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

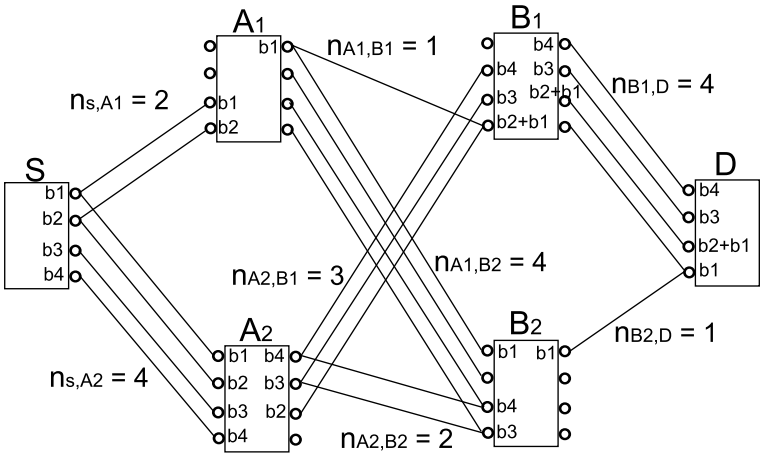
Lecture 11 — October 2

Lecturer: Anant Sahai *Scribe: Kristen Woyach*

This lecture covers the layered deterministic model of relays, specifically a proof that in such networks it is possible to approach the rate of the mincut.

11.1 Example

We begin with an example, shown below, of a network in which it is clearly possible to achieve the mincut:



Here, $n_{i,j}$ is the number of bits able to be transmitted above the noise floor from node i to node j; so, the bigger the $n_{i,j}$, the stronger the channel between i and j. Let q be the maximum number of bits sent or received by any node in the network (in this example, $q = 4$).

Note that this model captures both the broadcast and superposition properties of the wireless medium: broadcast allows each bit from each node to go to multiple destinations. Superposition dictates that when multiple messages are received by the same node, they add (with modulo 2 addition).

In this example, the mincut is 4, which can be achieved by following the transmission pattern denoted with the b_i's in the figure. The destination node receives a set of four linearly independent equations, and so can decode to get the correct bits.

Now we need to show that it is always possible to achieve the mincut.

11.2 Proof of achievability of the mincut

Notice that the network in the example can be divided into layers, where all the nodes within a layer have the same distance to the source and the same distance to the destination. In the example, the layers include: the source as Layer 0, the A_i nodes as Layer 1, B_i nodes as Layer 2, and the destination D as the final Layer. This proof is restricted to networks that can be thought of as a layered series of of relays.

How does time work?

Allow block codes of length T time steps, and divide time into epochs:

Epoch	What Happens
1	Source sends message 1
2	Source sends message 2, Layer 1 relays message 1
3	Source sends message 3, Layer 1 relays message 2, Layer 2 relays message 1
etc.	etc.

The source will transmit at rate R bits/transmission; therefore there are a total of 2^{RT} distinct messages.

How do we encode?

Consider A_1 — it observes $2T$ bits and transmits $4T$ bits during every epoch. It needs some mapping between the input and output bits. Since we are proving achievability here, we are free to assume any structure we want. So we will restrict attention to all linear maps:

Let $\vec{y}_j$ = Received bits at node j during times $1 \dots T$

Let $\vec{x}_j$ = Transmitted bits from node j during times $1 \dots T$

Then the encoding $\vec{x}_j = F_j \vec{y}_j$

$\vec{x}$ and $\vec{y}$ are both of size qT, so F is of size $qT \times qT$. Note that these definitions are for a single node, so F_j is determining the encoding node j performs using its own input. Also, F is not necessarily causal (lower-triangular) because time is causal across *epochs*, not within each epoch of T.

Achievability by random coding:

Generate F_j's randomly using iid $B(\frac{1}{2})$ rv's.(Independence is across rows, columns, and nodes). For nodes that have fewer than q bits of output, just assume that they have q bits, but some of them are simply not connected to anything. (Alternatively, censor the matrix to match the outputs correctly by filling the appropriate places with zeros.)

For notational purposes, let Ancestor(j) be the list of nodes whose outputs $\vec{x}$ show up in $\vec{y}_j$. To understand the superposition effect, define

$$\vec{x}_{Anc(j)} = \begin{bmatrix} \vec{x}_{Anc(j)[1]} \\ \vec{x}_{Anc(j)[2]} \\ \vdots \\ \vec{x}_{Anc(j)[|Anc(j)|]} \end{bmatrix} \tag{11.1}$$

This is all of the $\vec{x}$'s stacked up from the nodes feeding into node j, so $\vec{x}_{Anc(j)}$ is of size $qT \cdot (\#$ of ancestors)

Then:

$$\vec{y}_j = K_j \vec{x}_{Anc(j)} \tag{11.2}$$

where the system K_j must respect causality, and is defined as:

$$K_j = (I_{T\times T} \otimes [G_{Anc(j)[1],j}, \ldots, G_{Anc(j)[|Anc(j)|],j}]). \tag{11.3}$$

where the Gs are the matrices representing between-node transitions, defined in previous lectures: G_{s,A_1}, the transition matrix for the edge between the source and node A_1 in the example from the beginning of this lecture, is:

$$G_{s,A_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{11.4}$$

Also, $\otimes$ is the Kronecker product: every element of the first matrix is replaced by the element of the first matrix multiplied by the entire second matrix. For example:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{bmatrix} \tag{11.5}$$

so, the size of K_j is $qT \times qT \cdot |Anc(j)|$.

Calculating the Probability of Error

An error occurs if two messages, w and w' lead to the same $\vec{y}_D$, the observation $\vec{y}$ at the destination node, so that the destination cannot tell these two messages apart. Since everything is deterministic once the coding matrices are specified, the observation $\vec{y}$ is a function of the message transmitted.

So, the probability of error for message w' is:

$$P(error) = P_e \leq \sum_{w} P(\vec{y}_D(w) = \vec{y}_D(w')) \tag{11.6}$$

where the w ranges over all $2^{RT} - 1$ possible messages that are not w'.

The key simplifying insight is to notice that because everything is linear, we can say that if there is an error, there will also be a code-word colliding with the zero vector. So, we can assume without loss of generality that $\vec{y}_D(w') = \vec{0}$. Furthermore, because of the random nature of the coding matrices, each of the non-zero messages result in statistically identical $\vec{y}$ observations.

Then:

$$P_e \leq 2^{RT} P(\vec{y}_D(w) = \vec{0}) \tag{11.7}$$

where w is some non-zero message.

In Layer 1, the transition matrices F are made up of $B(\frac{1}{2})$ rv's. So, after Layer 1, any non-zero message will have transmissions that look like $B(\frac{1}{2})$ rv's. It doesn't really matter what the non-zero message is. We are therefore looking for the probability that something made up of $B(\frac{1}{2})$ rv's will collide with the zero codeword. The connection to the rank (used in evaluating the value of a cut) should now become intuitively clearer.

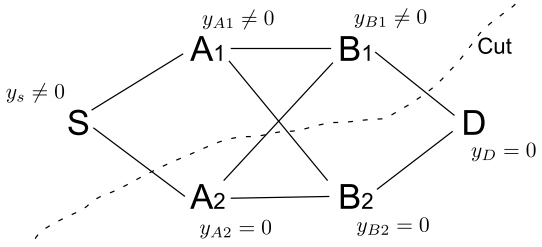
There are many ways that a zero could result at the destination. Let's look at this in terms of cuts. The probability of error is now:

$$P_e \leq 2^{RT} \sum_{cuts} P(\vec{y}_{cut_R} = 0 | \vec{y}_j \neq 0 \forall j \in cut_L) P(\vec{y}_j \neq 0 \forall j \in cut_L)) \tag{11.8}$$

$$\leq 2^{RT} \sum_{cuts} P(\vec{y}_{cut_R} = 0 | \vec{y}_j \neq 0 \forall j \in cut_L). \tag{11.9}$$

In other words, the probability of error is bounded by the sum of the probabilities over cuts that all the observations are non-zero on the left (source) side of the cut but the observations are all zero on the right (destination) side of the cut.

The first kind of cut looks like this (using the original example):

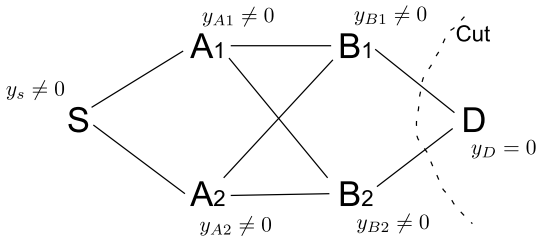


In this cut, none of the relevant outputs are superimposed with anything other than zero, so the probability of this happening is the probability that the F matrices of the nodes to the left of the cut are such that the relevant transmitted bits are all zero:

$$\begin{aligned} \text{Prob at } D &= 2^{-n_{B_1,D}T} \\ \text{Prob at } B_2 &= 2^{-n_{A_1,B_2}T} \\ \text{Prob at } A_2 &= 2^{-n_{S,A_2}T} \\ \text{Total Probability} &= \prod_{D,B_2,A_2} P = 2^{-|cut|\cdot T} \end{aligned}$$

where in this case the value of a cut $|cut| = \sum n_{i,j}$ for all i and j that cross the cut since no two of these edges actually meet in superposition at a single node. The total probability is the product because all of the F_i's are independent.

The second type of cut is shown here:



In this cut, the inputs superimpose, so we need the probability that the modulo 2 sum of the relevant outputs leads to a zero message at the input of the destinations. The probability of this is:

$$\begin{aligned} \text{For each level with only one input:} &\quad P = 2^{-T} \\ \text{For each level with two superposed inputs:} &\quad P = 2^{-T} \\ \vdots &\quad \vdots \\ \text{For any other level:} &\quad P = 2^{-T} \\ \text{Total:} &\quad P = 2^{-|cut|\cdot T} \end{aligned}$$

These are all the same because $\sum_n B(\frac{1}{2}) =_d B(\frac{1}{2})$ by the fundamental theorem of cryptography that states that the mod sum of uniform random variables with any other random variable is again just a uniform random variable. If the original uniform random variables are independent of each other, then so are these.

Putting this together for the general case where the same nodes output touches two or more different nodes on the right of the cut reveals why the definition of the value of a cut involves the rank of the transfer matrix across the cut. This is because any rank-$|cut|$ matrix G with α rows (representing all the levels of the node inputs on the right of the cut) and β columns (representing all the levels of the node outputs on the left of the cut) can be represented as $G = G'H$ where H is a full rank matrix with only $|cut|$ rows. The probability that $G\vec{x} = 0$ when $\vec{x}$ is filled with fair coin tosses is the same as the probability that $H\vec{x} = 0$. By the above arguments, this probability is just $(2^{-|cut|})$ for each time step in the epoch and since the random coding matrix entries

are iid across time, this again gives $2^{-|cut|\cdot T}$ for the probability of getting all zeros on the right of the cut.

Thus overall:

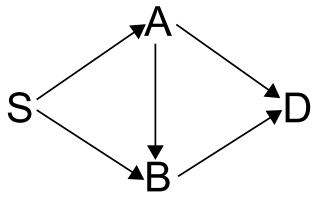
$$\begin{aligned} P_e &\leq 2^{RT} \cdot \sum_{cut} 2^{-|cut|\cdot T} && (11.10)\\ &\leq 2^{RT} \cdot \#cuts \cdot 2^{-\min|cut|\cdot T} && (11.11)\\ &= \#cuts 2^{-(\min|cut|-R)T} && (11.12)\\ & && (11.13) \end{aligned}$$

Let T get big, and as long as $\min|cut| > R$, $P_e \to 0$. Here however, there is no need for R to be an integer since only RT must be an integer.

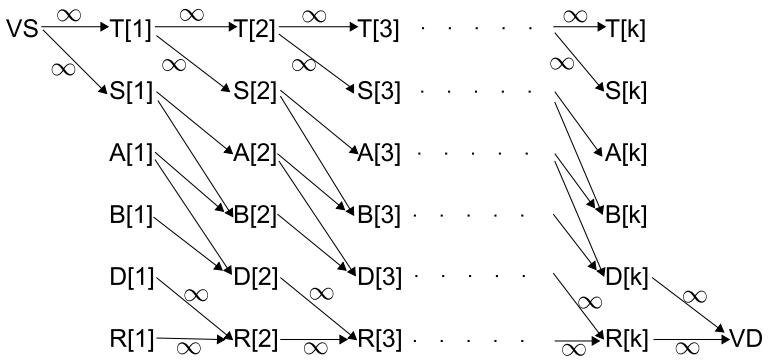
So, good codes must exist. Note that because all of this is deterministic, once $P_e < 1$, a good code must exist, so codes exist even for finite T.

11.3 How this extends to DAG's that are *not* layered (Preview of next lecture)

Consider a network like this:



in which the nodes cannot be grouped into layers because B is both one and two hops from the source along different paths. We can use the same kind of trick to analyze this case as we did for cyclic graphs — unroll time into a trellis:



With cyclic graphs, we used virtual sources and virtual destinations. But, if we did that here, the new trellis would no longer be layered. So, the trick here is to add memory nodes: T nodes are the source's proxy, and R nodes are rememberers for a virtual destination. The proof will go something like this: if the graph is unrolled k-times, make the k big enough to reflect the mincut in the original. Then, make the epoch big enough for codes to exist. This argument will be shown in detail next time. The challenge is to show that the mincut for the unrolled network cannot be much smaller than k times the mincut in the original network.