Network Coding Problem - An Introduction

Dott. Marco Calderini

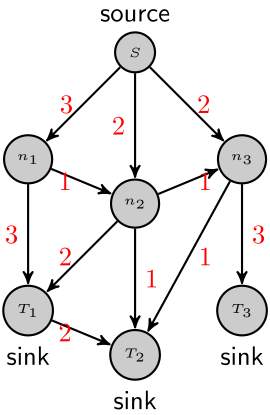
University of Trento, Italy

22 May 2013

Flow Network

Definition

A directed graph, with sources and sinks, where each edge e has a capacity c_e, where each edge receives a non-negative flow f_e (limited by c_e), and where the net flow into any non-source non-sink vertex is zero.



The Min-Cut Max-Flow Theorem

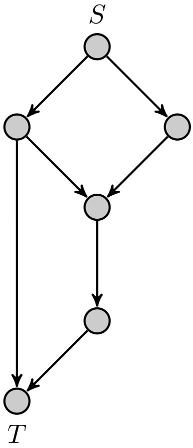
Let $G = (V, E)$ be a graph (network), and let S be the source a T a sink.

Definition

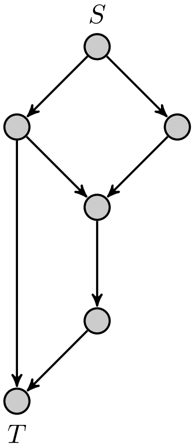
A *cut* between S and T is a set of graph edges whose removal disconnects S from T. A min-cut is a cut with the smallest (minimal) value. The *value* of the cut is the sum of the capacities of the edges in the cut.

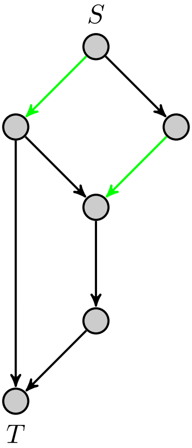
We will consider unit capacity edge.

Example

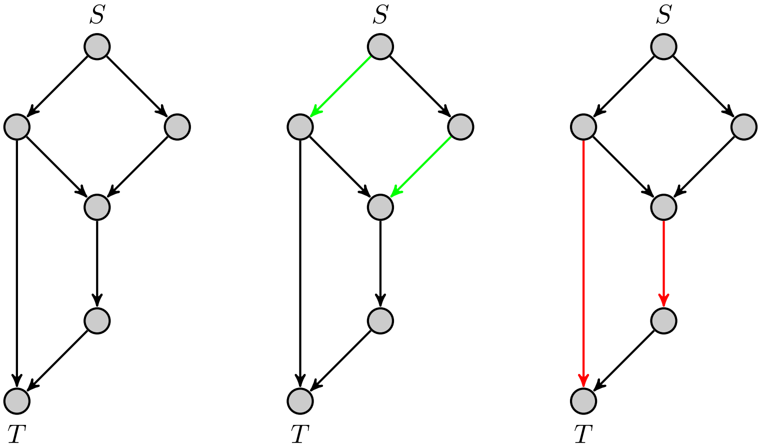


Example





Example

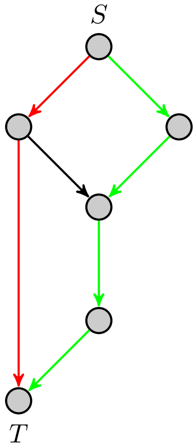


Theorem (Min-Cut Max-Flow Theorem)

Consider a graph $G = (V, E)$ with unit capacity edges, a source vertex S, and a receiver vertex T. If the min-cut between S and T equals h, then the information can be send from S to T at a maximum rate of h. Equivalently, there exist exactly h edge-disjoint paths between S and T.

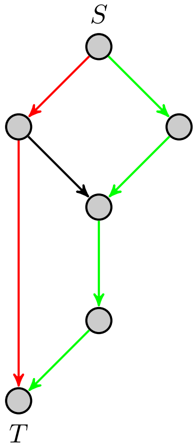
Theorem (Min-Cut Max-Flow Theorem)

Consider a graph $G = (V, E)$ with unit capacity edges, a source vertex S, and a receiver vertex T. If the min-cut between S and T equals h, then the information can be send from S to T at a maximum rate of h. Equivalently, there exist exactly h edge-disjoint paths between S and T.



Theorem (Min-Cut Max-Flow Theorem)

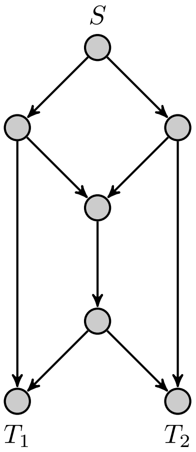
Consider a graph $G = (V, E)$ with unit capacity edges, a source vertex S, and a receiver vertex T. If the min-cut between S and T equals h, then the information can be send from S to T at a maximum rate of h. Equivalently, there exist exactly h edge-disjoint paths between S and T.



Corollary: max-flow from a single source S to a single sink T over network of unit-capacity edges is achievable via routing

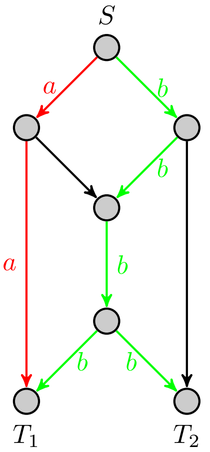
Butterfly Network

- Network: one source, two sinks, unit-capacity edges



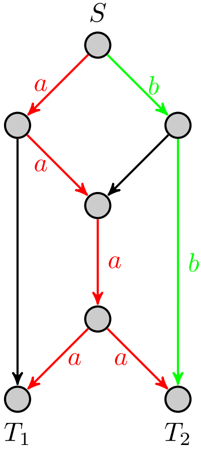
Butterfly Network

- Network: one source, two sinks, unit-capacity edges
- can route two packets to one sink, one to the other



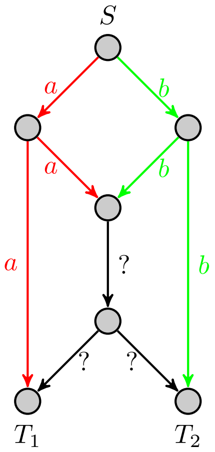
Butterfly Network

- Network: one source, two sinks, unit-capacity edges
- can route two packets to one sink, one to the other
- and vice-versa. Time-sharing between these two strategies can achieve a multicast rate of 1.5 packets per use of the network.



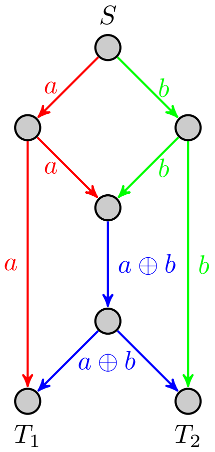
Butterfly Network

- Network: one source, two sinks, unit-capacity edges
- can route two packets to one sink, one to the other
- and vice-versa. Time-sharing between these two strategies can achieve a multicast rate of 1.5 packets per use of the network.
- We can do better?



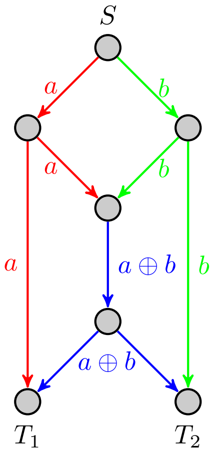
Butterfly Network

- Network: one source, two sinks, unit-capacity edges
- can route two packets to one sink, one to the other
- and vice-versa. Time-sharing between these two strategies can achieve a multicast rate of 1.5 packets per use of the network.
- We can do better?
- Yes! Perform coding at the bottle-neck



Butterfly Network

- Network: one source, two sinks, unit-capacity edges
- can route two packets to one sink, one to the other
- and vice-versa. Time-sharing between these two strategies can achieve a multicast rate of 1.5 packets per use of the network.
- We can do better?
- Yes! Perform coding at the bottle-neck



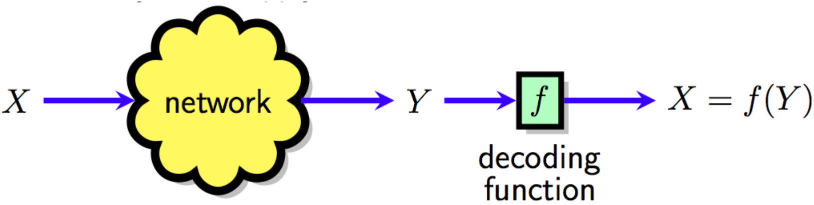
a and b are packets of bits; $a \oplus b$ is the modulo-two sum (XOR) of a and b. Since $a \oplus (a \oplus b) = b$, and $b \oplus (a \oplus b) = a$, both sinks can recover both messages! Network coding achieves a multicast rate of 2 packets per use of the network (the best possible).

Some Lessons Learned from The Butterfly Example

The information superhighway is not like a real superhighway; a bit is not a car!

Nodes in a network are allowed to form outgoing streams from incoming streams in any way (not only time-multiplexing).

The aim of network coding is to provide a receiver with sufficient "evidence" Y about the message X; we want $H(X|Y) = 0$. It is not necessary to to supply the receiver with X itself.



Linear Network Coding

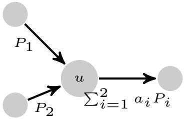
A combinational packet network $\mathcal{N} = (G, S, T, A)$ comprises:

- a finite directed acyclic graph $G = (V, E)$ where V is the set of vertices and E is the set of directed edges;
- a distinguished set $S \subset V$ of sources;
- a distinguished set $T \subset V$ of sinks;
- and a finite packet alphabet A with $|A| \geq 2$.

Vertices: communication nodes
Edges: error-free communication channels of unit capacity (one symbol from A).
Packets transmitted on non-source edges from a node v are functions of packets received at v.
Does not model errors, delay, cycles, etc., but suffices to capture main ideas.

$$X_i = [x_1^i, \ldots, x_m^i] \in \mathbb{F}_q^m$$

- Packets are length-m vectors over a finite field $\mathbb{F}_q$



- Nodes create outgoing packets as $\mathbb{F}_q$-linear combinations of incoming packets

- Original packets can be recovered by solving a linear system of equations

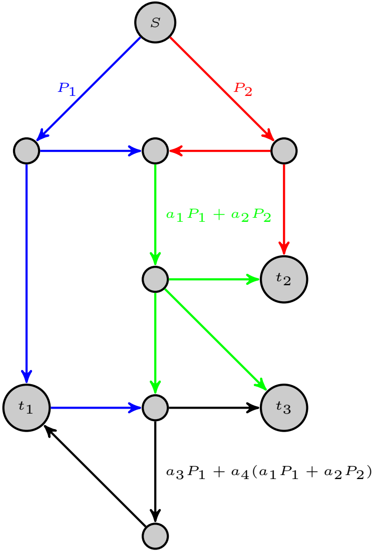
$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \underbrace{\begin{bmatrix} A_{11} & \ldots & A_{1n} \\ \vdots & & \vdots \\ A_{n1} & \ldots & A_{nn} \end{bmatrix}}_{\text{transfer matrix}} \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$$

Theorem (Linear Network Multicasting Theorem)

Let $\mathcal{N} = (G, s, T, \mathbb{F}_q)$. A multicast rate of $R(s,T) = \min_{t \in T} mincut(s,t)$ is achievable, for sufficiently large q, with linear network coding.

- Algebraic proof: Koetter and Médard, 2003. (Can show that $q > |T|$ suffices.)
- Linear Information Flow Algorithm: Jaggi, Sanders, et al., 2005. (Requires only $q \geq |T|$.)

Proof Sketch: Linear Network Multicasting Theorem



$$A_1 = \begin{bmatrix} 1 & 0 \\ a_3 + a_1 a_4 & a_2 a_4 \end{bmatrix} A_2 = \begin{bmatrix} 0 & 1 \\ a_1 & a_2 \end{bmatrix} A_3 = \begin{bmatrix} a_1 & a_2 \\ a_3 + a_1 a_4 & a_2 a_4 \end{bmatrix}$$

Condition for decoding: $det(A_i) \neq 0$ for $1 \leq i \leq 3$.
This is an algebraic condition: we require a nonzero value for the polynomial

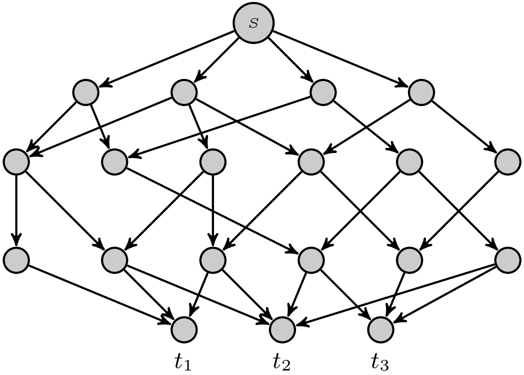
$$p(a_1, a_2, a_3, a_4) = det(A_1) \cdot det(A_2) \cdot det(A_3).$$

Lemma (Sparse Zeros Lemma)

Over a sufficiently large finite field, a nonzero polynomial takes on a nonzero value.

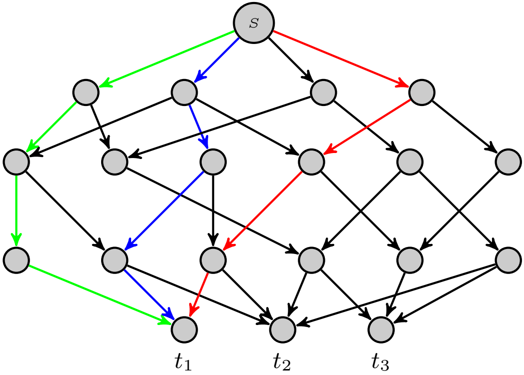
(Here, one can show that $q > |T|$ suffices.) Thus linear network coding achieves the multicast capacity.

Linear Information Flow Algorithm



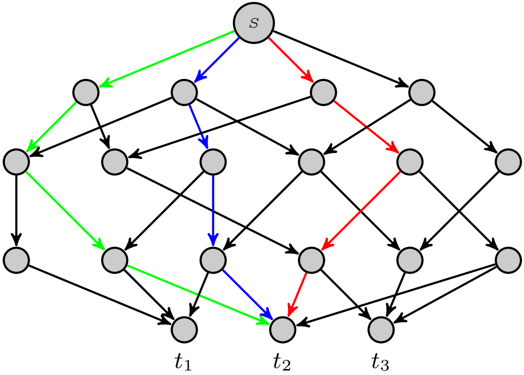
- Network: one source, three sinks, multicast capacity 3

Linear Information Flow Algorithm



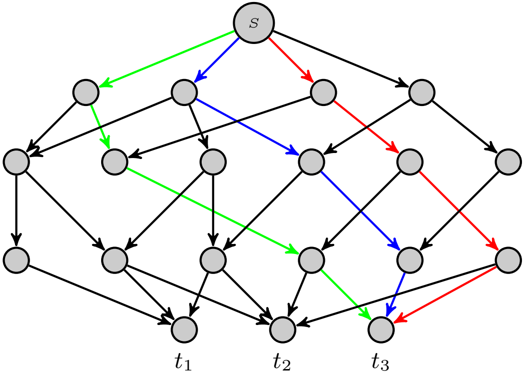
- Network: one source, three sinks, multicast capacity 3
- find 3 edge-disjoint paths to t_1

Linear Information Flow Algorithm



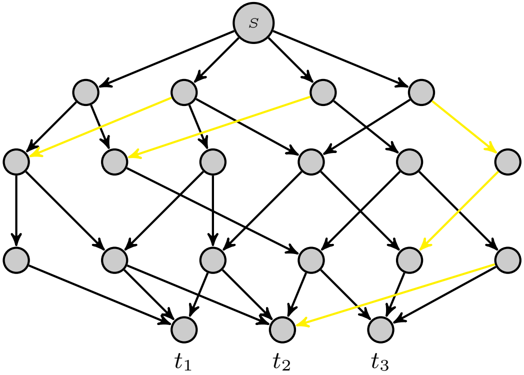
- Network: one source, three sinks, multicast capacity 3
- find 3 edge-disjoint paths to t_1
- find 3 edge-disjoint paths to t_2

Linear Information Flow Algorithm



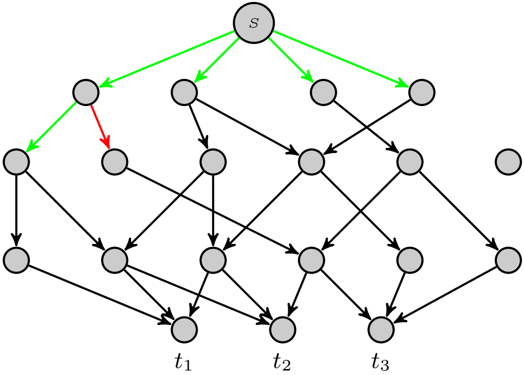
- Network: one source, three sinks, multicast capacity 3
- find 3 edge-disjoint paths to t_1
- find 3 edge-disjoint paths to t_2
- find 3 edge-disjoint paths to t_3

Linear Information Flow Algorithm



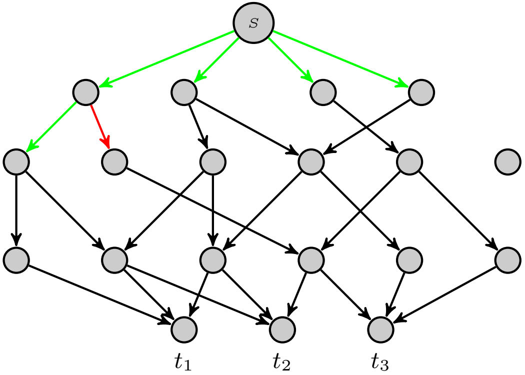
- Network: one source, three sinks, multicast capacity 3
- find 3 edge-disjoint paths to t_1
- find 3 edge-disjoint paths to t_2
- find 3 edge-disjoint paths to t_3
- discard unused vertices and edges

Linear Information Flow Algorithm



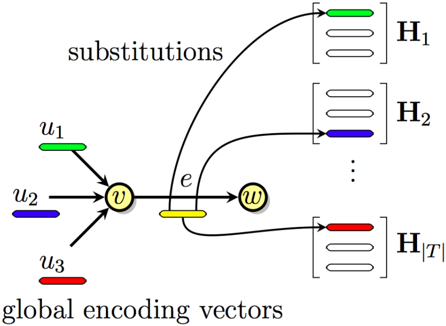
- Network: one source, three sinks, multicast capacity 3
- find 3 edge-disjoint paths to t_1
- find 3 edge-disjoint paths to t_2
- find 3 edge-disjoint paths to t_3
- discard unused vertices and edges
- process edges in topological order (i.e., upstream before downstream)

Linear Information Flow Algorithm



- Network: one source, three sinks, multicast capacity 3
- find 3 edge-disjoint paths to t_1
- find 3 edge-disjoint paths to t_2
- find 3 edge-disjoint paths to t_3
- discard unused vertices and edges
- process edges in topological order (i.e., upstream before downstream)

Idea: maintain the invariant that the transfer matrix induced so far at each sink is invertible.



- e must be a linear combination of u_1, u_2 and u_3
- e, which replaces rows in some transfer matrices, must maintain their full rank

- Suppose u_1, u_2 and u_3 span an m-dimensional space V_1.
- Assuming multicast capacity r, at a sink node t, the vectors not being replaced span an $(r-1)$-dimensional space V_2.
- We have $\dim(V_1 \cap V_2) = \dim(V_1) + \dim(V_2) - \dim(V_1 \cup V_2) = m - 1$
- All vectors in $V_1 \cap V_2$ are ruled out as choices for .
- In total, the zero vector + at most $|T|(q^{m-1} - 1)$ nonzero vectors are ruled out, leaving at least

$$q^m - 1 - |T|(q^{m-1} - 1) = q^{m-1}(q - |T|) + |T| - 1$$

choices for e, which is strictly positive as long as $q \geq |T|$.

Random Network Coding

As observed by Tracey Ho, et al. [HMKKESL,2006], choosing the local network coding coefficients at random leads to full rank transfer matrices with high probability if the field size is sufficiently large.
If η random choices must be made,

$$P[success] \geq (1 - |T|/q)^{\eta} \geq 1 - \frac{\eta|T|}{q},$$

or

$$P[failure] \leq \frac{\eta|T|}{q} = \eta|T|2^{-m}$$

for $q = 2^m$, i.e., exponential decrease with the number of bits per field element.
Thus, a network code chosen at random is highly likely to achieve the multicast capacity, if q is large enough.

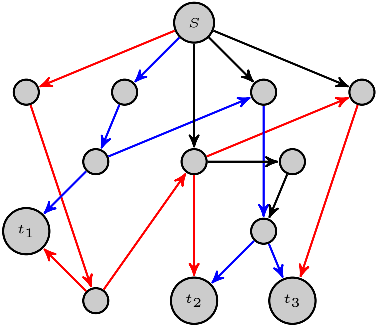
Routing versus Coding

Multicast by routing:

- requires packing of multiple distribution trees
- optimal packing may not achieve multicast capacity
- optimal fractional packing of distribution trees is NP hard [Jain, Mahdian, Salavatipour, SODA2003]

Multicast by linear network coding:

- achieves multicast capacity
- optimal solution can be found efficiently (polynomial time)



Multicasting via distribution trees

Linearly-coded Networks

We distinguish between two models for network coding:

1. Coherent: the network is given, and the local coding coefficients are fixed at design time (so that A_i is known at receiver t_i, where A_i is the transfer matrix of t_i)
2. Noncoherent: the local coding coefficients are chosen randomly at run time (so that A_i is not known to the transmitters or receivers)

Random (Noncoherent) Linear Network Coding

- Nodes draw coding coefficients uniformly at random from $\mathbb{F}_q$
- The transfer matrix will be invertible with high probability if q is sufficiently large
- The transfer matrix can be recorded by appending a header to each original packet

$$X = [I\, D]$$

$$Y = AX = [A\, AD] \Rightarrow A^{-1}Y = [I\, D] = X$$

- Captures most of the practical applications of network coding

Conclusion

Linear network codes:

- achieve the multicast capacity for sufficiently large q
- can be chosen randomly without sacrificing optimality

Thanks for your attention!