CS 466/666 Algorithm Design and Analysis . Spring 2022

Lecture 8 : Network coding

Network coding is a novel method to transmit data in a computer network.
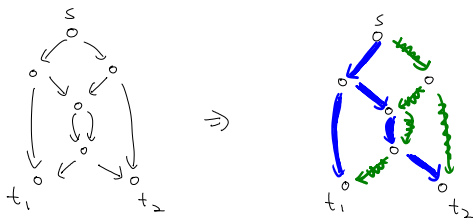Randomization makes it work in a fully distributed manner.

Network information flow

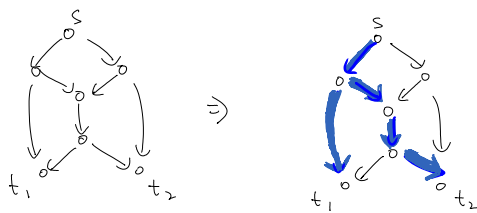For simplicity, we assume the graph is a directed acyclic graph.

Multicasting : Given a directed acyclic graph $G=(V,E)$, a source vertex $s \in V$, and a set of receiver
vertices $\{t_1, \dots, t_\ell\} \subseteq V$, the multicasting problem is for the source to send data to all the
receivers simultaneously, and the objective is to maximize the rate (speed) of transmission.

Tree packing : In traditional computer network setting, each intermediate node can be used to store and
forward data. Assume that each edge can transmit one unit of data per step.

To increase the transmission rate, we need to find edge-disjoint trees connecting the source to all receivers.

Two edge-disjoint trees in this example.

Only one edge-disjoint tree can be found,
even though there are two edge-disjoint
paths from s to each t_i.

It is a computationally hard problem to compute the maximum number of such edge-disjoint trees.
Also, there are graphs with high "connectivity" from the source to each receiver, but there are very
few edge-disjoint trees connecting the source to all receivers.
That is, while it is easy for the source to send data quickly to each individual receiver, it becomes
difficult for the source to send data quickly to all receivers simultaneously.
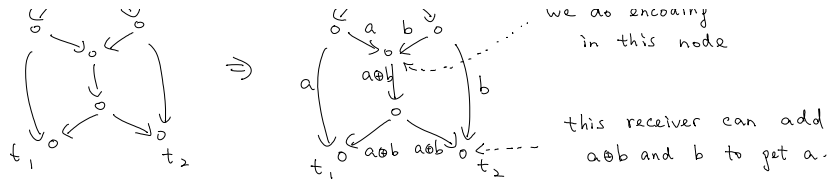
Network coding: The idea of network coding is that information flow is different from standard commodity flow.
We could do encoding in intermediate nodes so that the receivers can decode the messages
and the throughput is increased!

can send two units of
information in this example

we do encoding
in this node

can send two units of
information in this example
by doing encoding and decoding

More generally, network coding allows:

① arithmetic operations over a larger field (instead of just bit-wise operations).

② the data sent on an edge is any function of its predecessors (not just XOR).

Surprisingly, the rate of transmission could be significantly improved if network coding is used. In fact, it can always achieve the optimal rate for multicasting.

<u>Theorem</u> (max-information-flow min-cut theorem)

If the source has k edge-disjoint paths to each receiver, then the source can send k units of data to all receivers simultaneously with the use of network coding.

<u>Remarks</u> : ① This is clearly optimal. This is an amazing result that opens up a new research area.
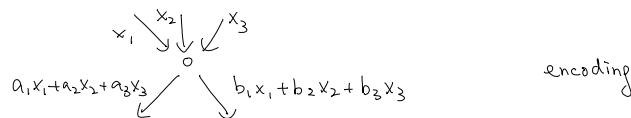
② There are examples for which the ratio of the rate achieved by network coding to the rate achieved without using network coding is unbounded.

Linear Network Coding

An important algorithmic question is to find an optimal encoding and decoding scheme efficiently.

The next major result is that <u>linear</u> network coding is enough to achieve the optimal rate for multicasting.

In linear network coding, the data sent on an edge is always a linear combination of its predecessors.

encoding

A receiver can decode if what it receives are linearly independent.

$$m_1 = a_1 x_1 + a_2 x_2 + a_3 x_3$$
$$m_2 = b_1 x_1 + b_2 x_2 + b_3 x_3 \quad =: \quad \vec{m} = A\vec{x}$$
$$m_3 = c_1 x_1 + c_2 x_2 + c_3 x_3$$

$\vec{x}$ can be uniquely determined if A is of full rank.

<u>Deterministic algorithm</u> : Subsequently, deterministic polynomial time algorithms are obtained to find an optimal coding scheme for multicasting, i.e. the coefficients of the linear combinations.

However, these algorithms will be very difficult to be used in practice, as it requires a "central authority" to look at the whole graph and carefully choose the coefficients and tell everyone how to encode.

<u>Randomization</u>: We study a randomized polynomial time algorithm to find an optimal coding scheme for multicasting.

The proof also provides an alternate proof for the max-information-flow min-cut theorem and the linear coding thm.

Most importantly, the algorithm is very simple and totally decentralized, a key feature to be practical.

<u>Setting</u> : By a simple reduction, we can assume that the source has exactly k edges, each reciever has

exactly k incoming edges, where k is the edge-connectivity from the source to the receivers.

The source would like to send k messages $X_1,...,X_k$ to all receivers.

<u>Algorithm</u>: Pick a large enough (prime) field $\mathbb{F}$.

The source sends $X_1,...,X_k$ on its outgoing edges, X_i on its i-th outgoing edge.

Follow a topological ordering $v_1 = S, v_2,..., v_n$ of the directed acyclic graph.

For vertex v_i, call the incoming messages $m_1, m_2,..., m_\ell$.

For each outgoing edge e of v_i, send a random linear combination of $m_1,...,m_\ell$,

i.e. send $b_1^{(e)} m_1 +...+ b_\ell^{(e)} m_\ell$ on e, where each $b_i^{(e)}$ is an independent random element in $\mathbb{F}$.

<u>Global encoding vector and local encoding coefficients</u> : These are important definitions for the analysis.

Each edge is sending a linear combination of the source messages, $c_1 X_1 + c_2 X_2 +...+ c_k X_k$, which follows

by induction on the topological ordering.

We call $(c_1,...,c_k)$ the global encoding vector on this edge, the actual linear combination that its sending.

For each edge e, the (random) coefficients $b_1^{(e)}, b_2^{(e)},...$ are called the local encoding coefficients.

The local encoding coefficients are the decisions that we need to make (which we decide to make randomly).

Once all the local encoding coefficients are fixed, all the global encoding vectors are determined.

<u>Decodability</u> : For a receiver t with k incoming edges, let the global encoding vector of its

i-th incoming edge be $(c_{i1}, c_{i2},..., c_{ik})$ and the i-th incoming message is m_i.

Then, whether t can decode the original messages depends on whether the k incoming global encoding

vectors are linearly independent, or equivalently whether the $k \times k$ matrix $C = \begin{pmatrix} c_{11}, c_{12},..., c_{1k} \\ c_{21}, c_{22},..., c_{2k} \\ \vdots \\ c_{k1}, c_{k2},..., c_{kk} \end{pmatrix}$ is

of full rank, or equivalently whether $\det(C) \neq 0$.

$$m_i = c_{i1} X_1 +...+ c_{ik} X_k$$

$m_1 = c_{11} X_1 +...+ c_{1k} X_k$ \ $\cdots$ $\downarrow$ $\cdots$ / $m_k = c_{k1} X_1 +...+ c_{kk} X_k$ $\quad \Rightarrow \quad$ $\begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{pmatrix} = \begin{pmatrix} c_{11}, c_{12},..., c_{1k} \\ c_{21}, c_{22},..., c_{2k} \\ \vdots \\ c_{k1}, c_{k2},..., c_{kk} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{pmatrix}$ $\quad \Rightarrow \quad$ $\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{pmatrix} = C^{-1} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{pmatrix}$.

$\circ t$

<u>Analysis</u> : We need to analyze the probability that $\det(C) \neq 0$ if we choose all local coefficients randomly.

Think of each local encoding coefficient as a variable (the decision variable).

Each entry of a global encoding vector is a multivariate polynomial involving the local encoding coefficients.
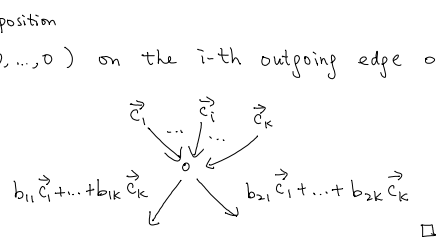
Claim : For vertex v_i in the topological ordering, for each outgoing edge e of v_i, each entry c_j of its global encoding vector is a multivariate polynomial of the local encoding coefficients of total degree at most $i-1$.

Proof : This follows from a simple induction.

The base case is clearly true, with vector $(0,0,...,1,0,...,0)$ on the i-th outgoing edge of s.

In each step, we just add degree one to the polynomial, as shown in the picture.

By the claim, each entry in a receiver matrix C is just a multivariate polynomial of the local encoding coefficients of total degree at most n.

This implies that $\det(C)$ is a multivariate polynomial of the local coefficients of total degree at most kn.

If $\det(C) \neq 0$, then by the Schwartz-Zippel lemma, the probability that $\det(C)=0$ when we put in random values is at most $\frac{kn}{|\mathbb{F}|}$.

By choosing $|\mathbb{F}| \geqslant kn^3$, this probability is at most $\frac{1}{n^2}$ for one receiver.

By the union bound, every receiver is of full rank with probability at least $1-\frac{1}{n}$.

It remains to show that $\det(C) \neq 0$, using our assumption that there are k edge-disjoint paths.
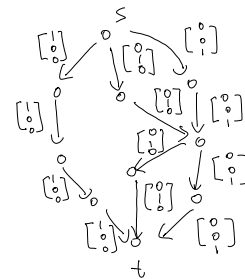
Claim If the source s has k edge-disjoint paths to a receiver t, then $\det(C) \neq 0$ where C is the receiver matrix of the global encoding vectors.

Proof Just use the k edge-disjoint paths from s to t to "forward" (traditional solution without coding) the k messages from s to t.

The receiver matrix will be an identity matrix.

So, there is one choice of local encoding coefficients s.t. $\det(C) \neq 0$, which implies that $\det(C)$ is a non-zero polynomial. $\square$

To summarize, if there are k edge-disjoint paths between the source and each receiver, then the determinant of each receiver matrix is a _non-zero_ polynomial of the local encoding coefficients.

The determinant polynomials are of _low degree_, and so Schwartz-Zippel implies that random substitutions will work to ensure that all receiver matrices are of full rank, so that we can decode.

Efficient Encoding

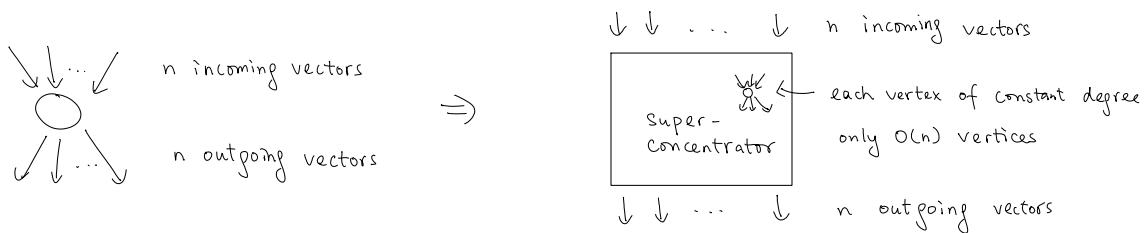Consider the problem of implementing the encoding operation at a node v.

Suppose each vector is of d-dimensional. There are n incoming vector and n outgoing vectors.

Then each outgoing vector can be computed in $O(dn)$ time, and all outgoing vectors can be computed in $O(dn^2)$ time.

Can we do it faster? Observe that the encoding can be done quickly if indegree is small.

So, it would be good if there is a transformation to reduce the degree while preserving connectivity.

A <u>superconcentrator</u> comes to rescue (see next lecture for its construction)

A superconcentrator is a directed acyclic graph with n input nodes and n output nodes, with $O(n)$ internal nodes each with constant indegree and outdegree.

An important property is that any k inputs can connect to any k outputs by vertex disjoint paths, for any k.

Thus, the connectivity from the source to a receiver would not decrease.

Therefore, we can first apply the transformation before doing network coding.

After the transformation, each vertex in the resulting graph has constant indegree.

So, the encoding in each vertex (all the outgoing edges) can be done in $O(d)$ time.

All the vertices in a superconcentrator can be done in $O(d \cdot n)$ time, as a vertex with indegree n is replaced by a superconcentrator with n inputs, and it has only $O(n)$ vertices.

Hence, the vectors of the outgoing edges of a vertex in the original graph can be computed in $O(dn)$ time, faster than the straightforward $O(dn^2)$ time algorithm.

Note that this is optimal, since only writing down n output vectors requires $\Omega(dn)$ time.

<u>**References and discussions**</u>   There are several surveys about network coding.

One could take a look at the book "Information theory and network coding" by Yeung.

There are more and more results using algebraic techniques to design fast algorithms.

Polynomial identity testing is also very useful in designing efficient "interactive proofs" (see [MR]).

More generally, randomness is very useful in "checking" things efficiently. A famous example is the celebrated PCP (probabilistically checkable proofs) theorem.