chosen yet. Now there are two cases:

- Case I: The determinant $f(\alpha_1, ..., \alpha_q)$ is the zero polynomial (and hence, independent of $\{\alpha_i\}$). This case cannot happen if $R \leq C$. To see why this is true, recall that in a single source - single destination network, it is possible to transmit bits by simply routing them at different paths, to reach the destination. By this way, it is possible to transmit bits at a rate equal to the capacity of the network (this was proved for one source - one destination topology). This simple routing scheme, is just a special case of linear network coding, when n is taken to be 1 (and the field F_2 consists of the scalars $\{0, 1\}$). In our case, we may have multiple destinations, but since we examine only one destination now, we can treat it as a one destination network. For that destination, the polynomial $f(\alpha_1, ..., \alpha_q)$ cannot be the zero polynomial, since we know that there exist coefficients $\alpha_1, ..., \alpha_q \in F_2$ such that the matrix G_d will be invertible.
- Case II: The determinant $f(\alpha_1, ..., \alpha_q)$ is not the zero polynomial. In that case, it can be shown, that if the coefficients $\{\alpha_i\}$ are chosen i.i.d from F_{2^n} then

$$\lim_{n \to \infty} \Pr(\{f(\alpha_1, ..., \alpha_q) = 0\}) = 0 \tag{26}$$

The above analysis was done for one destination, in a network that may contain many (but a finite amount of) destinations. Since for every destination, the probability in (26) tends to zero, the overall probability (that at least one matrix G_d isn't invertible) also tends to zero. ■

To see why (26) is true, we have the following corollary

Corollary 1 Let $f(\alpha_1, ..., \alpha_q)$ be a nonzero polynomial with coefficients in a field F, where $|F|$ is greater then the highest degree of f in α_i $(1 \leq i \leq q)$. If $(\alpha_1, ..., \alpha_q)$ are drown independently with uniform distribution from F, then we have

$$\Pr\{f(\alpha_1, ..., \alpha_q) \neq 0\} \geq \left(1 - \frac{m}{|F|}\right)^q \tag{27}$$

Example 2 (Zeros of polynomials) Lets look at the polynomial

$$g(x) = x^3 + 2x^2 + x + 1 \tag{28}$$

where x is chosen with uniform distribution from F. Since g has at most 3 roots (it may have less in case of degenerate), the probability that g wont be zero is

$$P \geq 1 - \frac{3}{|F|} \tag{29}$$

B. Initializing the network

If we use the above algorithm to choose the coefficients randomly, we need to know the global encoding kernel matrices in order to decode the messages. If the network is large, it can be difficult to learn the network. Instead, we can send pilot messages to learn the matrices G_d. The source should first send the messages

$$\begin{pmatrix} M_1 \\ M_2 \\ .. \\ M_R \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ .. \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ .. \\ 0 \end{pmatrix}, .., \begin{pmatrix} 0 \\ 0 \\ .. \\ 1 \end{pmatrix} \tag{30}$$

Every time, when the source sends the pilot message $\mathbf{m} = \mathbf{e_i} (1 \leq i \leq R)$, every destination d will receive the i column of its matrix G_d. After it receives all of it's columns, it can put all vectors together to get the matrix G_d.

C. The Second Algorithm (Deterministic)

Before we show a deterministic algorithm for finding the coefficients $\{\alpha_1, .., \alpha_q\}$, we should recall a statement that will help understand the algorithm.

Proposition 1 (Edge disjoint paths) In a single source network with all capacities equal to 1, the maximum number of edge-disjoint paths from the source node S to node t equals to the maximum flow from S to t (which is also equal to the capacity of the network between S and t).

We also recall the statement that the network capacity for a single source network, is given by (1). If we combine these two facts, we get to the conclusion, that for a single source - multiple destination network, there can be found at least C edge-disjoint paths from the source node, to each destination, where C is the capacity of the network. Note that for each destination, the paths from S to that destination are disjoint, but if we consider two paths for different destinations, they aren't necessary disjoint. If so, we can now consider the network, that consists only from these paths (we can ignore edges that don't belong to any such path, since we can reach the capacity even without them). So the first step in the algorithm, is to find C edge-disjoint path's from the source node, to each destination, and then to "delete" all remaining edges (that don't belong to any path). After doing so, we can order the nodes in our network, in a way, so that each path go's from top to bottom (like in Fig. 4). Now we shall explain, how the assignment of $\{\alpha_1, .., \alpha_q\}$ will be done. We'll start with a lemma that will give us intuition on how to assign the coefficients:

Lemma 1 (Global linear encoding) Consider a linear network coding network with one source s, and some destination d, which has R edge-disjoint paths going from s to d. On every path $r \in \{1, ..., R\}$, choose an arbitrary edge i_r. Then the following two statements are true:

1) For every edge $e_{(i_r)}$ you've chosen, the message that is sent over it $\tilde{m}_i$ is a linear combination of the original messages $m_1, .., m_R$, that is

$$\tilde{m}_i = g_{i,1} m_1 + ... + g_{i,R} m_R, \qquad i = 1, 2, .., R \tag{31}$$

2) If all original messages can be decoded at the destination node d, then all R equations (31) are linearly independent.

Proof:

1) This is obvious, since we use linear network coding.
2) If the equations would be linearly dependent, we couldn't decode the original messages from $\tilde{m}_1, .., \tilde{m}_R$, and therefore, we couldn't decode them from the information that d receives, since this information is a function of $(\tilde{m}_1, .., \tilde{m}_R)$.

■

The above lemma, gives us an intuition, to how should we assign the coefficients $\{\alpha_1, .., \alpha_q\}$ - we shall assign them in such a way, so that if we choose R arbitrary edge's on different paths, then the equations they are made of, should be linearly independent. We will do this in a recursive way: we'll start from the top edges, and go down, each time assigning the coefficients in the following way: for every edge, look at the node that it goes from, and at the edge's entering that node. Now choose a linear combination of the messages on these edges, such that the new equation will be linearly independent with all other equations for the messages from other paths, for all of the destinations.

We'll summarize the algorithm:

Algorithm 1 Deterministic algorithm for assigning local encoding kernels

1) Calculate the capacity C of the network, using equation (1), and choose a rate $R \leq C$.
2) Order the nodes of the network in a sequence such that if there is an edge from node i to node j, then node i appears before node j in the sequence. Such a sequence is called *upstream-to-downstream order*.
3) For every destination $d \in \mathcal{D}$, find R edge-disjoint paths going from the source node S to d, and write all of them, so that you'll be able to see, which edges belong to every path. Write them in the following way: $P_d(r) =$ a sequence of all edges in the path r from the source to node d. Delete all of the edge's, that don't belong to any path.
4) For every edge (i, j) in your network, write a vector $r_{(i,j)} = [b_1, .., b_{|\mathcal{D}|}]$, where b_k is the number of path that goes through it, from S to d_k. If there is no such path, then $b_k = 0$.
5) For every destination $d \in \mathcal{D}$, create a global encoding kernel matrix G_d, which we'll initialize as the Identity matrix, with R raws and columns. These matrices will contain the coefficients g_{il}, given in (31), and they'll change as we go from top to bottom.
6) Add R virtual edges entering the source (with no origin). Near every virtual edge e_i write a raw vector $\mathbf{v_i}$ which is the i'th vector in the standard basis of the vector space F^R.
7) Now start going over all nodes by the upstream-to-downstream order. For every node t, and for every edge $e \in \mathcal{O}(t)$, do the following:
   - Look at all the vectors $\mathbf{f_l}$ near every edge $l \in \mathcal{I}(t)$. Find a linear combination of these vectors:
   $$\mathbf{f_e} = \sum_{\mathbf{l} \in \mathcal{I}(\mathbf{t})} \alpha_{\mathbf{e},\mathbf{l}} \mathbf{f_l} \tag{32}$$
   That will satisfy the next condition: for every destination $d \in \mathcal{D}$, if it has a path r going through the edge e, then the new matrix G_d obtained from the old one by replacing its raw r by $\mathbf{f_e}$ should be invertible. It can be shown that if $|F| > |\mathcal{D}|$ then such linear combination can be found, with $\alpha_{e,l} \in F, \forall (e, l)$.
   - Write the vector $\mathbf{f_e}$ near the edge e, and replace it with the corresponding raws in the matrices $\{G_d\}_{d \in \mathcal{D}}$, as described above. Note: the vectors $\{\mathbf{f_e}\}_{\mathbf{e} \in \mathbf{E}}$ are called *global encoding kernels*, and they satisfy
   $$x_e = \mathbf{f_e} \cdot (m_1, ..., m_R)^T \tag{33}$$
   where x_e is the symbol sent at edge e.
8) At the end of the process, in order to obtain the transmitted symbols $(m_1, ..., m_R)$, every destinations solves
$$\begin{pmatrix} m_1 \\ m_2 \\ .. \\ m_R \end{pmatrix} = (G_d)^{-1} \begin{pmatrix} \tilde{m}_1 \\ \tilde{m}_2 \\ .. \\ \tilde{m}_R \end{pmatrix} \tag{34}$$
where $\{\tilde{m}_1, ..., \tilde{m}_R\}$ are the symbols received at the destination.

Example 3 (Assigning local encoding kernels) As an example, we can consider the network given in Fig. 4. We'll follow the above steps, to find $\alpha_1, .., \alpha_{20}$.

1) The capacity of this network is C=2 (you can calculate it, as an exercise). Thats also the reason, why we included $R = 2$ packets m_1, m_2 in the source.
2) Already ordered from top to bottom in Fig.4.
3) There are two edge-disjoint path's from source node 1 to node 6, and also to node 7.

   There are no "useless nodes" in this example, so there is nothing to "delete". The paths going from $S = 1$ to $d_1 = 6$ and to $d_2 = 7$ are:

$$\begin{aligned} &\{P_1(1) = (1, 2, 6), P_1(2) = (1, 3, 4, 5, 6)\} \\ &\{P_2(1) = (1, 3, 7), P_2(2) = (1, 2, 4, 5, 7)\} \end{aligned} \tag{35}$$

4) The edge-vectors are:

$$\begin{aligned} &r_{(1,2)} = [1, 2], r_{(1,3)} = [2, 1], r_{(2,4)} = [0, 2], \\ &r_{(3,4)} = [2, 0], r_{(2,6)} = [1, 0], r_{(4,5)} = [2, 2], \\ &r_{(1,2)} = [1, 2], r_{(5,6)} = [2, 0], r_{(5,7)} = [0, 2] \end{aligned} \tag{36}$$

5) We create the global encoding kernel matrices:

$$G_1 = \begin{pmatrix} 10 \\ 01 \end{pmatrix}, G_2 = \begin{pmatrix} 10 \\ 01 \end{pmatrix} \tag{37}$$

6) We add the virtual edges with the standard basis vectors written close to them.

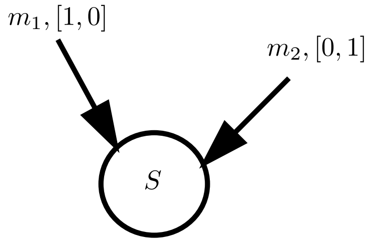

Fig. 6. Sources virtual edges.

7) Let's go over all nodes, from top to bottom:

   $\underline{\text{Edge } (1, 2)}$: it's edge-vector is $r_{(1,2)} = [1, 2]$, and therefore, the first raw in G_1, and the second raw in G_2 correspond to the paths that pass through this edge. The raw vectors are:

$$\mathbf{f_{1,s}} = (1, 0), \mathbf{f_{2,s}} = (0, 1) \tag{38}$$

If we choose

$$\alpha_1 = 1, \alpha_2 = 1 \tag{39}$$

Then the new vector

$$\mathbf{f}_{(\mathbf{1},\mathbf{2})} = \alpha_1 \mathbf{f}_{(\mathbf{1},\mathbf{s})} + \alpha_2 \mathbf{f}_{(\mathbf{2},\mathbf{s})} = (1,1) \tag{40}$$

can replace the raws 1 and 2 in G_1 and G_2 respectively, keeping them invertible. So now, we have

$$G_1 = \begin{pmatrix} 11 \\ 01 \end{pmatrix}, G_2 = \begin{pmatrix} 10 \\ 11 \end{pmatrix} \tag{41}$$

Edge (1,3): It's edge-vector is $r_{(1,3)} = [2,1]$, and therefore, the second raw in G_1, and the first raw in G_2 correspond to the paths that pass throw this edge. We have

$$\mathbf{f}_{\mathbf{1},\mathbf{s}} = (1,0), \mathbf{f}_{\mathbf{2},\mathbf{s}} = (0,1) \tag{42}$$

If we choose

$$\alpha_3 = 1, \alpha_4 = 0 \tag{43}$$

Then the new vector

$$\mathbf{f}_{(\mathbf{1},\mathbf{3})} = \alpha_3 \mathbf{f}_{\mathbf{1},\mathbf{s}} + \alpha_4 \mathbf{f}_{\mathbf{2},\mathbf{s}} = (1,0) \tag{44}$$

can replace the second and the first raws in G_1, G_2 respectively, by keeping G_1 and G_2 invertible. The new global encoding kernel matrices will be:

$$G_1 = \begin{pmatrix} 11 \\ 10 \end{pmatrix}, G_2 = \begin{pmatrix} 10 \\ 11 \end{pmatrix} \tag{45}$$

Edge $(2,4)$: We can of course go as we mentioned before, but in this case, there is only one input to the node (2), and therefore only one coefficient α_5. In such cases, we can always choose that coefficient to be 1, and don't change the global encoding kernel matrices. That's also the case with the edges $(2,6), (3,4), (3,7)$.

Edge $(4,5)$: The edge-vector is: $r_{(2,4)} = [2,2]$ and therefore, the second raw in G_1 and in G_2 correspond to the paths that pass throw this edge. We have

$$\mathbf{f}_{(\mathbf{2},\mathbf{4})} = (1,1), \mathbf{f}_{(\mathbf{3},\mathbf{4})} = (1,0). \tag{46}$$

We should find α_9, α_{10} such that the vector

$$\mathbf{f}_{(\mathbf{4},\mathbf{5})} = \alpha_9 \mathbf{f}_{(\mathbf{2},\mathbf{4})} + \alpha_{10} \mathbf{f}_{(\mathbf{3},\mathbf{4})} \tag{47}$$

will replace the second raws in G_1, G_2, and keep them invertible. We can take, for instance,

$$\alpha_9 = \alpha_{10} = 1 \tag{48}$$

(we're working in the F_2 field for now), so the new vector that will replace the second raws is:

$$\mathbf{f_{(4,5)}} = (0,1) \tag{49}$$

And the transfer matrices will be:

$$G_1 = \begin{pmatrix} 11 \\ 01 \end{pmatrix}, G_2 = \begin{pmatrix} 10 \\ 01 \end{pmatrix} \tag{50}$$

Now we've almost finished, since in edges $(5,6),(5,7)$ we can take $\alpha_{11} = \alpha_{12} = 1$ from the same considerations as in $(2,4)$.

The destination nodes $6,7$ can now decode the original messages sent by the source, using the final global encoding matrices:

$$\begin{pmatrix} M_1^1 \\ M_2^1 \end{pmatrix} = (G_1)^{-1} \begin{pmatrix} m_{2,6} \\ m_{5,6} \end{pmatrix}, \begin{pmatrix} M_1^2 \\ M_2^2 \end{pmatrix} = (G_2)^{-1} \begin{pmatrix} m_{3,7} \\ m_{5,7} \end{pmatrix} \tag{51}$$

where M_i^d is the reconstruction of message M_i by decoder d.