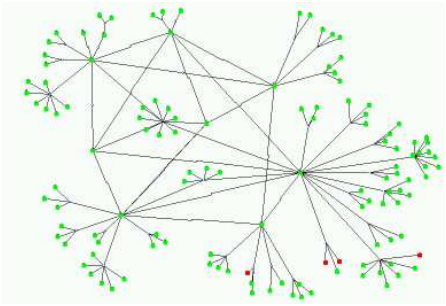
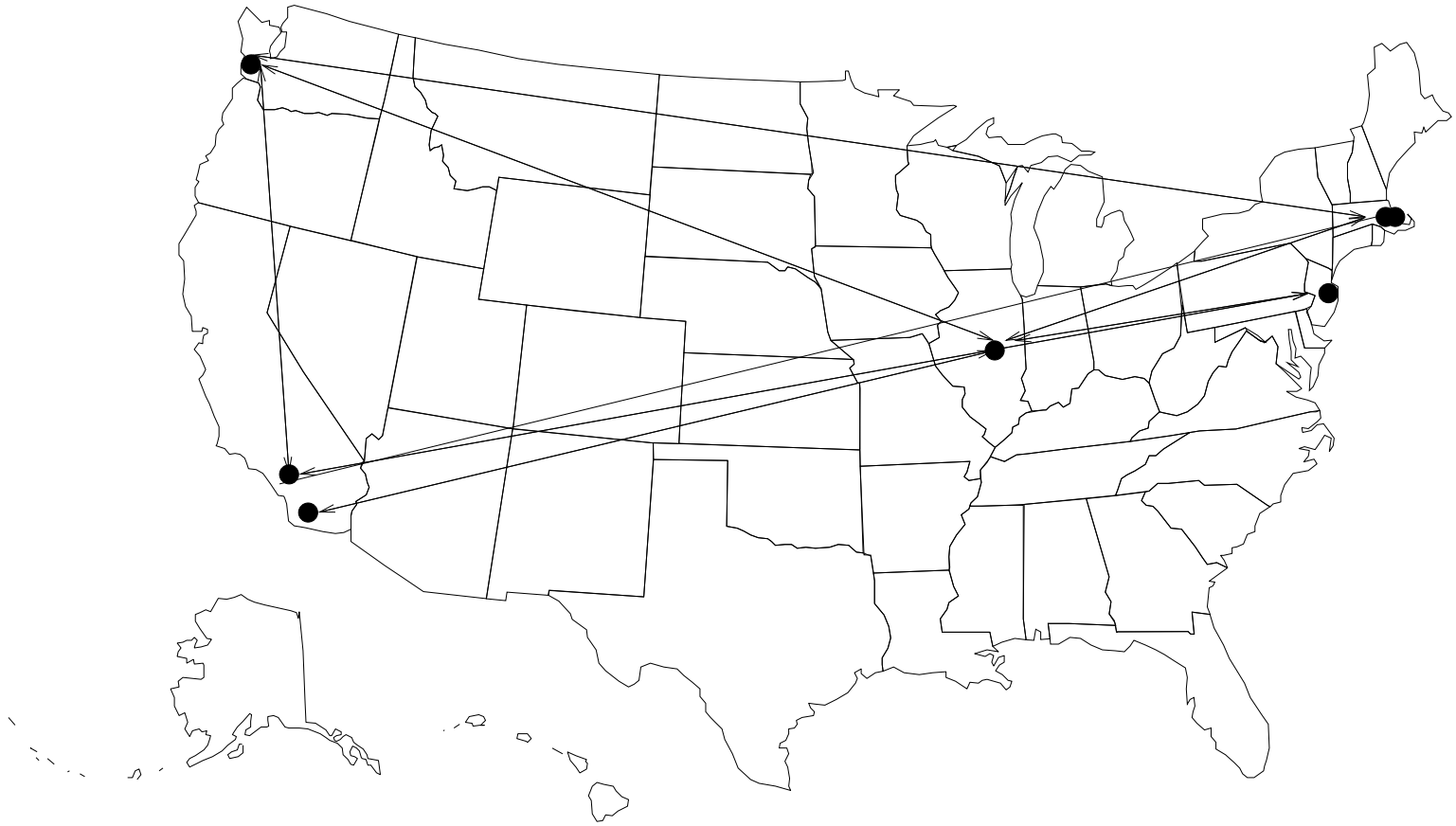
Network coding for the non-multicast case

Ralf Koetter, Coordinated Science Lab.,
University of Illinois

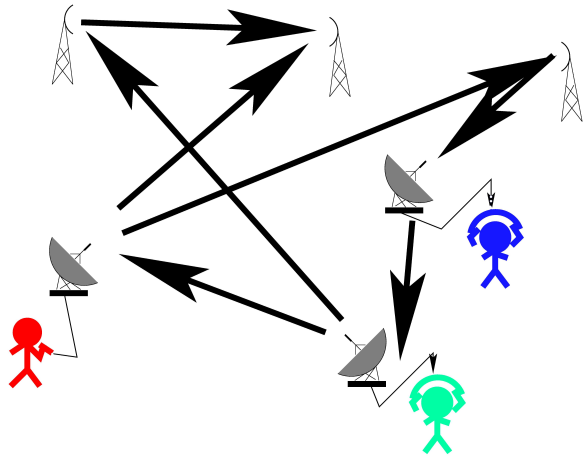


e-mail: koetter@csl.uiuc.edu

The network coding network for this talk .....

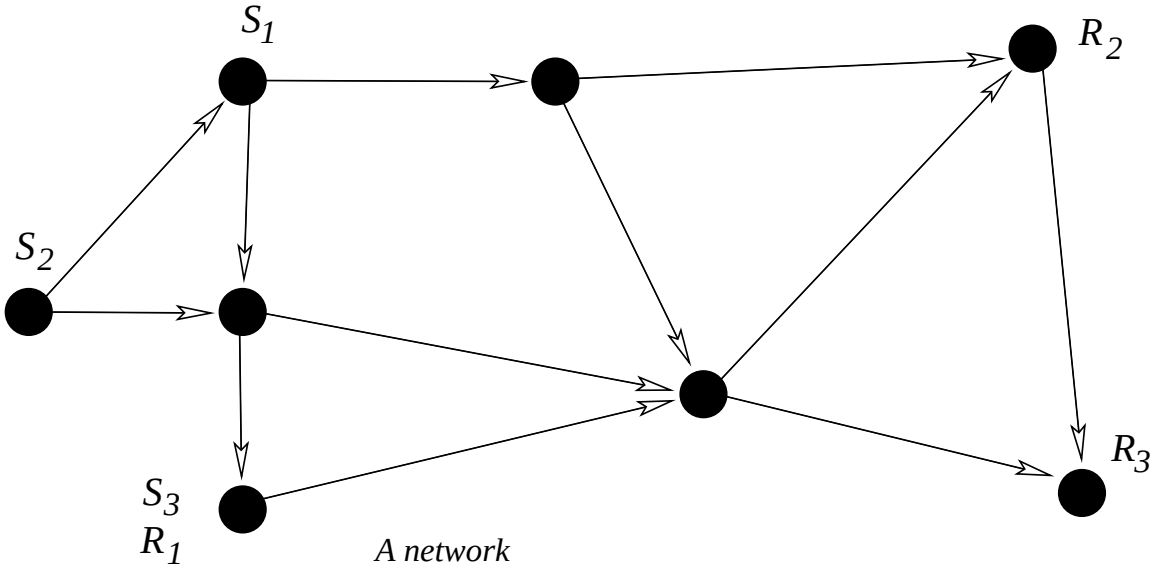


M. Médard, M. Effros, N. Ratnakar, T. Ho, D. Lun, ...



- What is capacity?
- How robustly can we communicate?
- Do we know the network?
- How do we achieve capacity?
- ??????

Problem Description



A network

Vertices: V
Edges: $E \subseteq V \times V$, $e = (v, u) \in E$
Edge capacity: $C(e)(= 1)$
Network: $\mathcal{G} = (V, E)$
Source nodes: $\{v_1, v_2, \ldots, v_N\} \subseteq V$
Sink nodes: $\{u_1, u_2, \ldots, u_K\} \subseteq V$

μ (rate one) input random processes at v:
$\mathscr{X}(v) = \{X(v,1), X(v,2), \ldots, X(v,\mu(v)\}$

ν Output random processes at u:
$\mathscr{Z}(u) = \{Z(u,1), Z(u,2), \ldots, Z(u,\nu(u))\}$

Random processes on edges: $Y(e)$

A connection:
$c = (v, u, \mathscr{X}(v,u))$, $\mathscr{X}(v,u) \subseteq \mathscr{X}(v)$

A connection is established if $\mathscr{Z}(u) \supset \mathscr{X}(v,u)$

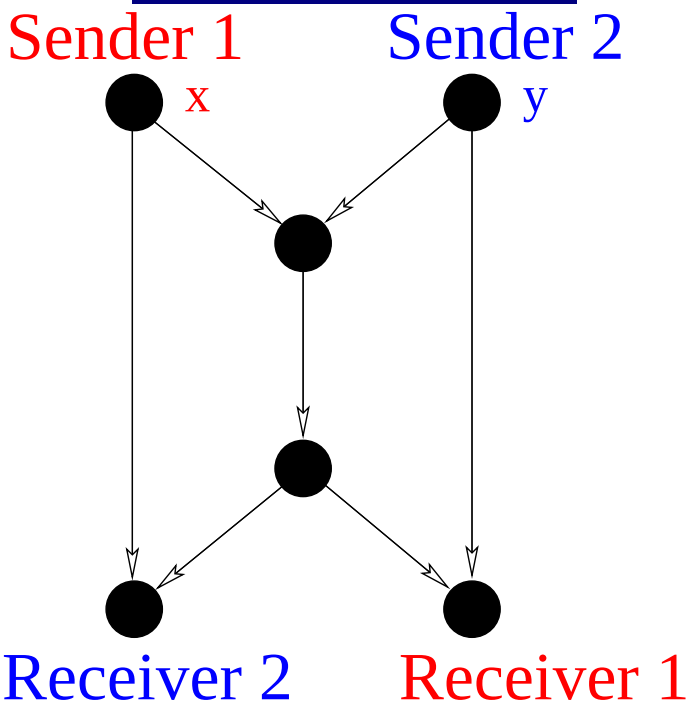
Set of connections: $\mathscr{C}$

The pair $(\mathcal{G}, \mathscr{C})$ defines a network coding problem .

Is the problem $(\mathcal{G}, \mathcal{C})$ solvable?

How do we find a solution?

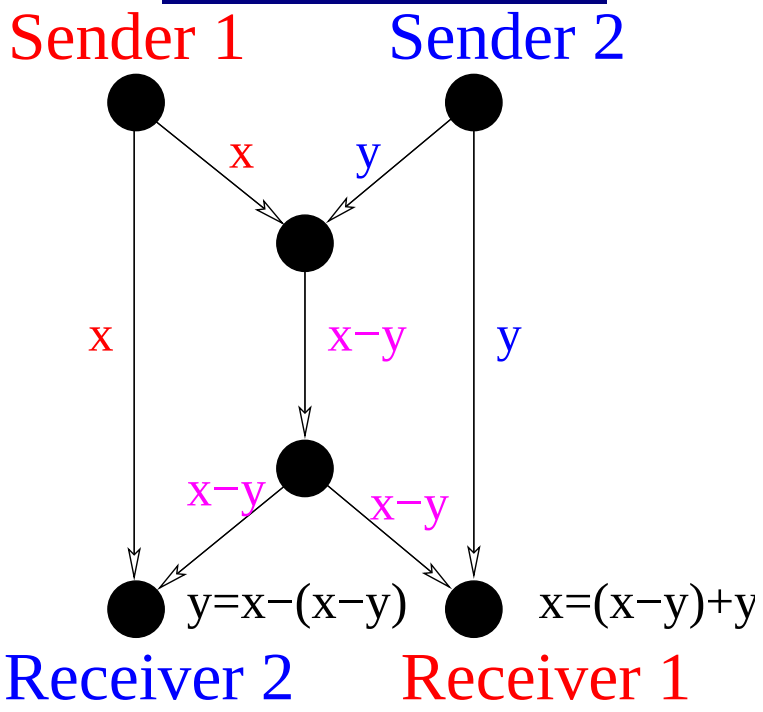
An Example



[1] Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", IEEE-IT, vol. 46, pp. 1204-1216, 2000

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai "Linear Network Coding", preprint, 2000

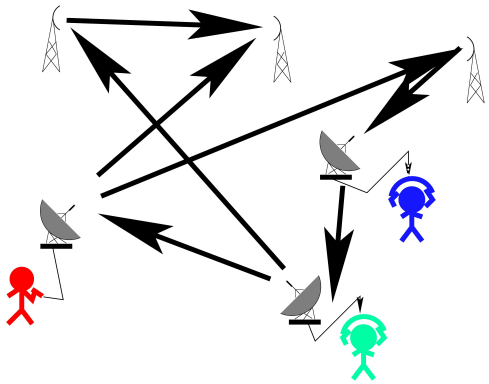
An Example



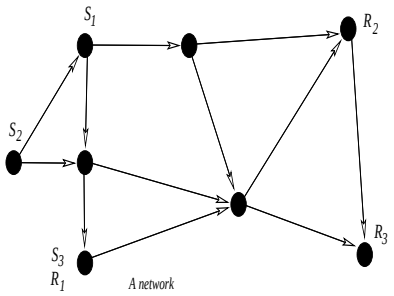
[1] Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", IEEE-IT, vol. 46, pp. 1204-1216, 2000

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai "Linear Network Coding", preprint, 2000

Towards an "algebraic" characterization?

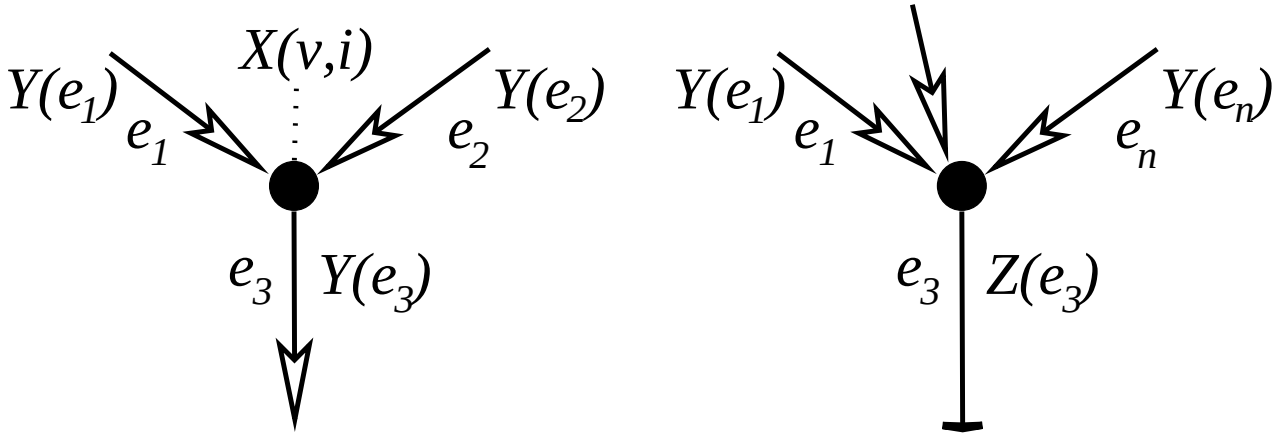


$\Rightarrow$



$\Rightarrow \quad A(I-F)^{-1}B^T = I$

All operations at network nodes are assumed linear*!

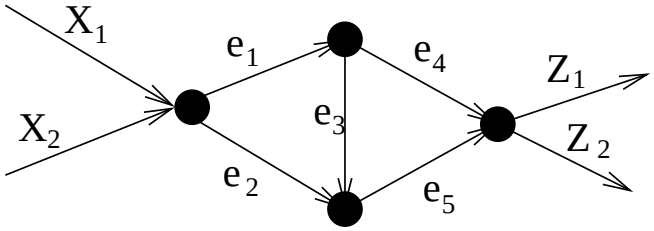


$$Y(e_3) = \sum_i \alpha_i X(v,i) + \sum_{j=1,2} \beta_j Y(e_j)$$

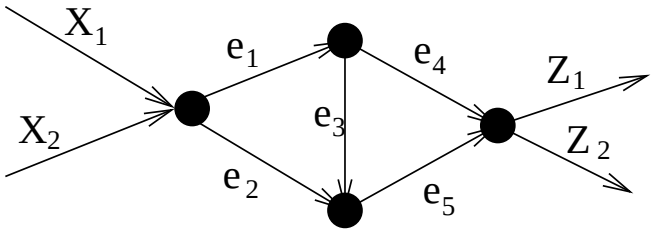
$$Z(v,j) = \sum_{j=1}^{n} \varepsilon_j Y(e_j).$$

$*\mathbb{F}_{2^m}$ is the finite field with m elements: we can add, subtract, divide and multiply elements in $\mathbb{F}_{2^m}$ without going crazy!

A simple example



$$
\begin{aligned}
Y(e_1) &= \alpha_{1,e_1} X_1 + \alpha_{2,e_1} X_2 \\
Y(e_2) &= \alpha_{1,e_2} X_1 + \alpha_{2,e_2} X_2 \\
Y(e_3) &= \beta_{e_1,e_3} Y(e_1) \\
Y(e_4) &= \beta_{e_1,e_4} Y(e_1) \\
Y(e_5) &= \beta_{e_2,e_5} Y(e_2) + \beta_{e_3,e_5} Y(e_3) \\
Z_1 &= \varepsilon_{e_4,1} Y(e_4) + \varepsilon_{e_5,1} Y(e_5) \\
Z_2 &= \varepsilon_{e_4,2} Y(e_4) + \varepsilon_{e_5,2} Y(e_5)
\end{aligned}
$$



In matrix form (after solving the linear system)

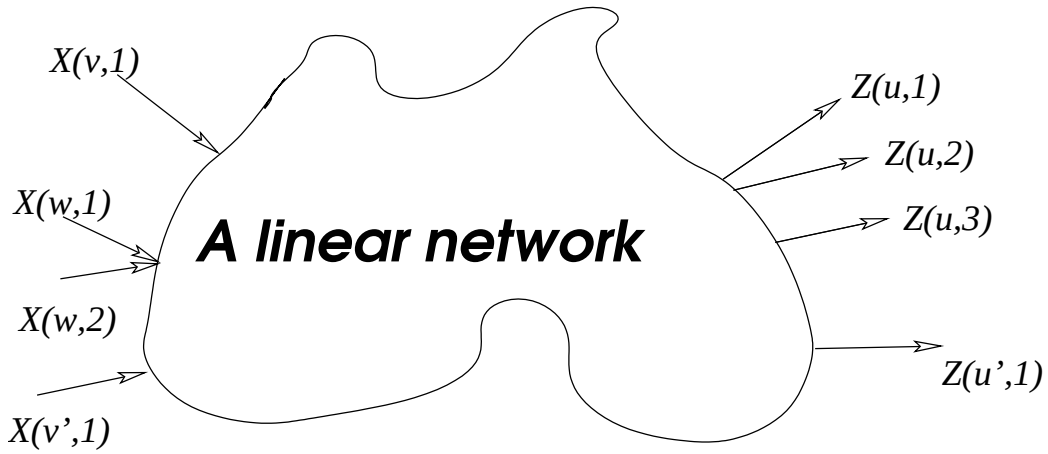
$$\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \varepsilon_{e_4,1} & \varepsilon_{e_5,1} \\ \varepsilon_{e_4,2} & \varepsilon_{e_5,2} \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} \beta_{e_1,e_4} & 0 \\ \beta_{e_1,e_3}\beta_{e_3,e_5} & \beta_{e_2,e_5} \end{pmatrix}}_{G} \underbrace{\begin{pmatrix} \alpha_{1,e_1} & \alpha_{1,e_2} \\ \alpha_{2,e_1} & \alpha_{2,e_2} \end{pmatrix}}_{A} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

We define three matrices A, G, B

The main question becomes: Is G invertible?

We collect all parameters as: $\underline{\xi} = (\xi_1, \xi_2, \dots,) = (\dots, \alpha_{e,l}, \dots, \beta_{e',e}, \dots, \varepsilon_{e',j}, \dots)$

A linear system



Input vector: $\underline{x}^T = (X(v,1), X(v,2), \ldots, X(v', \mu(v')))$

Output vector: $\underline{z}^T = (Z(u,1), Z(u,2), \ldots, Z(u', \nu(u')))$

Transfer matrix: M, $\underline{z} = M\underline{x} = B \cdot G \cdot A\ \underline{x}$

$\underline{z} = M\underline{x} = B \cdot G^T \cdot A\ \underline{x}$

where the entries in B, G, A are functions of the weights $\ldots, \alpha_{e,l}, \ldots, \beta_{e',e}, \ldots, \varepsilon_e$

$\underline{\xi} = (\xi_1, \xi_2, \ldots,) = (\ldots, \alpha_{e,l}, \ldots, \beta_{e',e}, \ldots, \varepsilon_{e',j}, \ldots)$

For acyclic networks the elements of G (and hence M) are polynomial functions in variables $\underline{\xi} = (\xi_1, \xi_2, \ldots,)$

$\Rightarrow$ an algebraic characterization of flows....

An algebraic Min-Cut Max-Flow condition

Let network be given with a source v and a sink v'. The following three statements are equivalent:

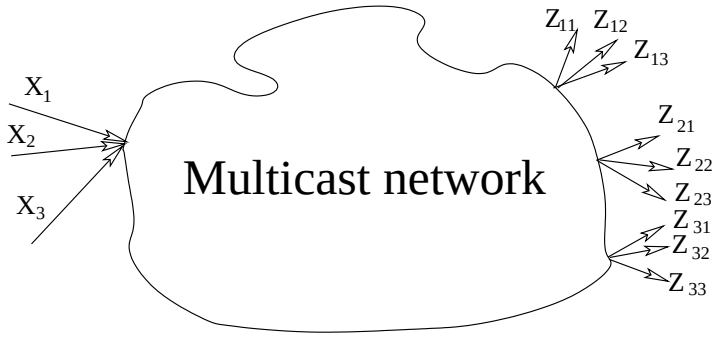
1. A point-to-point connection $c = (v, v', \mathscr{X}(v, v'))$ is possible.

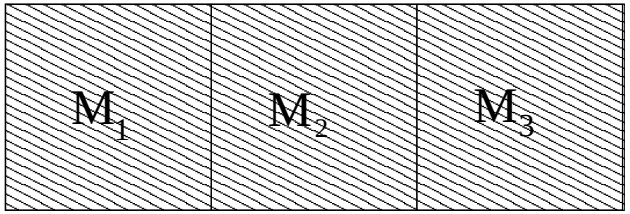
2. The Min-Cut Max-Flow bound is satisfied for a rate $R(c) = |\mathscr{X}(v, v')|$.

3. The determinant of the $R(c) \times R(c)$ transfer matrix M is nonzero over the ring of polynomials $\mathbb{F}[\underline{\xi}]$

3. $\Rightarrow$ We have to study the solution sets of polynomial equations.

Multicast:





System Transfer matrix

$\mathscr{C} = \{(v, u_1, \mathscr{X}(v)), (v, u_2, \mathscr{X}(v)), \ldots, (v, u_K, \mathscr{X}(v))\}$

M is a $|\mathscr{X}(v)| \times K|\mathscr{X}(v)|$ matrix.

$m_i(\underline{\xi}) = \det(M_i(\underline{\xi}))$

Choose the coefficients in $\overline{\mathbb{F}}$ so that all $m_i(\underline{\xi})$ are unequal to zero.

Find a solution of $\prod_i m_i(\underline{\xi}) \neq 0$

An innocent looking Lemma

Let $\mathbb{F}[X_1, X_2, \ldots, X_n]$ be the ring of polynomials over an infinite field $\mathbb{F}$ in variables $X_1, X_2, \ldots, X_n$. For any non-zero element $f \in \mathbb{F}[X_1, X_2, \ldots, X_n]$ there exists an infinite set of n-tuples $(x_1, x_2, \ldots, x_n) \in \mathbb{F}^n$ such that $f(x_1, x_2, \ldots, x_n) \neq 0$.

An innocent looking Lemma

Let $\mathbb{F}[X_1, X_2, \ldots, X_n]$ be the ring of polynomials over an infinite field $\mathbb{F}$ in variables $X_1, X_2, \ldots, X_n$. For any non-zero element $f \in \mathbb{F}[X_1, X_2, \ldots, X_n]$ there exists an infinite set of n-tuples $(x_1, x_2, \ldots, x_n) \in \mathbb{F}^n$ such that $f(x_1, x_2, \ldots, x_n) \neq 0$.

$(x^6 - x^4 - x^2 + x)$ does not have have a non-solution in $\mathbb{F}_2$, $\mathbb{F}_3$, $\mathbb{F}_4$ but in $\mathbb{F}_5$ we have $2^6 - 2^4 - 2^2 + 2 = 46 \equiv 1 (mod\ 5)$.

The main Multicast Theorem:

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem. There exists a linear network coding solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}_{2^m}$ for some large enough m if and only if there exists a flow of sufficient capacity between the source and each sink individually.

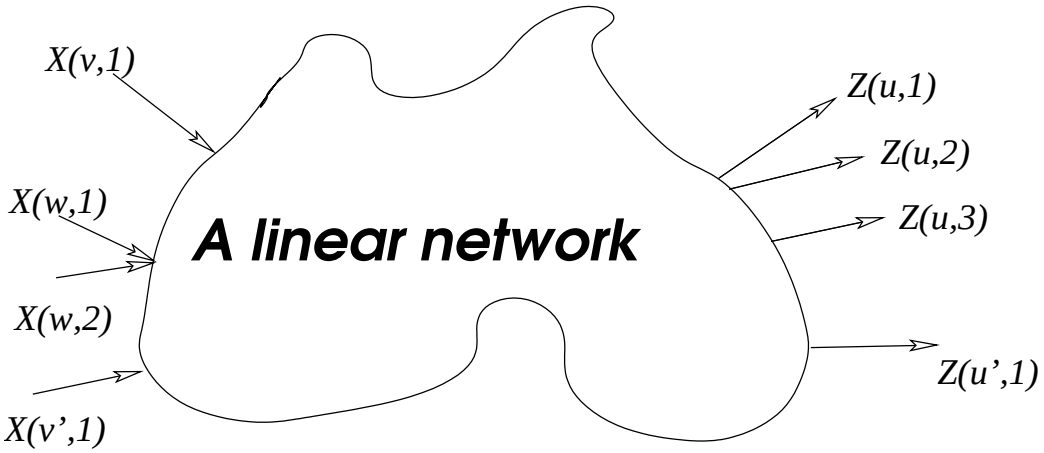
Random Network Coding

For large fields ($q \sim 2^{20}$) randomly finding a solution to the polynomial equations is very unlikely.

A random assignment will work with high probability.
Each node chooses coefficients at random.

The compound effect of the choice can be tracked by measuring the MIMO impulse response with pilot tones (as part of the packet of separately $\Rightarrow$ noncoherent communication)

One reason the non-multicast case is difficult - linear network coding

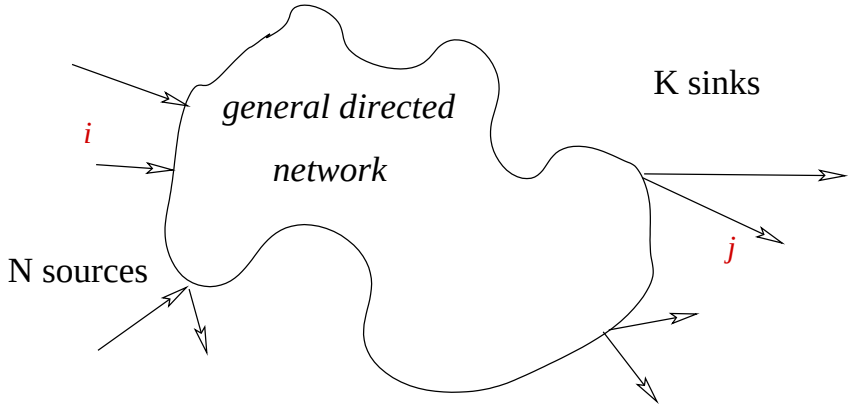


Input vector: $\underline{x}^T = (X(v,1), X(v,2), \ldots, X(v', \mu(v')))$

Output vector: $\underline{z}^T = (Z(u,1), Z(u,2), \ldots, Z(u', \nu(u')))$

Transfer matrix: M, $\underline{z} = M\underline{x}$

The non-multicast problem $(\mathcal{G}, \mathscr{C})$



$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,K} \\ M_{2,1} & M_{2,2} & & M_{2,K} \\ \vdots & M_{i,j} & & \vdots \\ M_{N,1} & M_{N,2} & \cdots & M_{N,K} \end{pmatrix}$$

$M_{i,j}$: transfer matrix between source i and sink j.

Some $M_{i,j}$ have to be non-singular — Some $M_{i,j}$ have to be zero!

So why is the general case so much harder?

For the general case we need to find solutions, i.e. zeros of some system of polynomial equations!

For the multicast case we need to find non solutions to some system of polynomial equations!

Another way to phrase this is: In a multicast setup everybody wants everything so the issue of interference is moot (there is no problem at all to distribute random processes of entropy rate corresponding to the demands)!

For the general case we may have carefully balanced solutions where some unwanted information cancels out in clever ways.....

The good news

There exists an algorithm (based on Grobner bases) to solve the general linear, scalar network coding problem!

(complexity critically dependent on maximal field size)

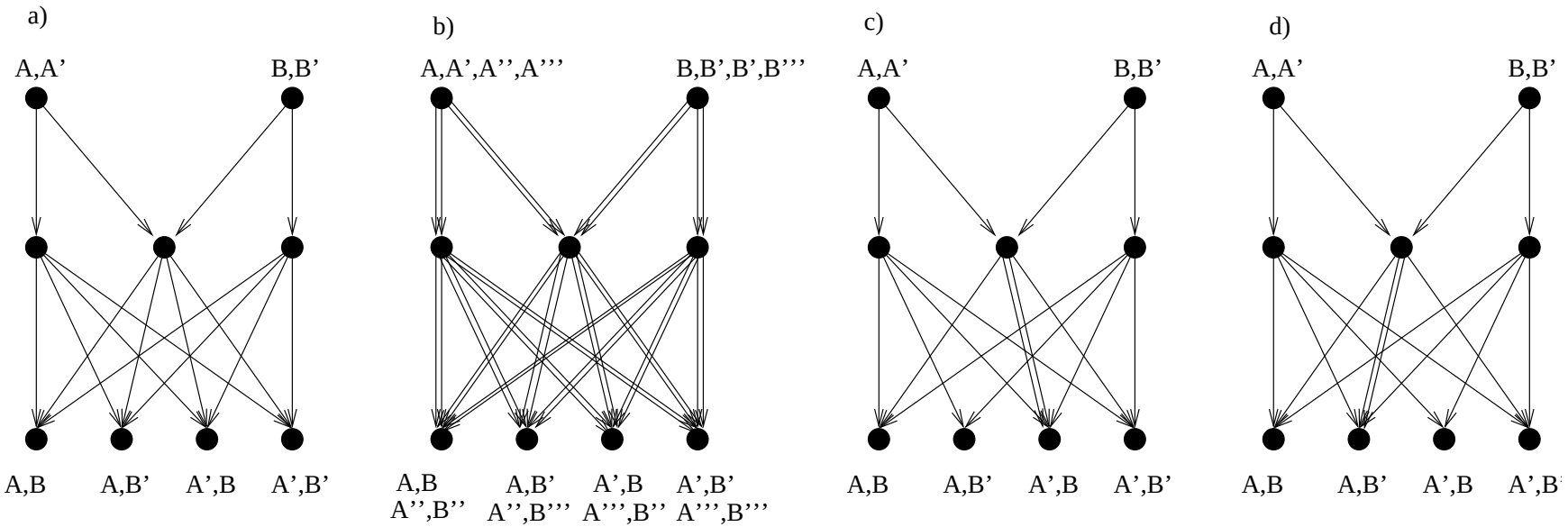
The bad news

From: A. Rasala-Lehman and E. Lehman, "Vector-Linear Network Codes: Is the Model Broken", preprint, March 2004

By combining networks requiring vector length that are multiples of primes the following bound is derived:

Theorem There exist directed networks with O(n) nodes such that a solution to the network coding problem requires at least an alphabet size of $2^{(e^{\sqrt{n^{1/3}}})}$

Worse news

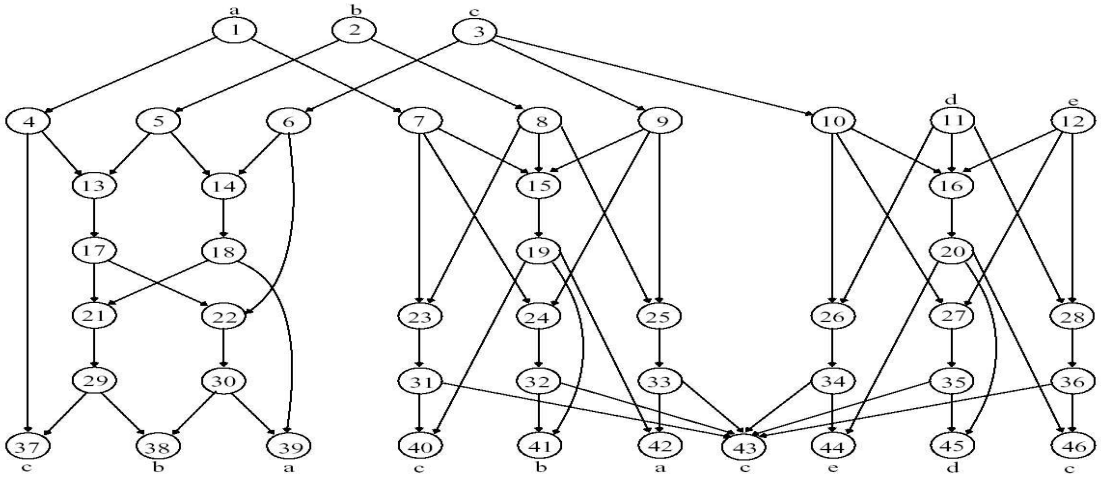


In the general problem a time sharing combination of several solutions which themselves violate the constraints may be necessary. (This cannot happen in the multicast case!)

Doubling the bandwidth more than doubles capacity! Tripling the bandwidth does not work! Not all network coding problems can be solved as convex combinations of scalar ones.

Really bad news... or maybe....

R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of Linear Coding in Network Information Flow", preprint, 2004



This network is not solvable over any Galois field, including vector versions thereof

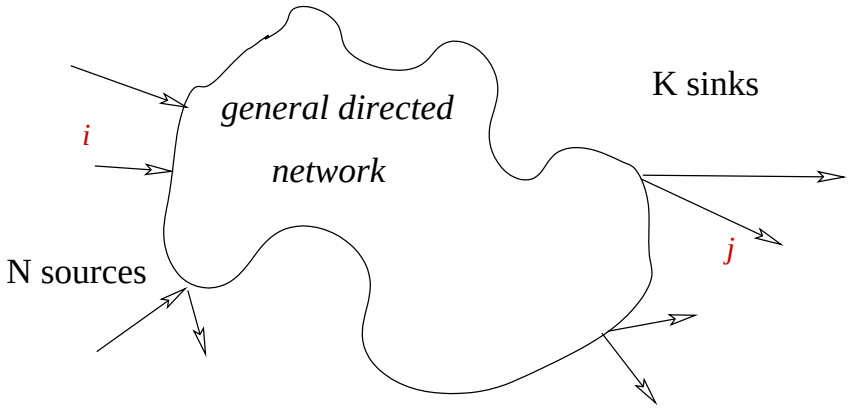
(still the network has a linear feel to it....)

Excellent news...

R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of Linear Coding in Network Information Flow", preprint, February 2004

So far we only have a collection of (very clever) countre examples — let's focus on practical constructions

Constructing non-optimal but good and generic solutions:

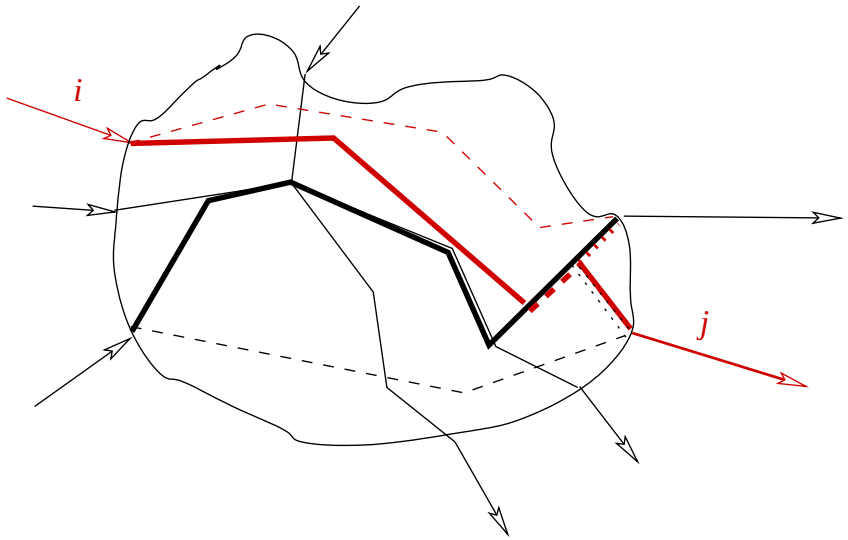


Scalability

Incremental/decremental solutions

Self-organizing,

Phasing in a new user



A new user and demand from i to j can only be accomodated if we can re-use a link in the network.

The dashed links provide the remedy we need to re-use a link.

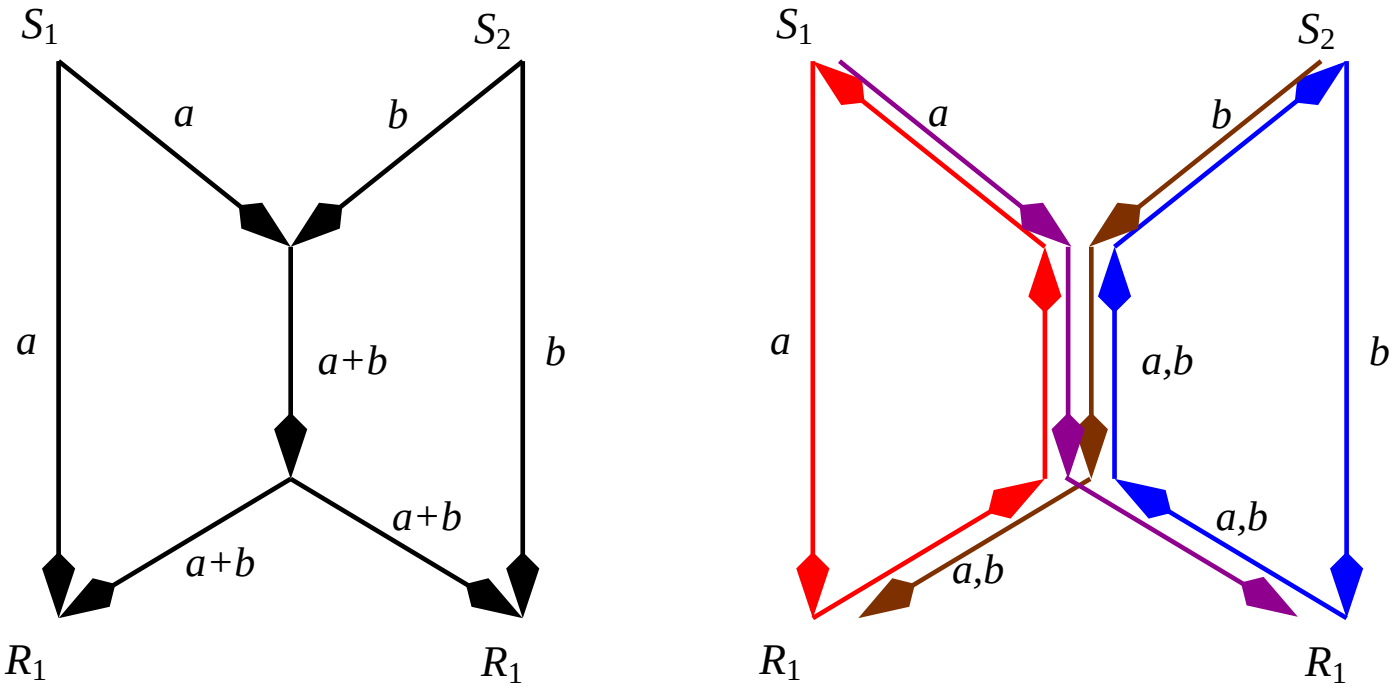
A classification of network coding solutions

Class 1: "Routing" - each Y_e is a function of at most on X_i

Class 2: Y_e is either 0, X_i or $X_i \oplus X_j$ - "butterfly" class

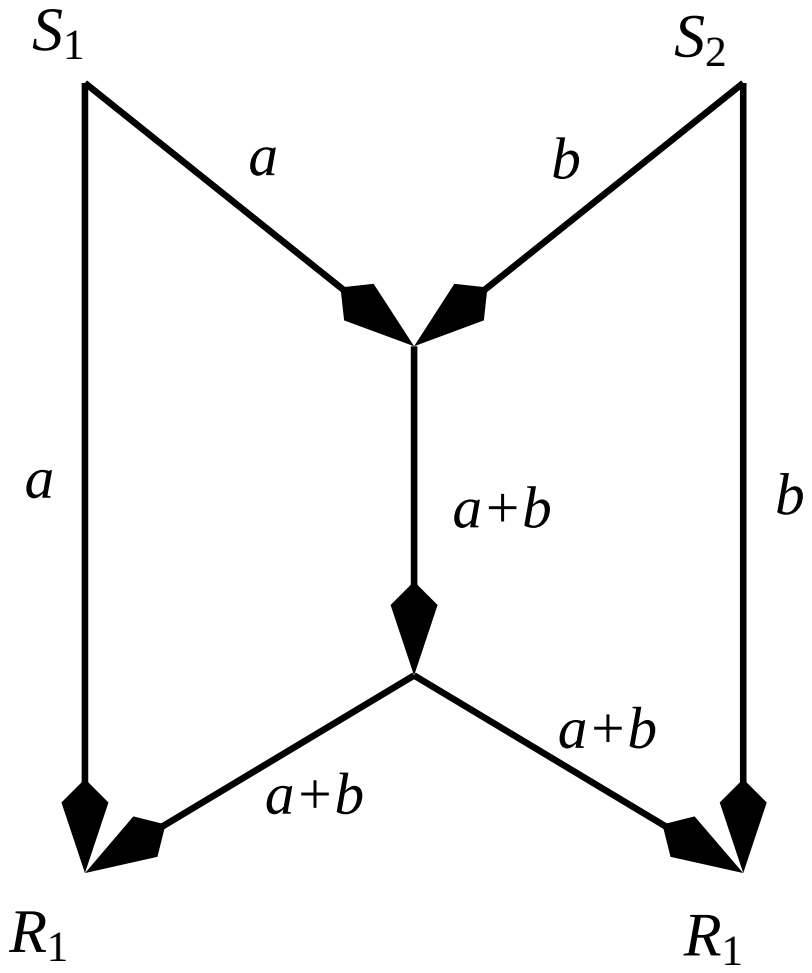
$\vdots$

Packing butterflies

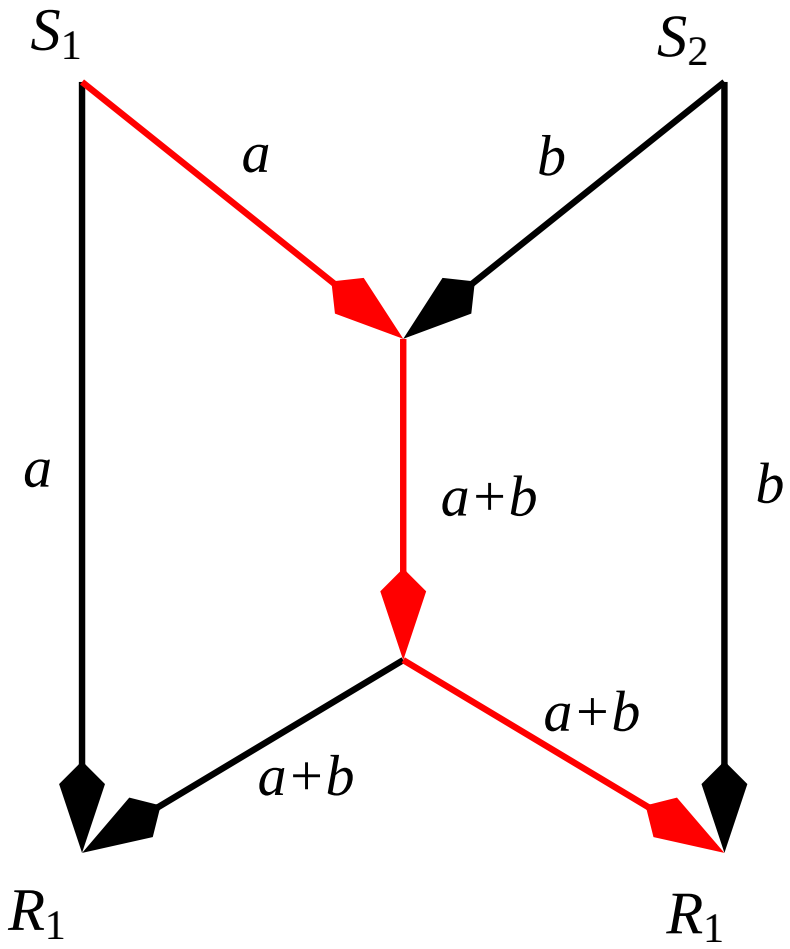


Generalizing the flow decomposition of butterfly networks we can formulate a linear program to solve Class II network coding problems!

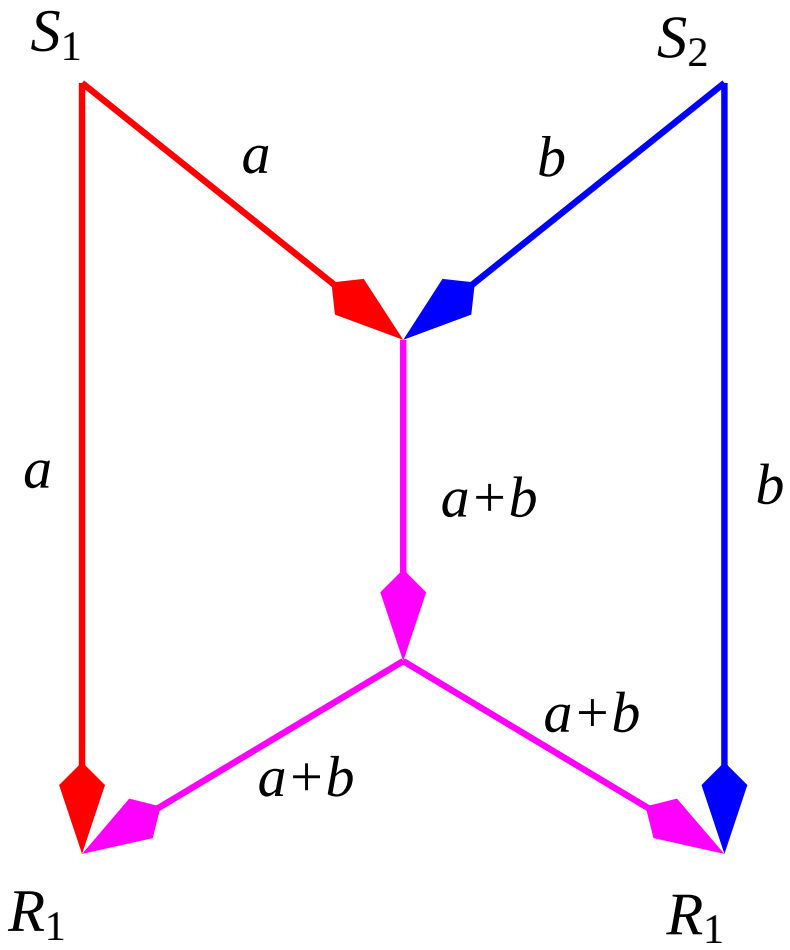
Finding solutions in Class 2 - a flow formulation



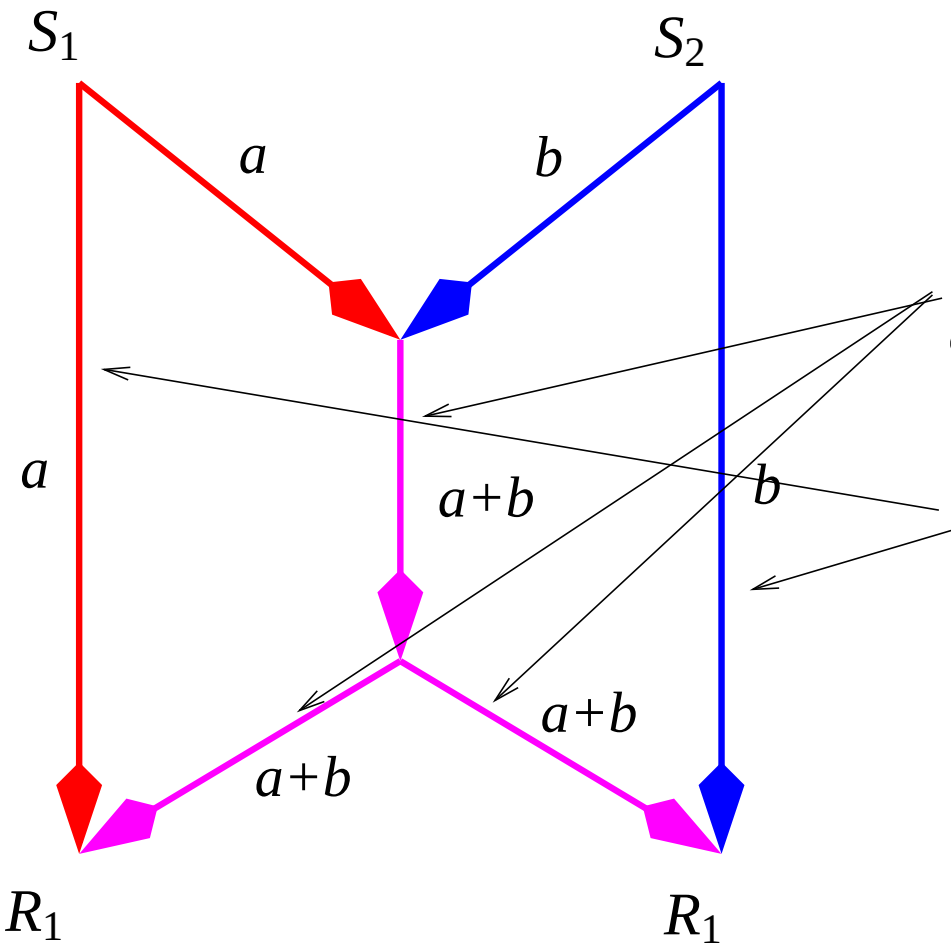
Finding solutions in Class 2 - a flow formulation



Finding solutions in Class 2 - a flow formulation



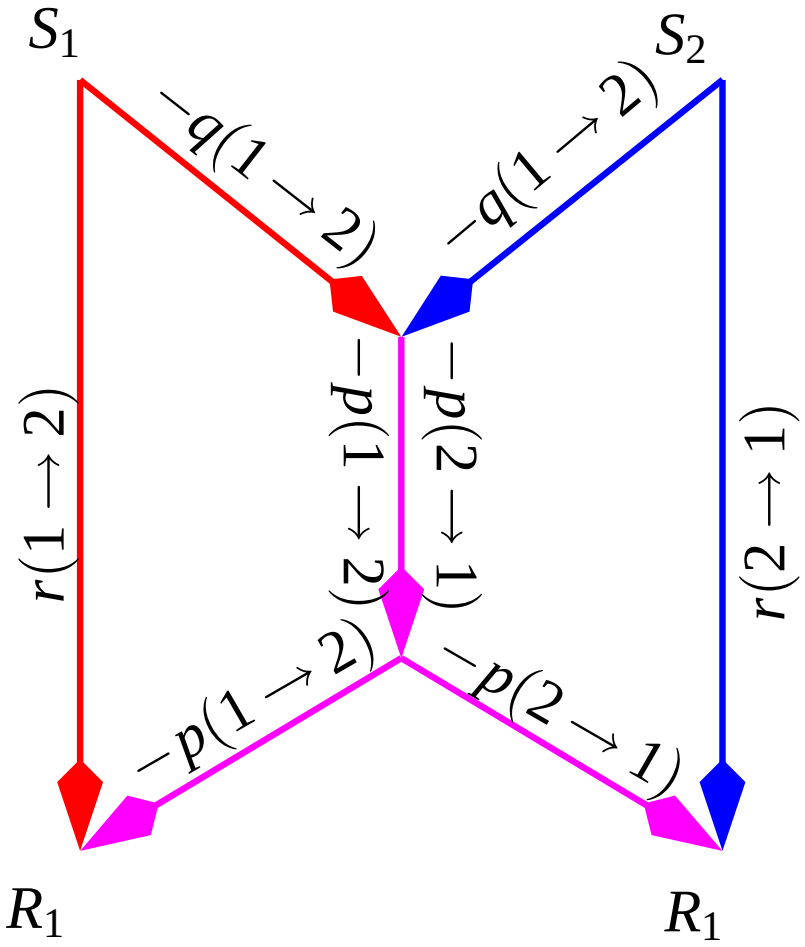
Finding solutions in Class 2 - a flow formulation



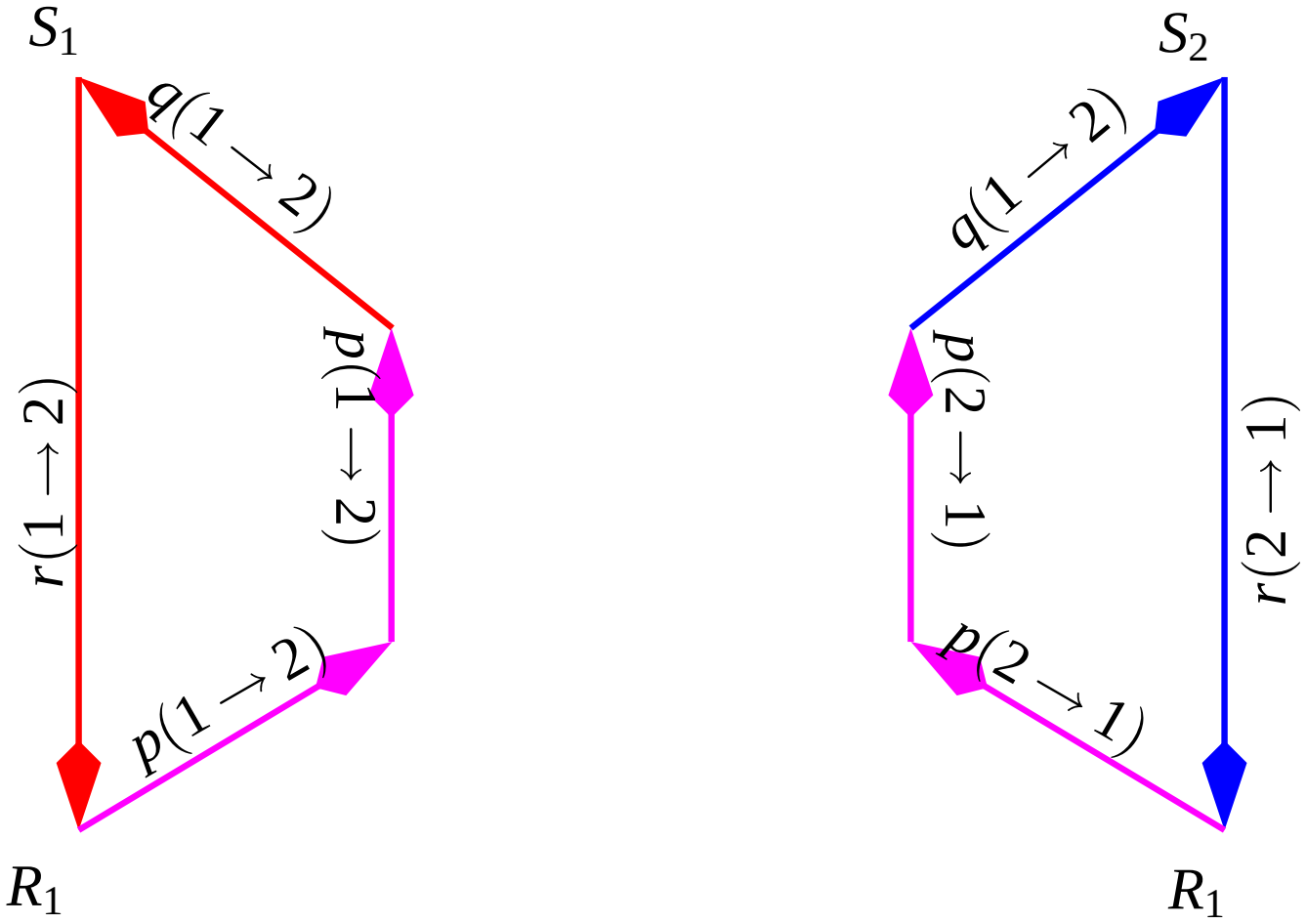
"poisened flows" (red poisened by blue and blue poisened by red)

"antidote flows"

Finding solutions in Class 2 - a flow formulation

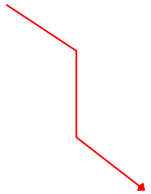


Finding solutions in Class 2 - a flow formulation

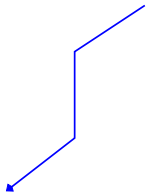


Observations

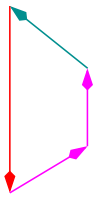
- $x(1)$ is a flow from S_1 to D_1.

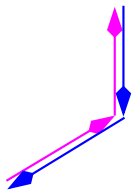
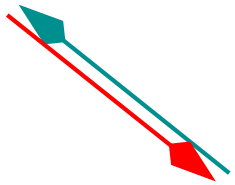
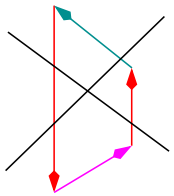


- $x(2)$ is a flow from S_2 to D_2.



- $p() + q(\cdot) + r(\cdot)$ forms a loop.



- $p(1 \rightarrow 2)$ is a 'virtual' flow with a 'host' $x(2)$. 
- $q(1 \rightarrow 2)$ is a 'virtual' flow with a 'host' $x(1)$. 
- $r(1 \rightarrow 2)$ is a flow that consumes resources and does not need a host. 
- $p(\cdot), q(\cdot)$, and $r(\cdot)$ are uninterrupted paths. 

Generalizing the idea

- $p_e(m \to n, u)$ for every edge e. u keeps track of the `origin' of the poison.

- Similarly $q_e(m \to n, u)$ and $r_e(m \to n, u)$.

- We will search for butterfly structures.

Searching for Butterfly structures

Allow loops made of $p(m \to n, u)$, $q(m \to n, u)$, and $r(m \to n, u)$.
For all nodes v, u, and for all flows m and n.

$$\sum_{e:\text{head}(e)=v} p_e(m \to n, u) + q_e(m \to n, u) + r_e(m \to n, u)$$
$$= \sum_{e:\text{tail}(e)=v} p_e(m \to n, u) + q_e(m \to n, u) + r_e(m \to n, u)$$

Searching for Butterfly structures

Ensure that each of $p(m \to n, u)$, $q(m \to n, u)$, and $r(m \to n, u)$ is an unbroken path. At node u

$$\sum_{e:\mathsf{head}(e)=u} q_e(m \to n, u) \leq \sum_{e:\mathsf{tail}(e)=u} q_e(m \to n, u) \qquad (1)$$

At any other node v,

$$\sum_{e:\mathsf{head}(e)=v} q_e(m \to n, u) \geq \sum_{e:\mathsf{tail}(e)=v} q_e(m \to n, u) \qquad (2)$$

Searching for Butterfly structures

Ensure that each of $p(m \to n, u)$, $q(m \to n, u)$, and $r(m \to n, u)$ is an unbroken path. At node u

$$\sum_{e:\text{head}(e)=u} p_e(m \to n, u) \geq \sum_{e:\text{tail}(e)=u} p_e(m \to n, u) \qquad (3)$$

At any other node v,

$$\sum_{e:\text{head}(e)=v} p_e(m \to n, u) \leq \sum_{e:\text{tail}(e)=v} p_e(m \to n, u) \qquad (4)$$

Searching for Butterfly structures

- If m-th and n-th flows overlap, "generate" poison (and consequently the loops).

- Ensure that a maximum of two flows are overlapping. This ensures that the butterfly structures are disjoint.

List of equations

$x_e(n)$ is a flow of the desired rate from S_n to D_n.

$$p_e(n \to m, u) = p_e(m \to n, u) \text{ if } \mathsf{tail}(e) = u \tag{5}$$

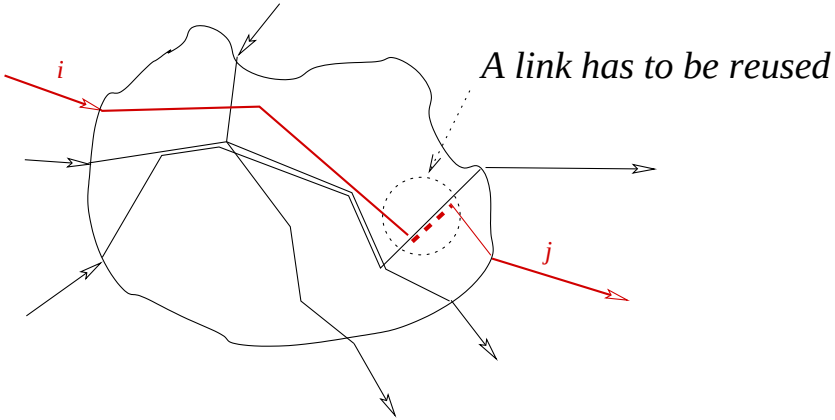
$$\overbrace{\sum_u \sum_{m,n} \max(p_e(m \to n, u), p_e(n \to m, u))}^{\text{the coding advantage}} + \underbrace{\sum_{i=1}^{n} x_e(i) + \sum_u \sum_{m,n} (r_e(m \to n, u) + r_e(n \to m, u))}_{\text{resource demanding flows}} \leq z_e \tag{6}$$

Virtual hosts

$$x_e(n) + \sum_u \sum_m p_e(m \to n, u) + q_e(m \to n, u) \geq 0 \qquad (7)$$

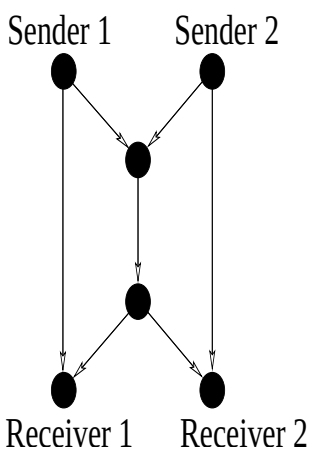
A solution to these equations can be used to identify the butterfly structures and a network coding solution can be computed.

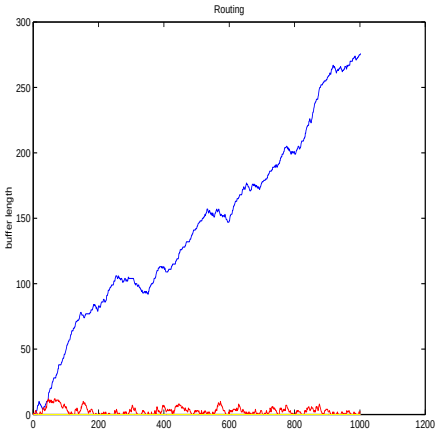
Queuing vs. Network Coding

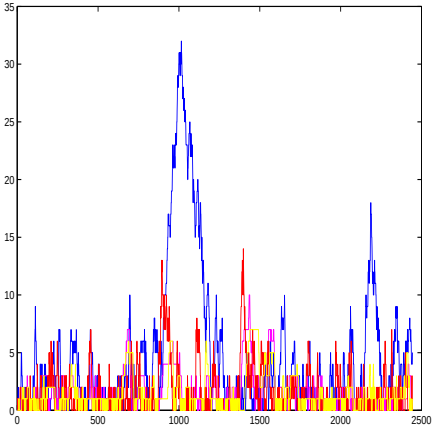


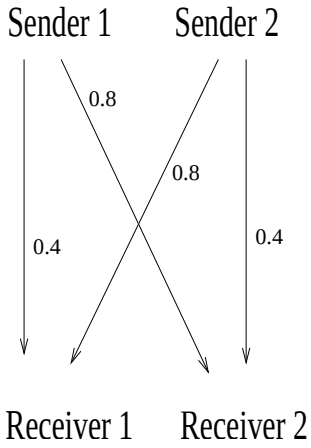
Should we queue packets or should we network-code them?

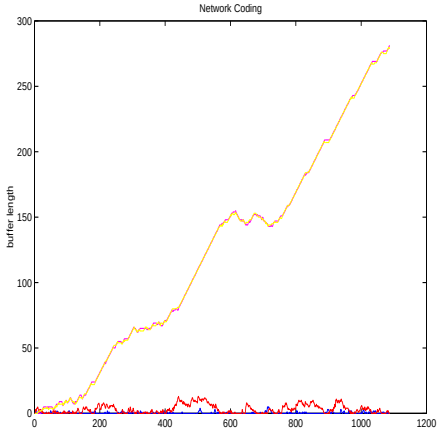
Queuing vs. Network Coding

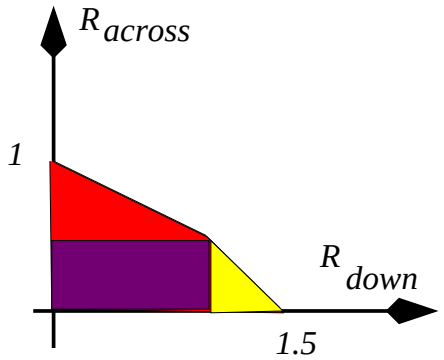












Some observations and statements

Network coding is a way to trade excess capacity in parts of the network for bottleneck capacity somewhere else.

Using an already used link comes at the price of providing other seemingly uncorrelated connections.

Network coding structures can be decomposed into these re-use and remedy patterns (with increasing levels of complexity)

Some observations and statements

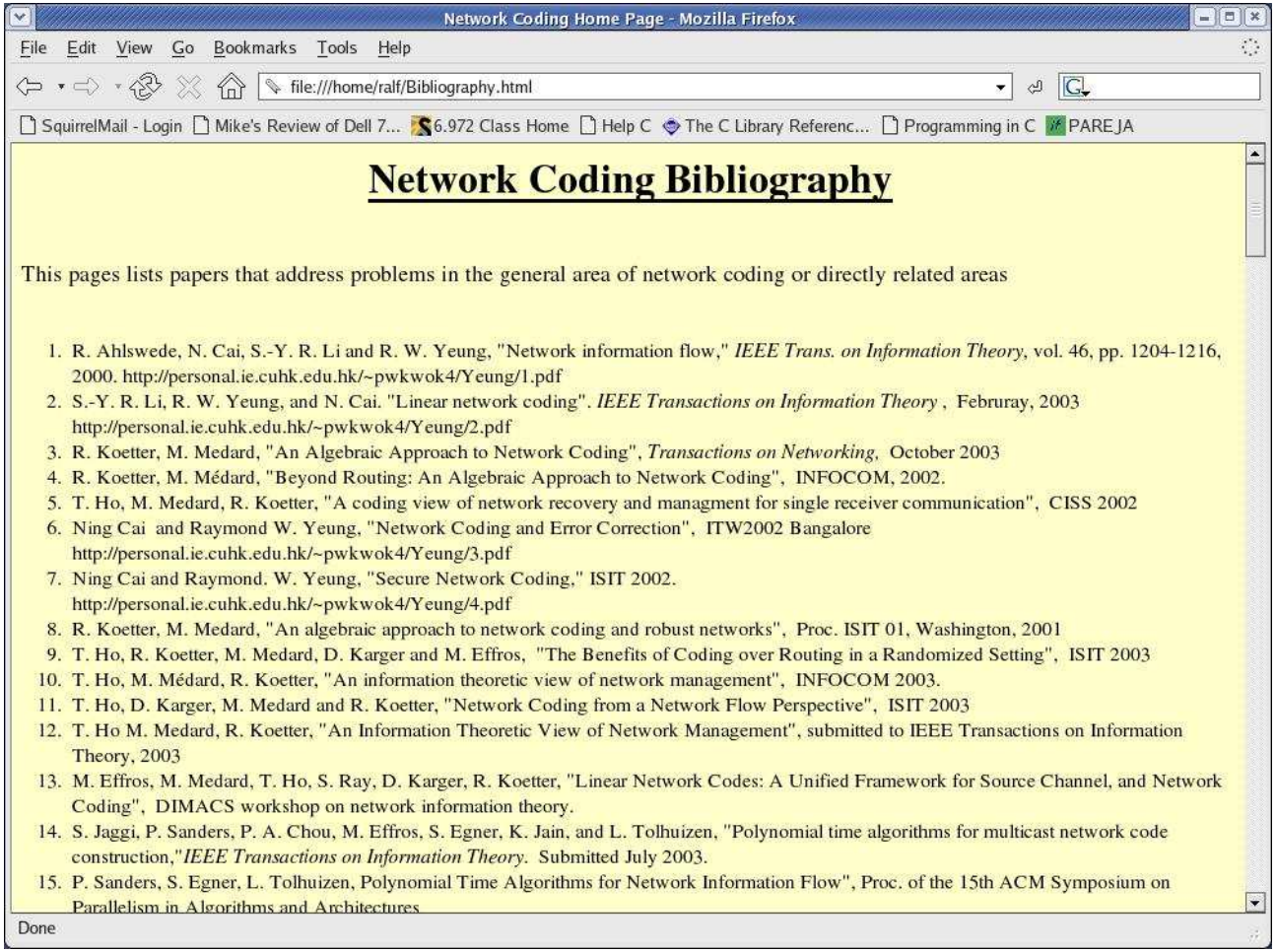
- The non-multicast case is far more difficult with even decide-ability of vector linear problems unknown.
- In general, linear solutions are not sufficient to achieve capac-ity
- Still, we do not need to only consider optimal solutions
- Class 2 network codes seem to be the sweet spot
- For Class 2 we can give an LP characterized by flows.

- The LP shows scalability, robustness and other desirable features.

- Network coding is characterized as transmitting evidence rather than information directly, We get all the benefits of diversity (not only the benefits of diversity routing).

Thanks!

www.networkcoding.info



Non-multicast connections -use of cost criterion

- We propose a linear optimization problem whose minimum cost is no greater than the minimum cost of any routing solution
- Moreover, feasible solutions correspond to network codes that perform linear operations on vectors created from the source processes
- Main idea: create a set partition of $\{1, \ldots, M\}$ that represents the sources that can be mixed (combined linearly) on links going into i.
- Code construction steps through the nodes in topological order, examining the outgoing links and defining global coding vectors on them.

Non-multicast connections -use of cost criterion

- For any node i, let $T(i)$ denote the sinks that are accessible from i
- Let $\mathcal{C}(i)$ be a set partition of $\{1, \ldots, M\}$ that represents the sources that can be mixed (combined linearly) on links going into i. For a given $C \in \mathcal{C}(i)$, the sinks that receive a source process in C by way of link (j, i) in A (set of arcs) either receive all the source processes in C or none at all.

Non-multicast connections -use of cost criterion

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{(i,j)\in A} a_{ij} z_{ij} \\
\text{subject to} \quad & c_{ij} \geq z_{ij} = \textstyle\sum_{C \in \mathcal{C}(j)} y_{ij}^{(C)}, && \forall\, (i,j) \in A, \\
& y_{ij}^{(C)} \geq \textstyle\sum_{m \in C} x_{ij}^{(t,m)}, && \forall\, (i,j) \in A, t \in T, C \in \mathcal{C}(j), \\
& x_{ij}^{(t,m)} \geq 0, && \forall\, (i,j) \in A, t \in T, m = 1, \ldots, M,
\end{aligned}$$

$$\sum_{\{j|(i,j)\in A\}} x_{ij}^{(t,m)} - \sum_{\{j|(j,i)\in A\}} x_{ji}^{(t,m)} = \begin{cases} R_m & \text{if } v = s_m \text{ and } m \in D(t), \\ -R_m & \text{if } m \in D(i), \\ 0 & \text{otherwise,} \end{cases}$$

$$\forall\, i \in A, t \in T, m = 1, \ldots, M, \quad (8)$$

where we define $D(i) := \emptyset$ for i in $N \setminus T$. Again, the optimization problem can be easily modified to accommodate convex cost functions.

Is the non-multicast case interesting?

Summary:

The non multicast scenario exhibits far more subtleties than the multicast setup. This is due to the fact that cancellations now need to be carefully arranged.

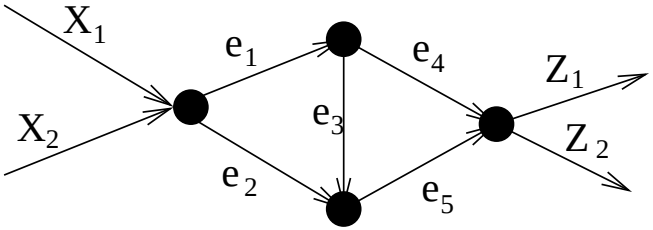
There are some generalizations to vector solutions which can be incorporated into the algebraic framework.

Not even the principle problem of linearity vs. nonlinear operation is entirely clear.

From a practical point of view a non interacting arrangement of multicast is most interesting and robust.

The transfer matrix

Let a matrix F be defined as an $|E| \times |E|$ matrix where $f_{i,j}$ is defined as β_{e_i,e_j}, i.e. the coefficient with which $Y(e_i)$ is mixed into Y_{e_j}.



$$F = \begin{pmatrix} 0 & 0 & \beta_{e_1,e_3} & \beta_{e_1,e_4} & 0 \\ 0 & 0 & 0 & 0 & \beta_{e_2,e_5} \\ 0 & 0 & 0 & 0 & \beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Summing the "path gains":

$$P = I + F + F^2 + \ldots = (I - F)^{-1} = \begin{pmatrix} 0 & 0 & \beta_{e_1,e_3} & \beta_{e_1,e_4} & \beta_{e_1,e_3}\beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & \beta_{e_2,e_5} \\ 0 & 0 & 0 & 0 & \beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Observe that $G = (I - F)^{-1}$ is polynomial

Young optimization students