EECS 290S: Network Information Flow **Fall 2008**

Lecture 4 — September 9

Lecturer: Anant Sahai and David Tse *Scribe: Jorge Ortiz*

4.1 Existence of Network Codes

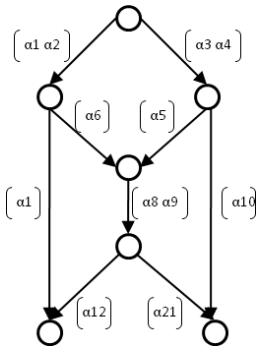

Figure 4.1. Standard butterfly graph. Shows a network code as a function of α.

In this section we wish to prove that there exists a network code, given an acyclic graph. Lets assume we have a graph, as shown in 4.1 above, where each node in the network has a local encoding matrix $A_{ij}(\alpha)$, where $\alpha = (\alpha_1, \alpha_2, \ldots)$ is the vector of coefficients of the linear encoding matrices. We want to show that there exists α such that the global end-to-end matrices $\{A_i(\alpha), i = 1, \ldots, n\}$ are all simultaneously invertible, i.e. every destination t_i can reconstruct the input from the source s. We assume that each symbol x is picked from a finite field F.

We start by trying to find an α such that $A_i(\alpha), \ldots, An(\alpha)$ is full rank. Let

$$C = min_i(mincut(s, t_i)),$$

where s is the source node and the t_i is a destination node, and suppose the source sends C symbols at each time. $A_i(\alpha)$ is a $\sum_k c_{k,t_i}$ by C matrix. For simplicity we will assume here that these matrices are square (ex: extend the argument to the case when they are not.). Since routing solutions are special cases of linear network coding solutions and Ford-Fulkerson guarantees the existence of a routing solution for each destination t_i, it follows that $A_i(\alpha^{(i)})$ is invertible for some $\alpha^{(i)}$, for every i. Since $A_i(\alpha^i)$ is invertible, then $\det(A_i(\alpha^i)) \neq 0$. Thus, $\det(A_i(\alpha))$ viewed as a polynomial in the indeterminates $\{\alpha_n\}_n$ is not identical to zero. Therefore, we want to show that $\exists \alpha^*$ such that $\prod_i \det(A_j(\alpha_j^*)) \neq 0$.

If we let $g(\alpha) \triangleq \prod_i \det(A_i(\alpha))$, then it is a polynomial of α and has at least one non-zero coefficient since it is a product of such polynomials. This by itself does not show that there exists an α for which $g(\alpha) \neq 0$. For example, $g(x) = x^2 + x$ for $x \in F_2 = \{0, 1\}$ is equal to zero for both $x = 0$ and $x = 1$ although it has non-zero coefficients. The theorem below, however, shows that what we want is indeed true if the field size is large enough.

Lemma 4.1. *If $g(\alpha)$ is a polynomial with non-zero coefficients in a field F such that $|F|$ is greater than the degree of g in every α_i, then there exists an $a_1, a_2, \ldots, a_k \epsilon F$ such that*

$$g(a_1, a_2, \ldots, a_k) \neq 0. \tag{4.1}$$

Proof: We prove lemma 4.1 by induction on k. When $k = 0$, g is nonzero as it's a constant in F. If we assume that this is true for k-1. We can express $g(a_1, a_2, \ldots, \alpha_k)$ as a polynomial in α_k with coefficients in F. For example:

$$g((a_1, a_2, \ldots, \alpha_k) = h(a_1, a_2, \ldots, a_{k-1})\alpha_k^m + \ldots, \tag{4.2}$$

where m is the degree of g in α_k and the leading coefficients $h(a_1, a_2, \ldots, a_{k-1})$ is such that $h(a_1, a_2, \ldots, a_{k-1}) \neq 0$. Therefore $g(a_1, a_2, \ldots, a_{k-1}, \alpha_k)$ is a nonzero polynomial with degree $k < |F|$ and g cannot have more than k roots in F. Since $|F| > m$, there must exist $a_k \in F$ such that

$$g((a_1, a_2, \ldots, a_{k-1}, a_k) \neq 0 \tag{4.3}$$

□

Now we ask, if the α_k's are iid in F, how likely is $A_i(\alpha)$ invertible for all i.

Lemma 4.2. *Let E be the event where $A_i(\alpha)$ invertible for all i. Then,*

$$\begin{aligned} Pr(E) &= Pr(g(\alpha_1, \ldots, \alpha_k) \neq 0) \\ &\geq \left(1 - \frac{m}{|F|}\right)^k \end{aligned}$$

where m is the maximum of the degrees of g in each of the variables, $|F|$ is the field size, and k is the number of variables in g.

Proof: (induction on k) Using the same notation as in the previous proof, we can use the induction hypothesis to see that

$$Pr(h(\alpha_i, \ldots, \alpha_{k-1}) \neq 0)) \geq \left(1 - \tfrac{m}{|F|}\right)^{k-1} \quad (\textit{hypothesis}) \tag{4.4}$$

$$Pr(g(\alpha_i, \ldots, \alpha_k) \neq 0)) \geq \left(1 - \tfrac{m}{|F|}\right)^{k-1} \cdot Pr(\alpha_k \text{ is not one of the } m \text{ roots}) \tag{4.5}$$

$$\geq \left(1 - \tfrac{m}{|F|}\right)^{k-1} \left(\tfrac{|F|-m}{|F|}\right) \tag{4.6}$$

$$= \left(1 - \tfrac{m}{|F|}\right)^{k} \tag{4.7}$$

And we can easily see that the probability approaches 1 as $|F|$ approaches infinity. □

4.2 Network Coding for Distributed Storage Systems

This section is from a guest lecture given by Alexandros G. Dimakis.

Storage systems rely on either replication or erasure codes for redundancy to provides recoverability in the case of failure. This section gives a high-level overview of a technique for reducing the amount of data that is transferred in the case where a fragment containing data is lost. By mapping the distributed storage problem into a networking coding problem, networking coding techniques can be used to construct a code that requires less bandwidth consumption than erasure codes. Traditionallly, erasure codes provide the mechanism for a new node to download data from a subset of the remaining nodes in order to recover the data that was lost. This section shows that the repair bandwidth consumption can be significantly reduced using network coding techniques.

Figure 4.2 is an example of the repair problem. In the repair problem, you start with a single node with some data on it. You may also start start with a set of nodes whose union is the complete data. Using regular erasure coding (i.e. a (4,2) MDS erasure code, as shown in the figure), you can generate a set of fragments such that if any one of the fragments is lost, a new fragment can be created using the data stored on the remaining fragments. The figure represents this as a link created from the remaining fragments tranferring β bits from nodes X^1, X^2, and X^3 to the new node X^5.

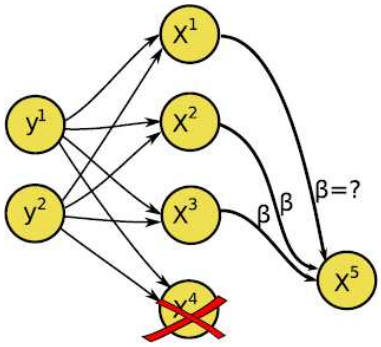

Figure 4.2. In this example, a (4,2) MDS erasure code is used to generate 4 fragments ($x_1, \ldots, x_4$ with the property that any 2 can be used to construct the original y_1, y_2 transfering β bits from 2 of the remaining 3 nodes.

Problem Formulation

Surprisingly, this problem can be reduced into a multicast problem, whose solution shows that when populating a new node to replace the node that went down, not all the information needs to be transferred over to the new node from any particular node. As such, the bandwidth requirement between the links of the remaining nodes and the new node is less than that necessary in a simple erasure code.

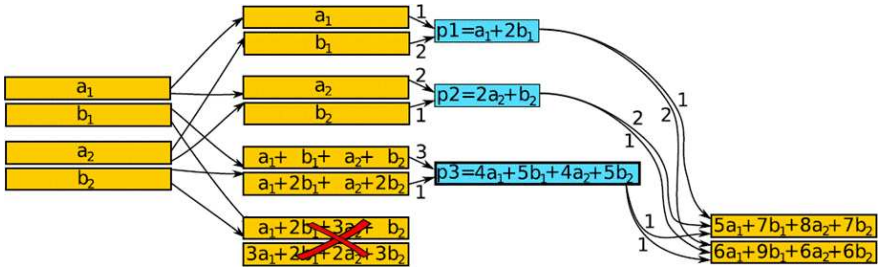

Figure 4.3.

The construction of the graph starts with the initial set of nodes that make up the whole data. Each new link is formed when a node goes down. Links are created between the remaining nodes and the new node. Also, at any time-step, a sink node may be present that connects to the new node and all remaining nodes. This structural composition can be generalized for an infinite number of time steps. Furthmore, any instance of the graph can be used to formulate the encoding scheme to route from a virtual source node that connects to the first n nodes to any sink t_i at time-step i.

Figure 4.3 shows an example of this construction, except erasure coding is used in this example. The initial nodes are the leftmost blocks, where at the first step links are created in the paritioning the data into replicas and codes. In that time-step, the last block goes down, at which point new links are created to new nodes (as shown by $p1$, $p2$, and $p3$).

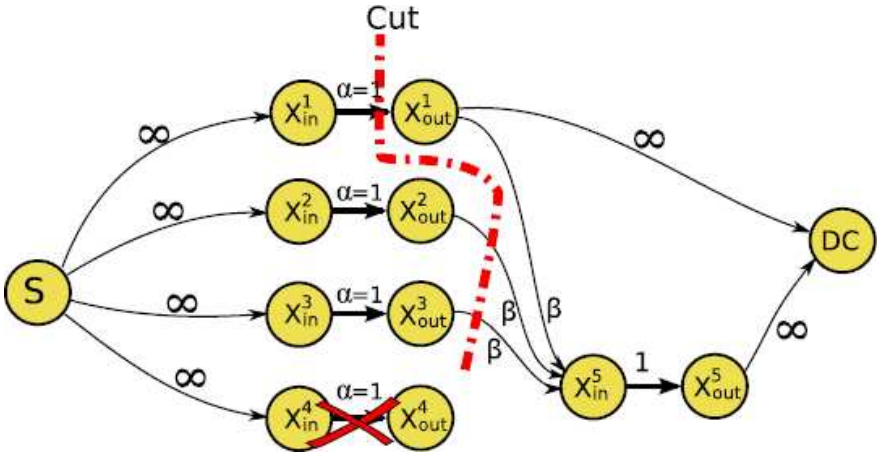

Figure 4.4.

Figure 4.4 shows a similar constructions, except here there is a formal construction of the network graph with sink DC. Using the notation as shown in the figure we can determine the linear code necessary for DC to be able to recontruct all of the data. In the figure, let there be d links between the remaining nodes the new node, where the bandwidth of each

link is β. Furthermore, let α be the number of bits which must be stored at the new node to make the data linearly independet of the remaining nodes.

Theorem 4.3. *For any $\alpha \geq \alpha^*(d, \gamma)$, the points $(n, k, d, \alpha, \gamma)$ are feasible, and linear network codes suffice to achieve them, where n is the number of active storage units, k is the number of storage nodes necessary to recover the original file, d is the number of nodes to download the information from at the new node, α is the number of bits stored at each node, and $\gamma = d\beta$ where β is the number of bits to transfer on a link to a new node.*

The minimum storage (α), connections (d), and bits to transmit (β) is achieved by the following:

$$\alpha_{min} = \frac{m}{k} \tag{4.8}$$

$$d_{min} = n - 1 \tag{4.9}$$

$$\beta_{min} = \frac{m}{d_{min} - k + 1} \tag{4.10}$$

Lemma 4.4. *For any collector t that connects to k remaining nodes:*

$$mincut(s, t) \geq \sum_{i=0}^{min(d,k)-1} min\left[(d - i)\,\beta, \alpha\right] \tag{4.11}$$

More details, including the proofs and related work and analysis can be found in his paper titled "Network Coding for Distributed Storage System" on his web site (http://www.eecs.berkeley.edu/∼adim/).