EECS 290S: Network Information Flow	Fall 2008

Lecture 10 — October 7

Lecturer: Anant Sahai *Scribe: Mohammad A. Maddah-Ali*

10.1 Outline

The linear deterministic network models are simplified abstractions that try to capture the essential broadcast and superposition aspects of wireless networks, but suppress the role of additive noise by pretending that everything above the noise-level is heard correctly while everything below the noise-level is completely scrambled.

This lecture's objective: Extending the min-cut theorem to general linear deterministic network models that are not necessarily layered or acyclic.

Technique: Unrolling the graph in time, adding memory arcs to capture the fact that nodes remember what they heard earlier, and forming a layered trellis-shape graph.

10.2 RECAP

RECAP: In the previous lecture,

- We considered a simplified version of linear deterministic networks that are also layered (consequently acyclic as well). Layered networks have the property that every node's distance from the source is uniquely defined since every path from the source to this node has exactly the same length. Thus nodes can be arranged in "layers" corresponding to how far away from the source that they are.

- Corresponding to each cut which includes the source on one side and the destination on the other side, the *Transform matrix* was defined as the channel matrix between the outputs of the source-side nodes and the inputs to the destination-side nodes. (See previous lecture notes for definition)

- The value of each cut was defined as the rank of the *Transform matrix* of that cut.

- We proved that the rate of min-cut per channel use is achievable by (i) using long blocks (in other words, long packets of information) (ii) linear random mappings at each relay node (iii) a "pipelining" strategy that kept all nodes at each layer busy by having the i-th layer working on message block number $m - i$ while the source was transmitting message block number m.

In this session, the objective is to extend this result to the non-layered and/or cyclic linear deterministic networks.

10.3 The challenge

Consider a non-layered deterministic graph (see Fig. 10.1 as an example). Assume that the source feeds the network with packets of length T.

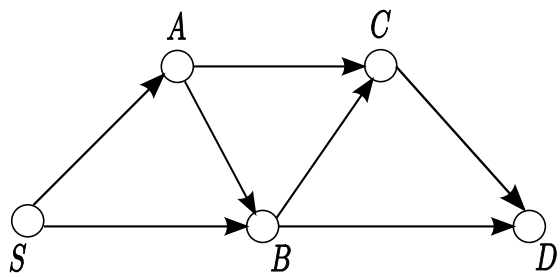

Figure 10.1. Non-Layered Graph

When the network is not layered, some nodes receive a superposition of the contributions of different bit-positions within the packet. As an example consider node B of the network shown in Fig 10.1. It is easy to see that node B receives a superposition of the time i contribution and time $i+1$ contribution coming from the direct link S-B and indirect path S-A-B, respectively. Therefore, it is not possible to directly apply the coding scheme employed for layered graph. In what follows, we show that by unfolding the graph in time, we can form what looks like a layered network, and adapt the former scheme.

10.4 Min-Cut for the Unrolled Graph

The technique is very similar to the unfolding scheme (copying each node for each time stage and copying the edges moving forward one-step in time) that was employed for the wireline cyclic graph to get back to the DAG case. In the wireline network-coding case, we were able to just have a single source feeding all layers as well as a single destination being fed by all the layers. However, that does is not a layered network by our definition. So we need to slightly modify the unrolling scheme to make the resulting graph layered.

In the wireline case, we never had self-edges since our ultimate goal was to have a network-code that was both memoryless and time-invariant. However, this is no longer an objective since the layered-network proof already has T (and hence the memory of each relay node) getting large. So here, we add K copies of the source node to the network and connect them with infinite-capacity links shown with blue edges in Fig. 10.2. Similarly, we add K copies of the destination and connect them with infinite-capacity links going forward in time. Therefore, the resulting trellis-shape graph is layered. These infinite-capacity links represent the internal memory of each node.

The original proof of Avestimehr and Tse only had these infinite capacity links. However, here we assume that the intermediate relay nodes also have memory. Again these memories are represented with infinite-capacity blue links connecting each node in current stage to the same node in the next stage.

Remark: The transfer matrix of the blue edge (self-arc) is chosen such that it does not have any collision/interaction with the other links at wireless bit levels. For example, it is chosen as

$$\mathbf{G}_{self-arc} = \begin{bmatrix} \mathbf{0}_{q\times L} & \mathbf{0}_{q\times q} \\ \mathbf{I}_{L\times L} & \mathbf{0}_{L\times q} \end{bmatrix} \tag{10.1}$$

where L is a very large number representing bits that would otherwise be below the noise level but are not since there is no noise on the node's own memory, q is the maximum signal level in the original graph, $\mathbf{I}_{L\times L}$ denotes the identity matrix, and $\mathbf{0}_{L\times q}$ is an all-zero matrix. There is no superposition effect on these sub-noise bits since they are simply not received on any of the wireless links.

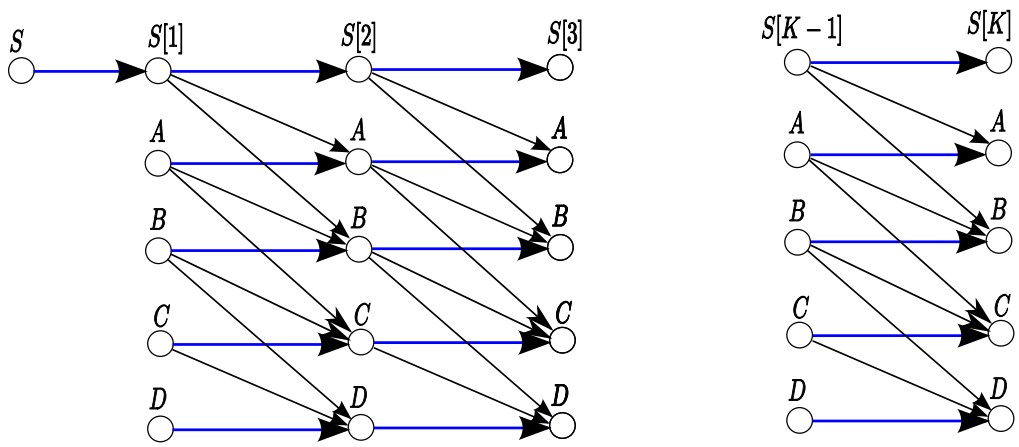

Figure 10.2. Unfolded Graph in Tim

In what follows, we show that the min-cut of the unrolled layered graph (divided by K) is within an $(1-\epsilon)$-factor gap of the min-cut of the original graph, for any arbitrary ϵ, by choosing large enough K. Therefore, the rate of min-cut per channel use is achievable for the original graph.

We can consider all possible cuts with the source and destination on different sides in the unrolled graph. Because the edges only go from one stage to the next, the transfer matrix of cuts in the unrolled graph is a block-diagonal matrix. The rank of a block-diagonal matrix is just the sum of the ranks of each block. The cuts themselves can be categorized into three types:

Type One: Straight Cuts (Fig. 10.3): This type of cut is just straight. This means that at every time-stage, the same nodes are on the source and destination side of the cuts. It is easy to see that there is a one-to-one map between these cuts and the cuts in the original graph. For example, the cut shown in Fig. 10.3 corresponds to the cut which separates $\{S, A\}$ and $\{B, C, D\}$ in the original graph.

The transfer matrix of such a cut in the unrolled graph is a block-diagonal matrix where the each block is the transfer matrix of the corresponding cut in the original graph. Therefore,

the value of the cut in the unrolled graph is K times the value of the corresponding cut in the original graph. Therefore, the min-cut of the unrolled graph is less than or equal to K times of the min-cut of the original graph, i.e.

$$C_{min-unrolled} \leq KC_{min-original}, \tag{10.2}$$

where $C_{min-unrolled}$ the min-cut of the unrolled graph and $C_{min-original}$ the min-cut of the original graph.

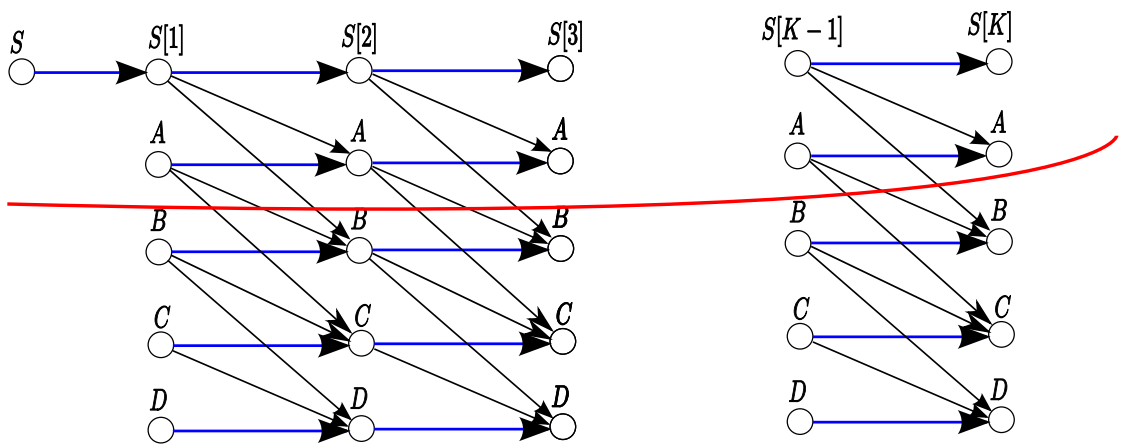

Figure 10.3. Cut Type 1

Unfortunately, there are also other kinds of cuts that we must consider.

Type Two: "Dipping Cuts" (Fig. 10.4): This type of cuts has just two kinds of segments: straight segments and dipping segments. The straight segments of the cut corresponds to those stages at which the nodes on both sides of the cut form a partition of all the nodes in the original graph. So for these segments, the cut matrix has a block that corresponds to a cut in the original graph. Thus, the value of this segment is just the value of this cut in the original graph.

The dipping segments of a cut are those in which a node moves from being on the destination-side of the cut at one stage to being on the source-side of the cut at the next stage. Thus, the edge of the cut "dips" and includes another node on the source (top) side. Graphically, it appears at first glance that when the dip part moves down, it cuts blue edges. However due to the directionality of such edges, their contributions to the value of the cut is just *zero* since only edges coming out of the source-side of the cut matter. These blue edges are leaving the destination-side of the cut and so do not matter.

However, these dipping segments still do not correspond to a cut in the original graph since they are missing nodes that should be on the destination side but aren't. Thus, the value of these segments can be less than the original mincut. So we lower-bound the value of these segments by zero.

What saves us is that there cannot be too many dips in a cut.

We can have at most $|V|$ dips in the cut ($|V|$ denotes number of the nodes in the original graph). On the other hand, the straight segments move through at least $(K - |V|)$ stages and therefore, the contribution of the straight segments seems to be at least $(K - |V|)TC_{min-original}$. However, there are transient effects at the beginning of time and the end of time where certain nodes might as well not exist in the unrolled network since they are unreachable from either the original source or the final destination. To take these transients into account, we have to subtract $2|V|TC_{min-original}$ from $(K - |V|)TC_{min-original}$ (see the proof for cyclic wireline graphs). Therefore, the value of this type of cut is at least $(K - 3|V|)TC_{min-original}$.

Dividing by K to normalize, this $\frac{3V}{K}$ can be made smaller than any ϵ by choosing K large enough.

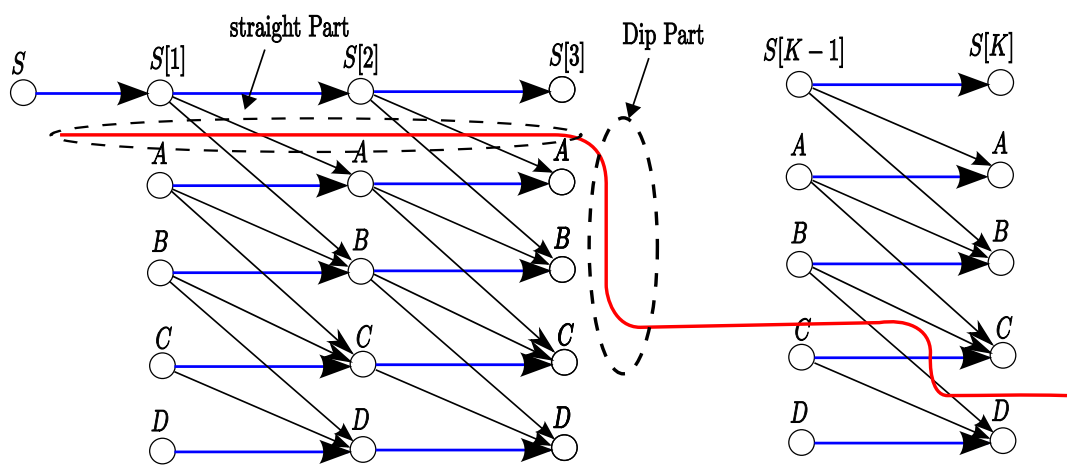

Figure 10.4. Cut Type 2

Type Three: Other Cuts (Fig. 10.5): These are cuts that include an "upward segment" in which a node is on the source-side of the cut at stage k but on the destination-side at stage $k + 1$. So, the cut edge seems to move upward in the figures as a node leaves the source-side. This type of cut involves cutting the blue edges in the upward part of the cut. This makes the value of the cut infinite. To understand this, we must remember the interpretation of the cut. Everything on the source side of the cut is believed to consider that the message is nonzero while those on the destination side of the cut have to get observations that are all zero. This is impossible for such cuts since it would have to have a node forget that it knew that the message was nonzero. Therefore, these other cuts do not provide any upper-bound on the min-cut value and can be ignored.

Considering the value of the cuts form the three types of cut we conclude that

$$(K - 3|V|)TC_{min-original} \leq TC_{min-unrolled} \leq KTC_{min-original}. \tag{10.3}$$

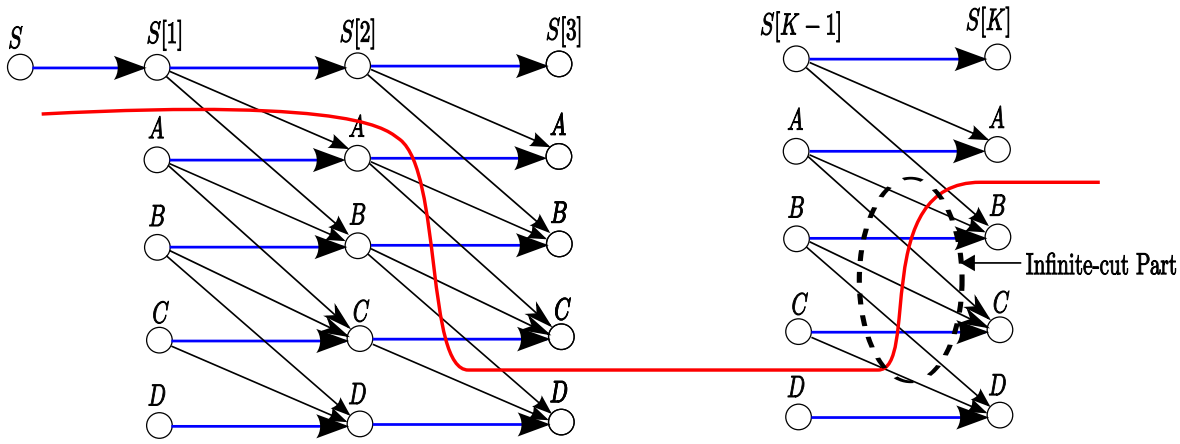

Figure 10.5. Cut Type 3

Therefore, by adapting the linear random mapping scheme for the layered code, we can achieve the rate of R per actual channel use, where

$$(1 - \frac{3|V|}{K})C_{min-original} \leq R \leq C_{min-original}. \tag{10.4}$$

Next time, we will show how this adaptation occurs. The main concern is how to interpret the unrolled time along with the T from the layered proof.