Network Coding - an Introduction

Ralf Koetter and Muriel Medard
University of Illinois, Urbana-Champaign
Massachusetts Institute of Technology

Goals of Class

- To provide a general introduction to the new field of network coding
- To provide sufficient tools to enable the participants to apply and develop network coding methods in diverse applications
- To place network coding in the context of traditional network operation

Outline

- Basics of networks, routing and network coding:
  - Introduction to routing in traditional networks
    - routing along shortest paths
    - routing for recovery
  - Introduction to concepts of network coding
- Algebraic foundations:
  - Formal setup of linear network coding
  - Algebraic formulation
  - Algebraic min cut max flow condition
  - The basic multicast theorem
  - Other scenarios solvable with algebraic framework
  - Delays in networks

Outline (contd)

- More multicast - constructing codes
  - Coding gain is unbounded
  - Construction based on algebraic system
  - Construction based on flows
  - Undirected networks

Outline (contd)

- Decentralized code construction and network coding for multicast with a cost criterion
  - Randomized construction and its error behavior
  - Performance of distributed randomized construction - case studies
  - Robustness of randomized methods
  - Traditional methods based on flows - a review
  - Trees for multicasting - a review
  - Network coding with a cost criterion - flow-based methods for multicasting through linear programming
  - Distributed operation - one approach
  - A special case - wireless networks
  - Sample ISPs

Outline(contd)

- Non-multicast:
  - The algebraic difficulty
  - Vector solutions vs. instantaneous
  - Issue of linearity
  - Is the non-multicast case interesting?

Outline(contd)

- Network coding for multicast - relation to compression and generalization of Slepian-Wolf
  - Review of Slepian-Wolf
  - Distributed network compression
  - Error exponents
  - Source-channel separation issues
  - Code construction for finite field multiple access networks
- Network coding for security and robustness
  - Network coding for detecting attacks
  - Network management requirements for robustness
  - Centralized versus distributed network management
- New directions

Main topics

- Routing in networks operates in a manner akin to a transportation problem in which we seek to transport goods (data) in a cost-efficient fashion (multicast is a notable exception)
- Data is compressed and recovered at the edges
- Cost is defined according to a given cost of routes or by adjusting to the flows
- Current approaches do not generally make use of the fact that data (bits) are being transmitted

Shortest Paths

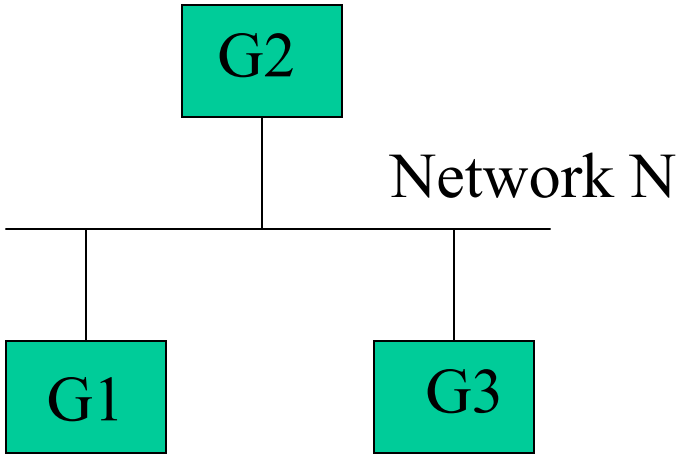
- Interior gateway protocol
- Option 1 (routing information protocol (RIP)):
  - vector distance protocol: each gateway propagates a list of the networks it can reach and the distance to each network
  - gateways use the list to compute new routes, then propagate their list of reachable networks
- Option 2 (open shortest path first (OSPF)):
  - link-state protocol: each gateway propagates status of its individual connections to networks
  - protocol delivers each link state message to all other participating gateways
  - if new link state information arrives, then gateway recomputes next-hop along shortest path to each destination

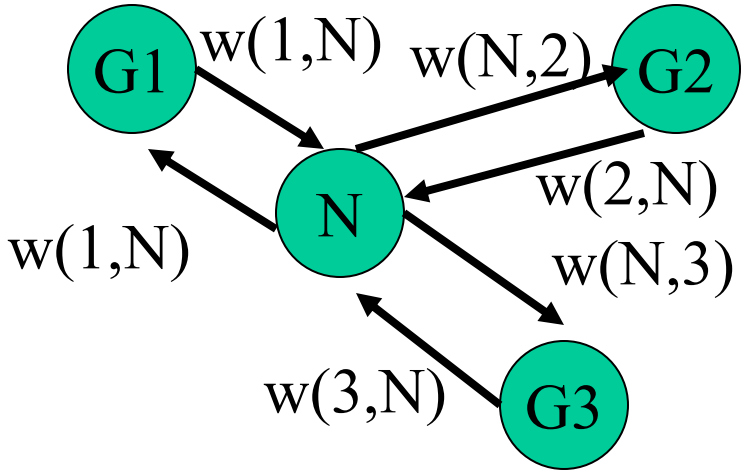
OSPF

- OSPF has each gateway maintain a topology graph
- Each node is either a gateway or a network
- If a physical connection exists between two objects in an internet, the OSPF graph contains a pair of directed edges between the nodes representing the objects
- Note: gateways engage in active propagation of routing information while hosts acquire routing information passively and never propagate it

OSPF

- Weights can be asymmetric: w(i,j) need not be equal to w(j,i)
- All weights are positive
- Weights are assigned by the network manager





Shortest Path Algorithms

- Shortest path between two nodes: length = weight
- Directed graphs (digraphs) (recall that MSTs were on undirected graphs), edges are called arcs and have a direction $(i,j) \neq (j,i)$
- Shortest path problem: a directed path from A to B is a sequence of distinct nodes A, n_1, n_2, …, n_k, B, where (A, n_1), (n_1, n_2), …, (n_k, B) are directed arcs - find the shortest such path
- Variants of the problem: find shortest path from an origin to all nodes or from all nodes to an origin
- Assumption: all cycles have non-negative length
- Three main algorithms:
  - Dijsktra
  - Bellman-Ford
  - Floyd-Warshall

Bellman-Ford

- Allows negative lengths, but not negative cycles
- B-F works at looking at negative lengths from every node to node 1
- If arc (i,j) does not exist, we set d(i,j) to
- We look at walks: consider the shortest walk from node i to 1 after at most h arcs
- Algorithm:
  - $D^{h+1}(i) = \min_{\text{over all } j}[d(i,j) + D^h(i)]$ for all i other than 1
  - we terminate when $D^{h+1}(i) = D^h(i)$
- The $D^{h+1}(i)$ are the lengths of the shortest path from i to 1 with no more than h arcs in it

Bellman-Ford

- Let us show this by induction
  - $D^1(i) = d(i,1)$ for every i other than 1, since one hop corresponds to having a single arc
  - now suppose this holds for some h, let us show it for h+1: we assume that for all $k \leq h$, $D^k(i)$ is the length of the shortest walk from i to 1 with k arcs or fewer
  - $\min_{\text{over all } j}[d(i,j) + D^h(i)]$ allows up to h+1 arcs, but $D^h(i)$ would have fewer than h arcs, so $\min[D^h(i), \min_{\text{over all } j}[d(i,j) + D^h(i)]] = D^{h+1}(i)$
- Time complexity: A, where A is the number of arcs, for at most N-1 nodes (note: A can be up to $(N-1)^2$)
- In practice, B-F still often performs better than Dijkstra ($O(N^2)$)

Distributed Asynchronous B-F

- The algorithms we investigated work well when we have a single centralized entity doing all the computation - what happens when we have a network that is operating in a distributed and asynchronous fashion?
- Let us call N(i) the set of nodes that are neighbors of node i
- At every time t, every node i other than 1 has available :
  - $D^i_j(t)$: estimate of shortest distance of each neighbor node j in N(i) which was last communicated to node i
  - $D^i(t)$: estimate of the shortest distance of node i which was last computed at node i using B-F

Distributed Asynchronous B-F

- $D^1(t) = 0$ at all times
- Each node i has available link lengths d(i,j) for all j in N(i)
- Distance estimates change only at time $t_0, t_1, .., t_m$, where t_m becomes infinitely large at m becomes infinitely large
- At these times:
  - $D^i(t) = \min_{j \text{ in } N(i)}[d(i,j) + D^i_j(t)]$, but leaves estimate $D^i_j(t)$ for all j in N(i) unchanged

OR

  - node i receives from one or more neighbors their D^j, which becomes D^i_j (all other D^i_j are unchanged)

OR

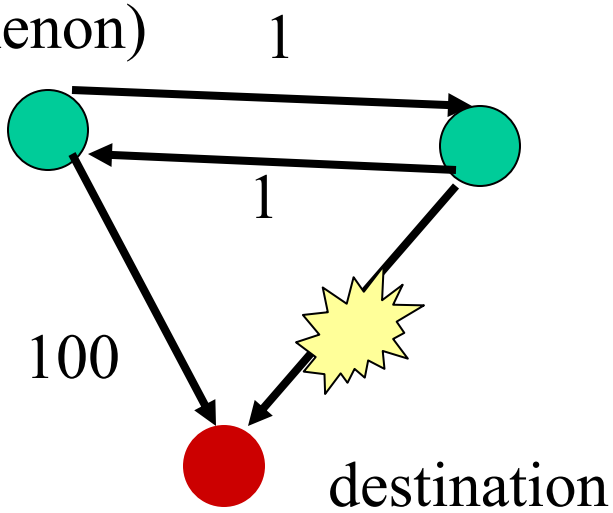
  - node i is idle

Distributed Asynchronous B-F

- Assumptions:
  - if there is a link (i,j), there is also a link (j,i)
  - no negative length cycles
  - nodes never stop updating estimates and receiving updated estimates
  - old distance information is eventually purged
  - distances are fixed
- Under those conditions:for any initial $D^i_j(t0)$, $D^i(t)$, for some t_m, eventually all values $D^i(t) = D^i$ for all t greater than t_m

Failure recovery

- Often asynchronous distributed Bellman-Ford works even when there are changes, including failures
- However, the algorithm may take a long time to recover from a failure that is located on a shortest path, particularly if the alternate path is much longer than the original path (bad news phenomenon)



Rerouting

- We have considered how to route when we have a static network, but we must also consider how to react when we have changes, in particular when we need to avoid a location because of failures or because of congestion

- Preplanned:
  - fast (ms to ns)
  - typically a large portion of the whole network is involved in re-routing
  - traditionally combines self-healing rings (SHRs) and diversity protection (DP) => constrains topology
  - hard-wired
  - all excess capacity is preplanned

- Dynamic:
  - slow (s to mn)
  - typically localized and distributed
  - well-suited to mesh networks => more flexibility in topology
  - software approach
  - uses real-time availability of spare capacity

Example of rerouting in the IP world

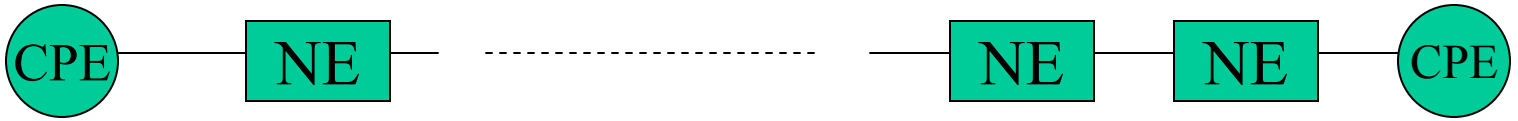
- Internet control message protocol (ICMP)
- Gateway generates ICMP error message, for instance for congestion
- ICMP redirect: “ipdirect” specifies a pointer to a buffer in which there is a packet, an interface number, pointer to a new route
- How do we get new route?
  - First: check the interface is other than the one over which the packet arrives
  - Second: run “rtget” (route get) to compute route to machine that sent datagram, returns a pointer to a structure describing the route
- If the failure or congestion is temporary, we may use flow control instead of a new route

Rerouting for ATM

- ATM is part datagram, part circuit oriented, so recovery methods span many different types
- Dynamic methods release connections and then seek ways of re-establishing them: not necessarily per VP or VC approach
  - private network to network interface (PNNI) crankback
  - distributed restoration algorithms (DRAs)
- Circuit-oriented methods often have preplanned component and work on a per VC, VP basis
  - dedicated shared VPs, VCs or soft VPs, VCs

PNNI self-healing

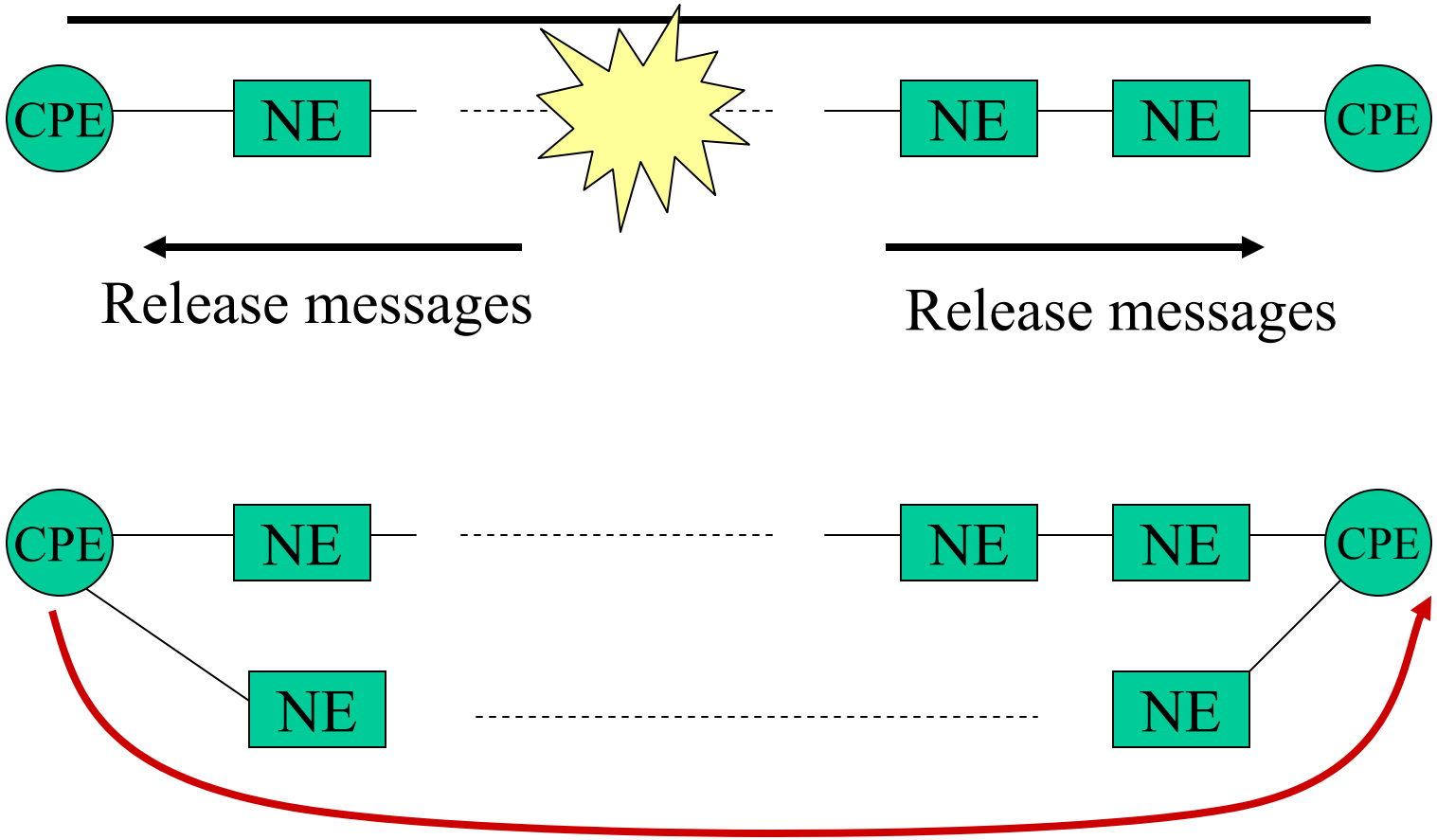
- PNNI is how ATM switches talk to each other
- Around failure or congestion area, initiate crankback
- End equipment (CPE: customer premise equipment) initiates a new connection
- In phase 2 PNNI, automatic call rerouting, freeing up CPEs from having to instigate new calls, the ATM setup message includes a request for a fault-tolerant connection



Network element

Before failure

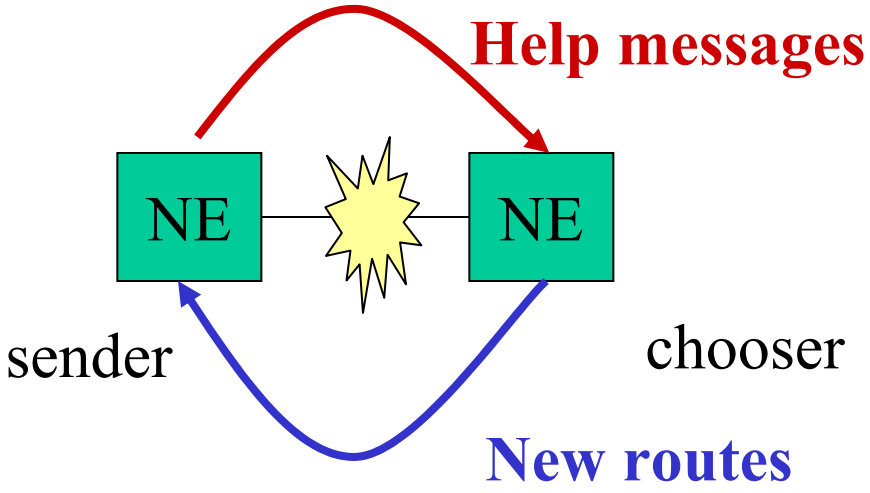
Connection re-establishment

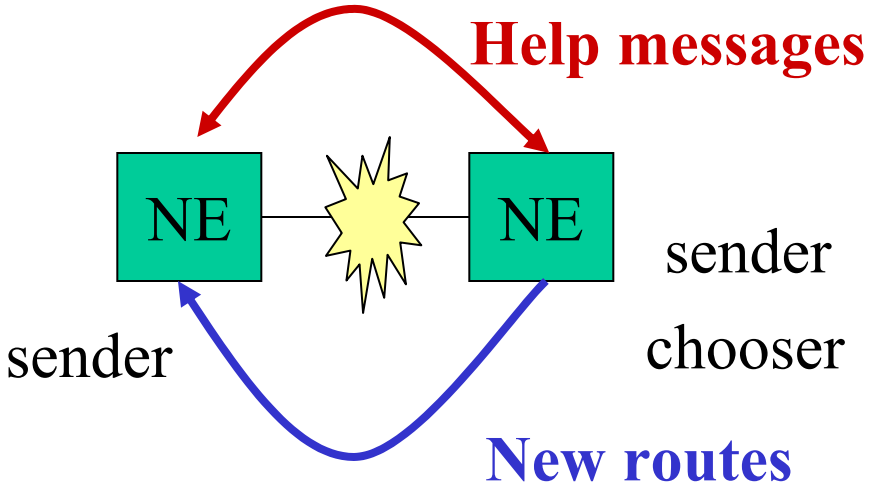


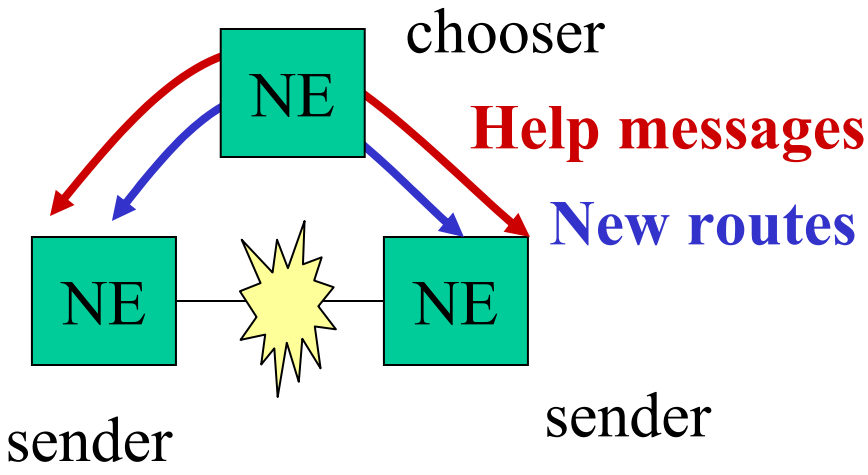
New connection is established

Issue: the congestion may cascade, giving unstable conditions, which cause an ATM storm

DRAs







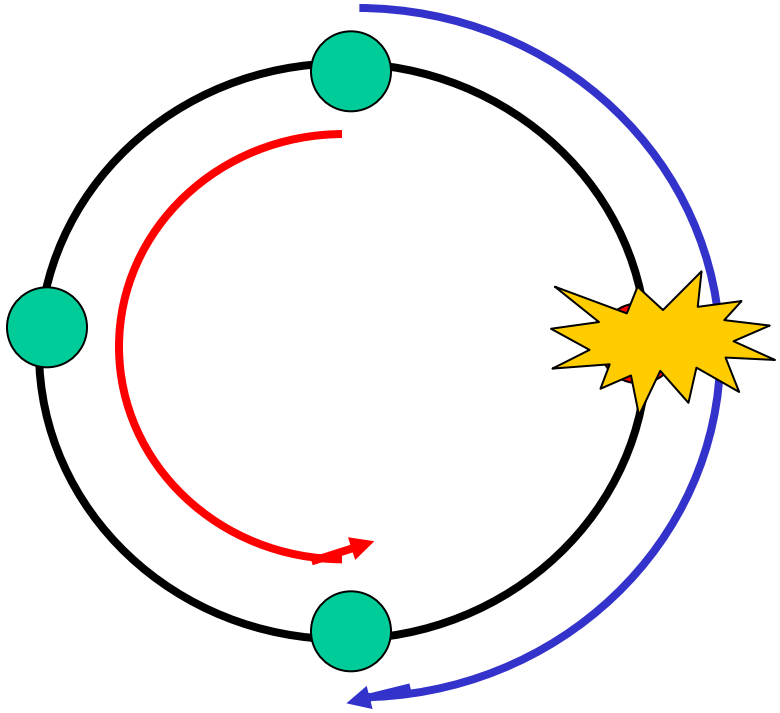
The DRAs have at least one end node transmit help messages to some nodes around them, usually within a certain hop radius, and new routes, possible splitting flows, are selected and used

Circuit-oriented methods

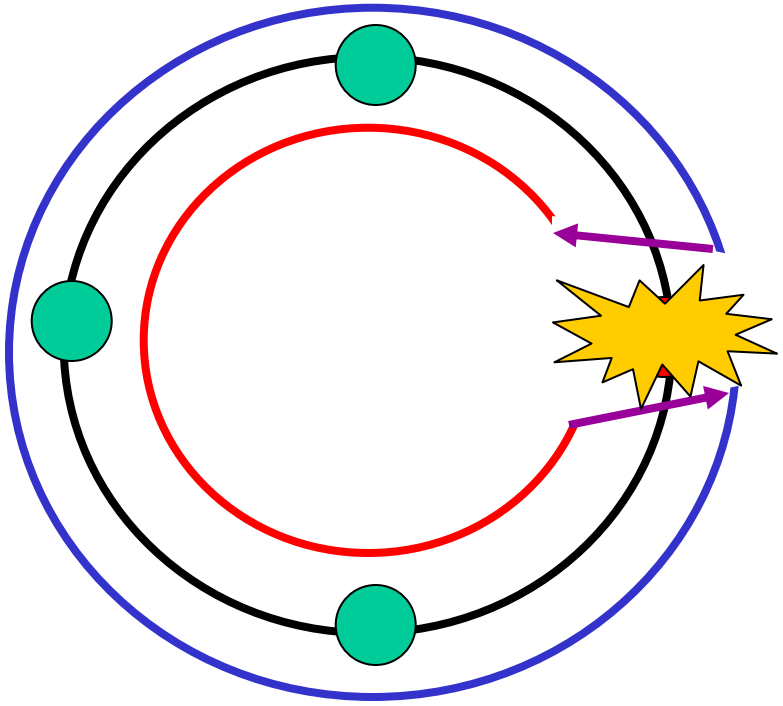
- Circuit-oriented methods seek to replace a route with another one, whether end-to-end or over some portion that is affected by a failure
- Several issues arise:
  - How do we perform recovery in a bandwidth-efficient manner
  - How does recovery interface with network management
  - What sort of granularity do we need
  - What happens when a node rather than a link fails

Rings: Path and Link/Node Rerouting

UPSR: automatic path switching on Unidirectional Path Switched Ring

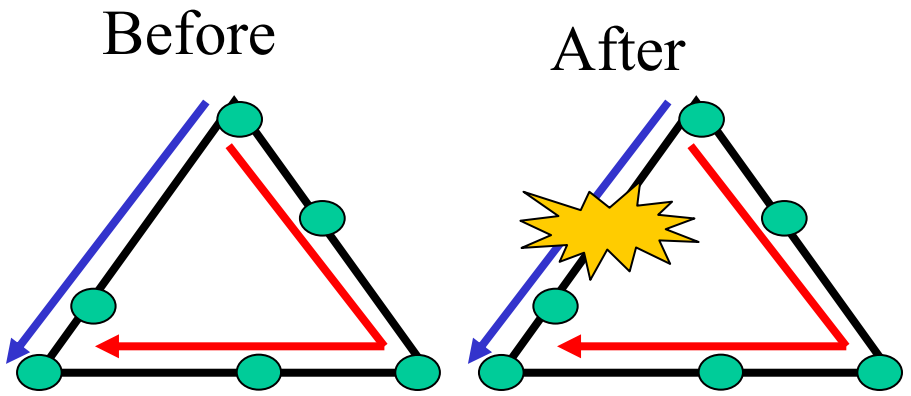


BLSR: link/node rerouting on Bidirectional Line Switched Ring

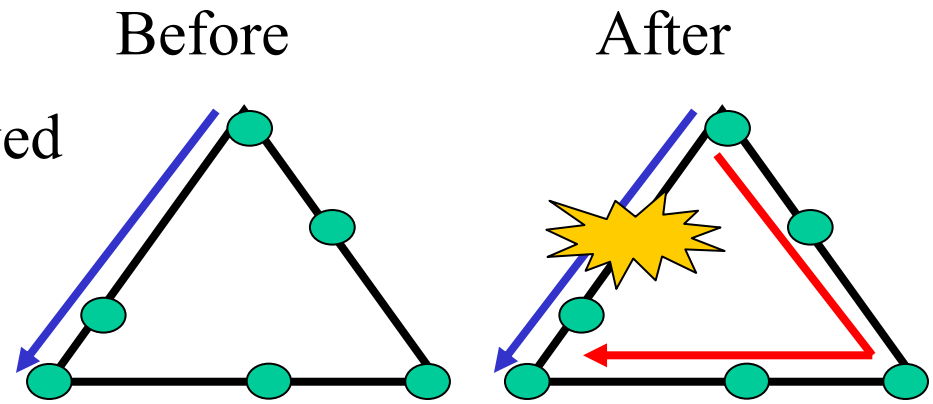


Path-based methods

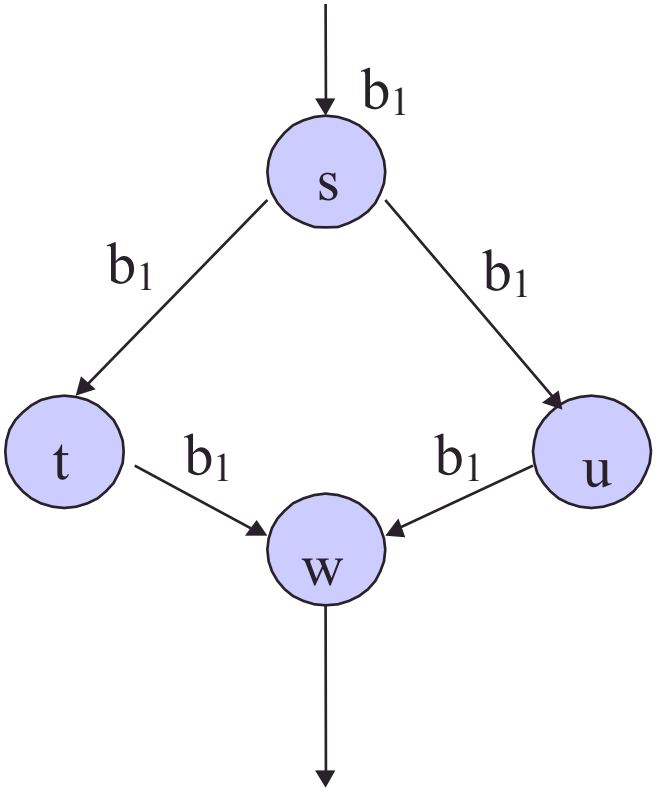
- Live back-up
  - backup bandwidth is dedicated
  - only receiver is involved
  - fast but bandwidth inefficient



- Failure triggered back-up
  - backup bandwidth is shared
  - sender and receiver are involved
  - slow but bandwidth efficient

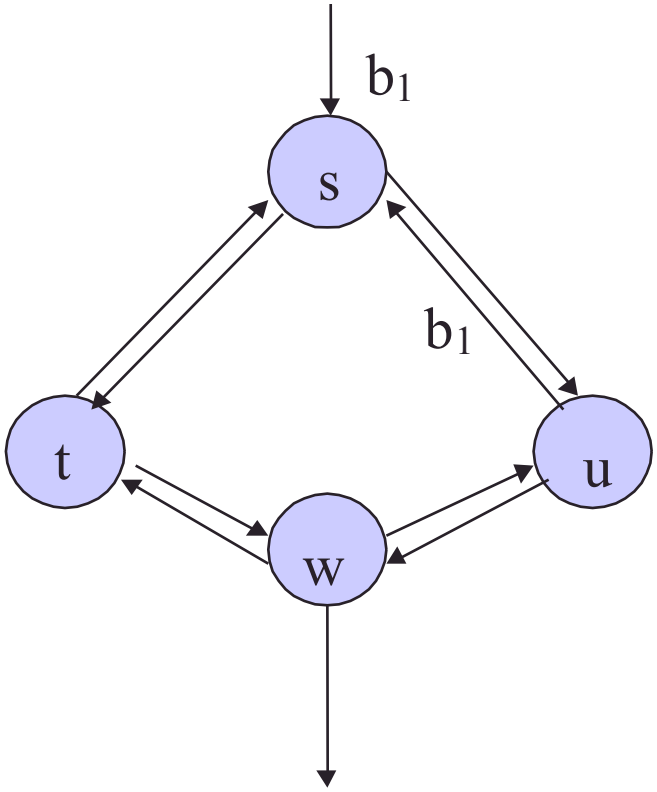


Rerouting as a code



$\overline{s} \cdot d_{t,w} + s \cdot d_{u,w} = b_1$

a. Live path protection

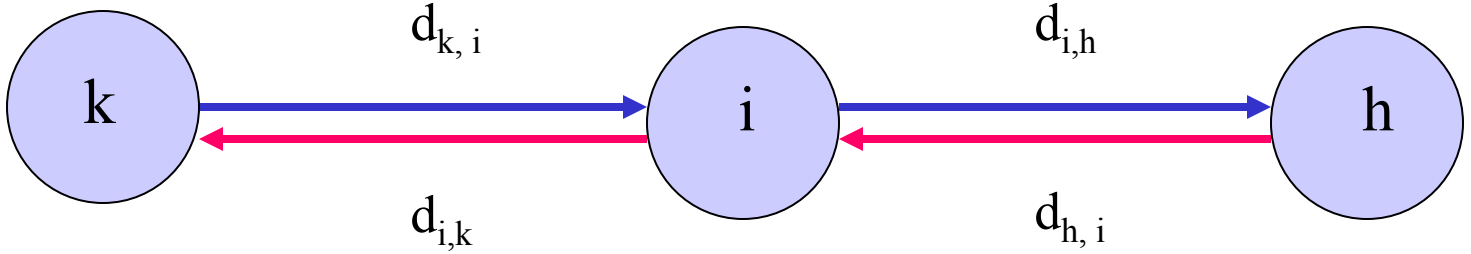


$d_{t,w} + d_{u,w} = b_1$

b. Link recovery

Rerouting as a code

- Live path protection: we have an extra supervisory signal s =1 when the primary path is live, s =0 otherwise
- Failure-triggered path protection: the backup signal is multiplied by $\bar{s}$
- Link recovery:
  - $d_{i,h} = d_{k,i} + d_{h,i}$ for the primary link (i, h) emanating from i, where (k, i) is the primary link into i and (h, i) is the secondary link into i
  - for secondary link emanating from i, the code is $d_{i,k} = d_{i,h} \cdot s_{i,h} + d_{i,h}$

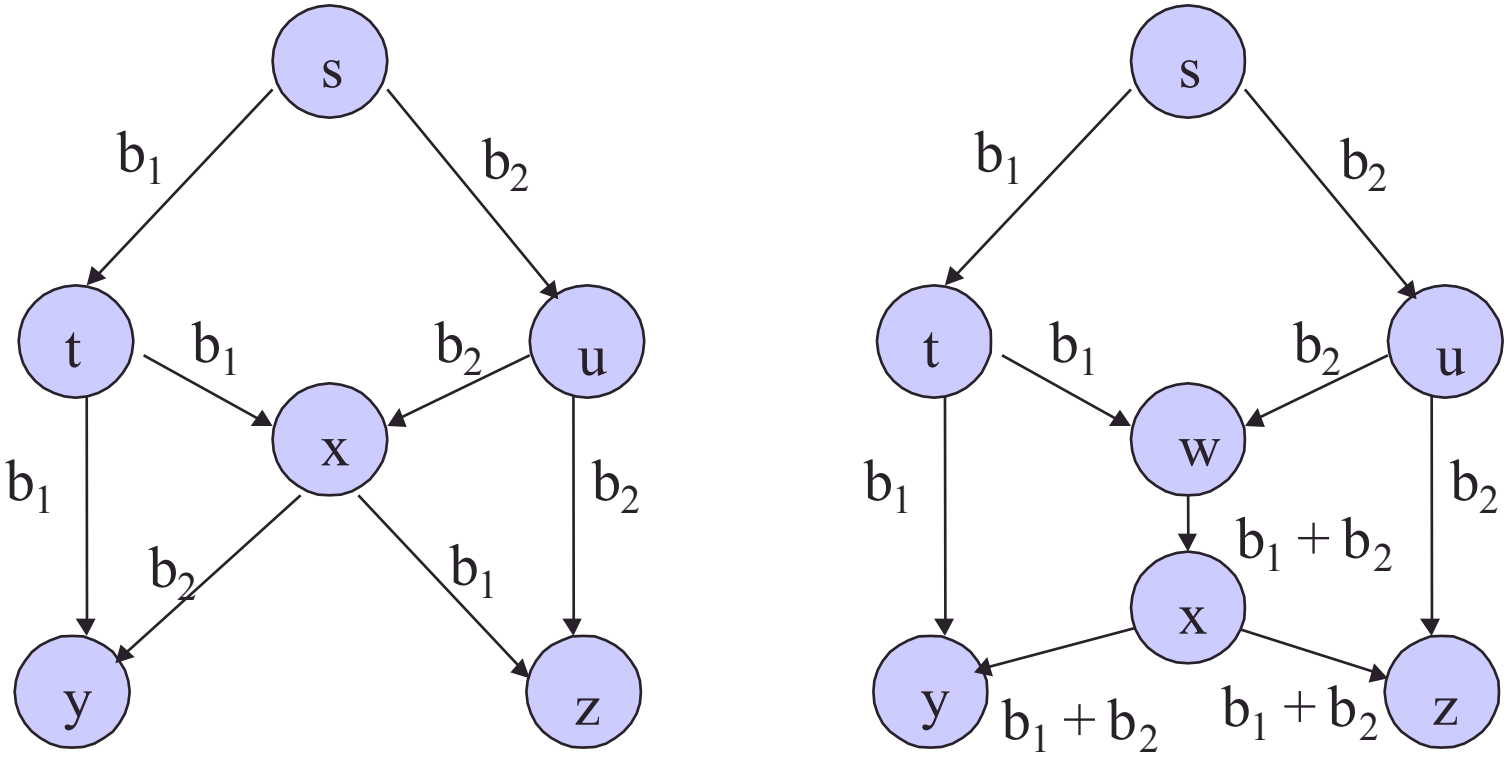


Codes and routes

- In effect, every routing and rerouting scheme can be mapped to some type of code, which may involve the presence of a network management component
- Thus, removing the restrictions of routing can only improve performance - can we actively make use of this generality?

Network coding

- The canonical example



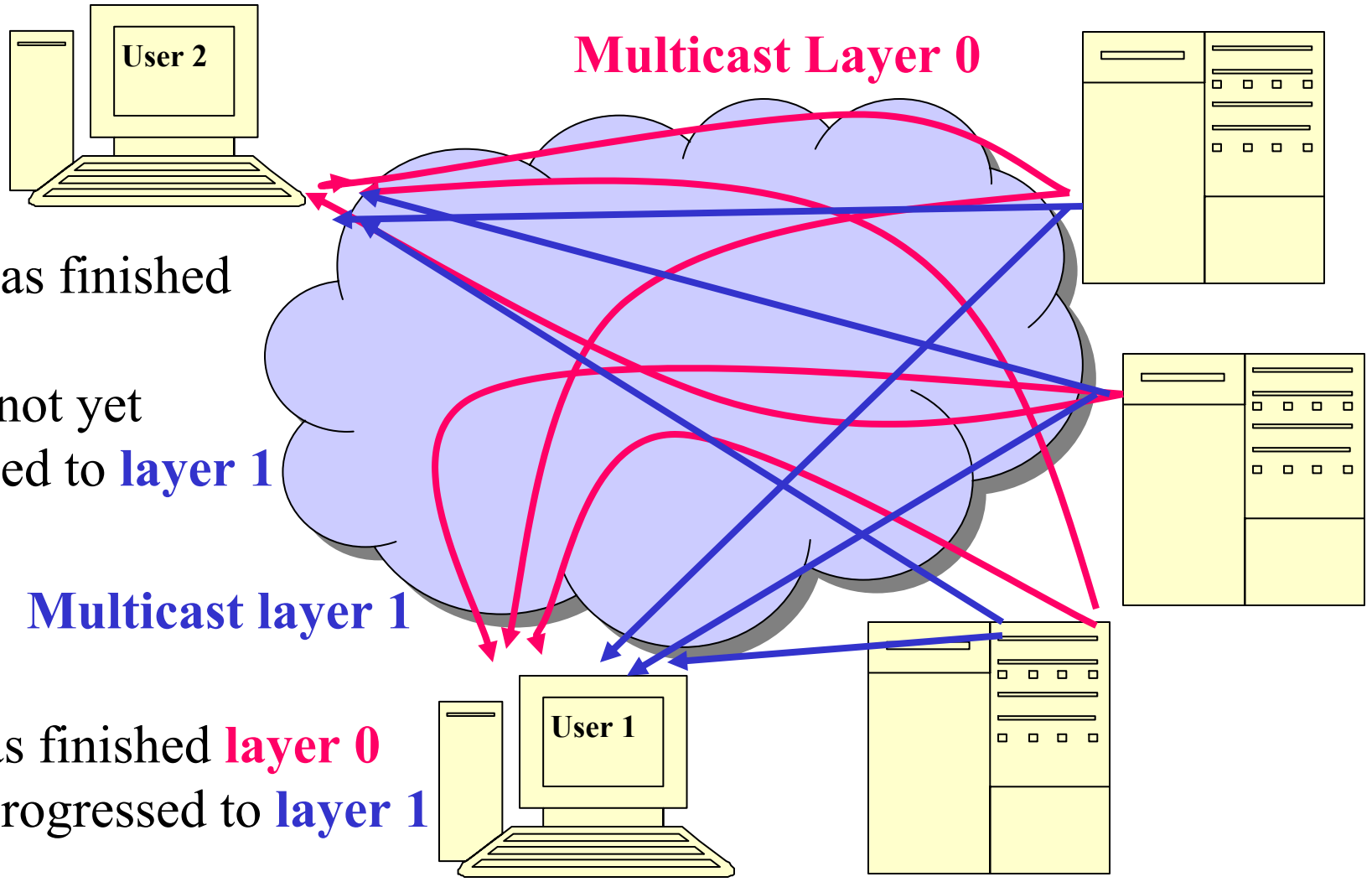
Coding across the network - have I seen this before?

- Several source-based systems exist or have been proposed
- Routing diversity to average out the loss of packets over the network
- Access several mirror sites rather than single one
- The data is then coded across packets in order to withstand the loss of packets without incurring the loss of all packets
- Rather than select the “best” route, routes are diverse enough that congestion in one location will not bring down a whole stream
- This may be done with traditional Reed-Solomon erasure codes or with Tornado codes

The Digital Fountain approach

- Idea: have users tune in whenever they want, and receive data according to the bandwidth that is available at their location in the network – “fountain” because the data stream is always on
- Create multicast layers: each layer has twice the bandwidth of the lower layer (think of progressively better resolution on images, for instance), except for the first two layers
- If receiver stays at same layer throughout, and packet loss rate is low enough, then receiver can reconstruct source data before receiving any duplicate packets : "One-level property"
- Receivers can only subscribe to higher layer after seeing a*synchronization point* (SP) in their own layer
- The frequency of SPs is inversely proportional to layer bandwidth

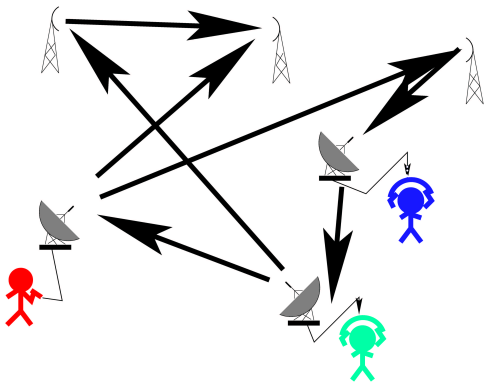
Digital fountain



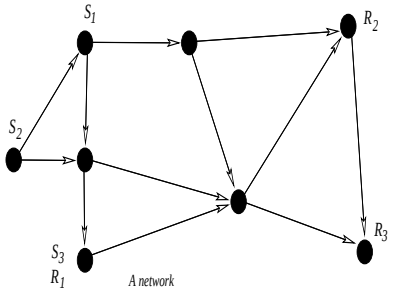
Network coding vs. Coding for networks

- The source-based approaches consider the networks as in effect channels with ergodic erasures or errors, and code over them, attempting to reduce excessive redundancy
- The data is expanded, not combined to adapt to topology and capacity
- Underlying coding for networks, traditional routing problems remain, which yield the virtual channel over which coding takes place
- Network coding subsumes all functions of routing - algebraic data manipulation and forwarding are fused

II — Algebraic Foundations of Network Coding



$\Rightarrow$

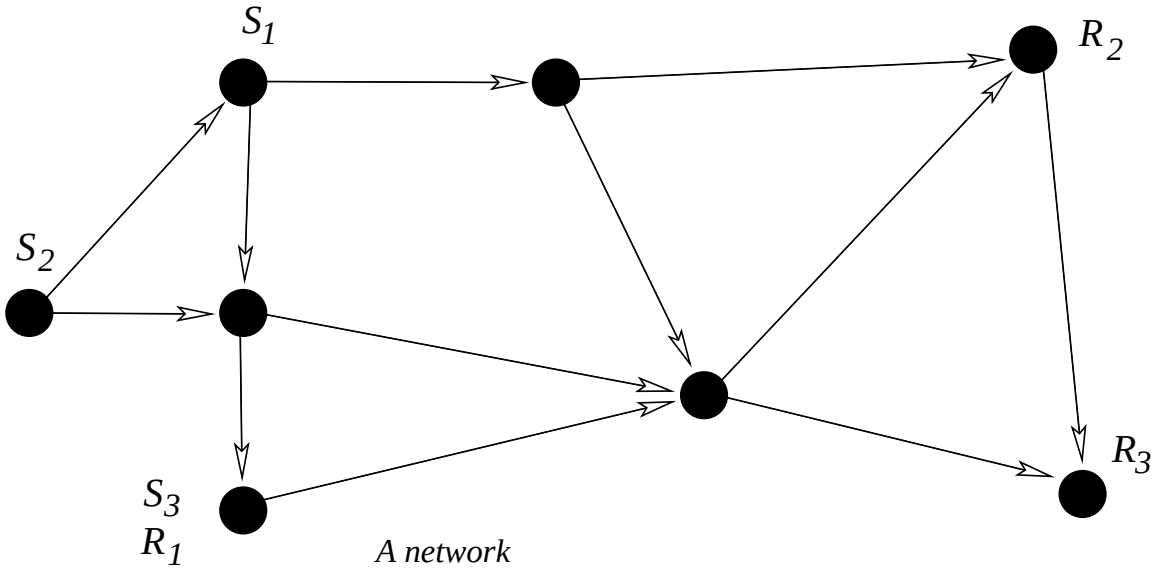


$\Rightarrow \quad A(I-F)^{-1}B^T = I$

Why an "algebraic" characterization?

- Graph-theoretic proofs are cumbersome
- Generalizations are possible
- Equations are easier managed than graphs
- Powerful tools available

Problem Description



A network

Vertices: V
Edges: $E \subseteq V \times V$, $e = (v, u) \in E$
Edge capacity: $C(e)$
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 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?

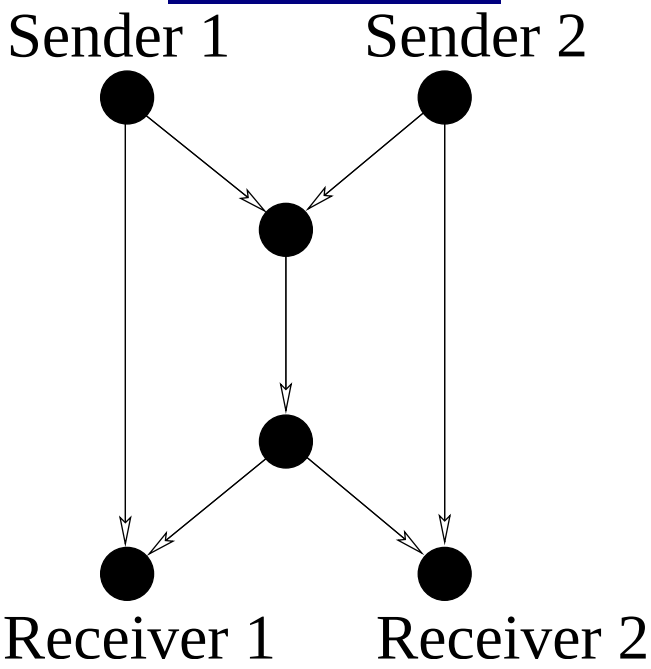
How do we find a solution?

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

How do we find a solution?

This is fairly idealized (synchronization, protocol, dynamic behaviour, error free operation,...) but gives insights into possible limits and opportunities.

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

More Simplifications — Linear Network Codes

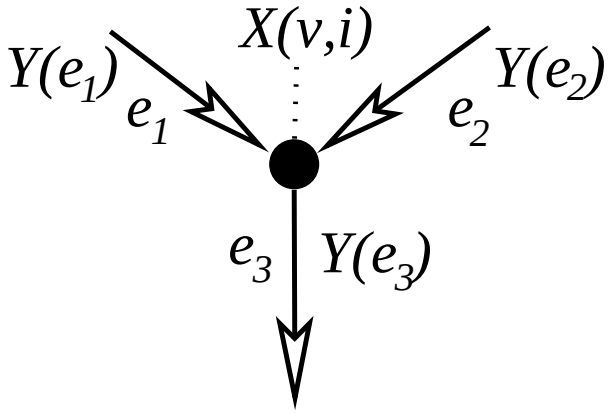
$C(e) = 1$ (links have the same capacity)
$H(X(v,i)) = 1$ (sources have the same rate)
The $X(v,i)$ are mutually independent.
Vector symbols of length m elements in $\mathbb{F}_{2^m}$.

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

This is necessary to define linear operations.

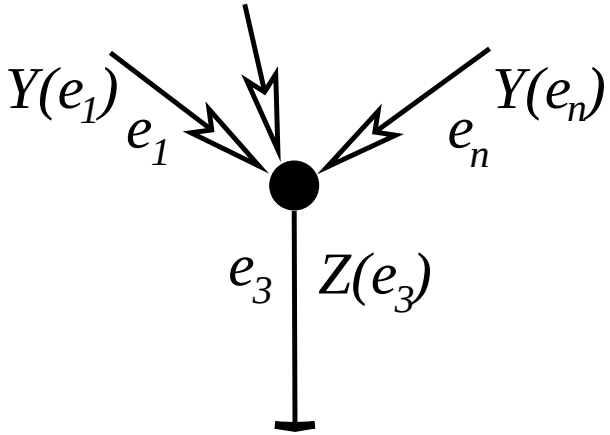
More Simplifications — Linear Network Codes

All operations at network nodes are linear!



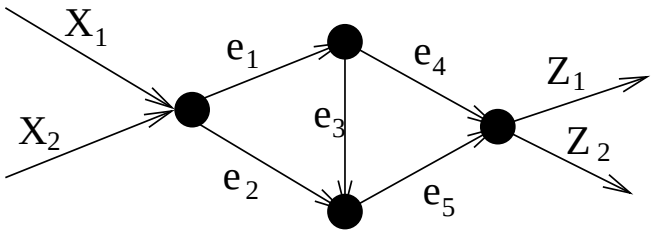
$$Y(e_3) = \sum_i \alpha_i X(v,i) + \sum_{j=1,2} \beta_j Y(e_j)$$

At a receiver (terminal) node:

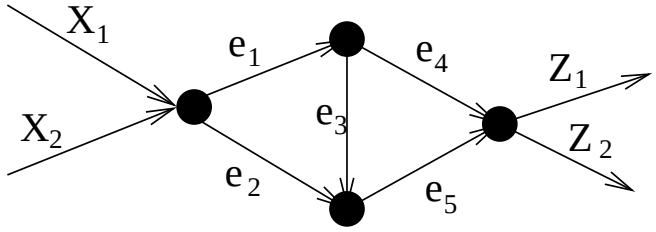


$$Z(v,j) = \sum_{j=1}^{n} \varepsilon_j Y(e_j).$$

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)

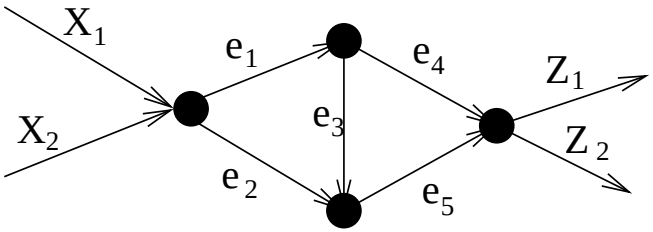
$$\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \varepsilon_{e_4,1} & \varepsilon_{e_5,1} \\ \varepsilon_{e_4,2} & \varepsilon_{e_5,2} \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} \beta_{e_1,e_4} & 0 \\ \beta_{e_1,e_3}\beta_{e_3,e_5} & \beta_{e_2,e_5} \end{pmatrix}}_{G} \underbrace{\begin{pmatrix}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

We define three matrices A, G, B

The main question becomes: Is G invertible?

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}.



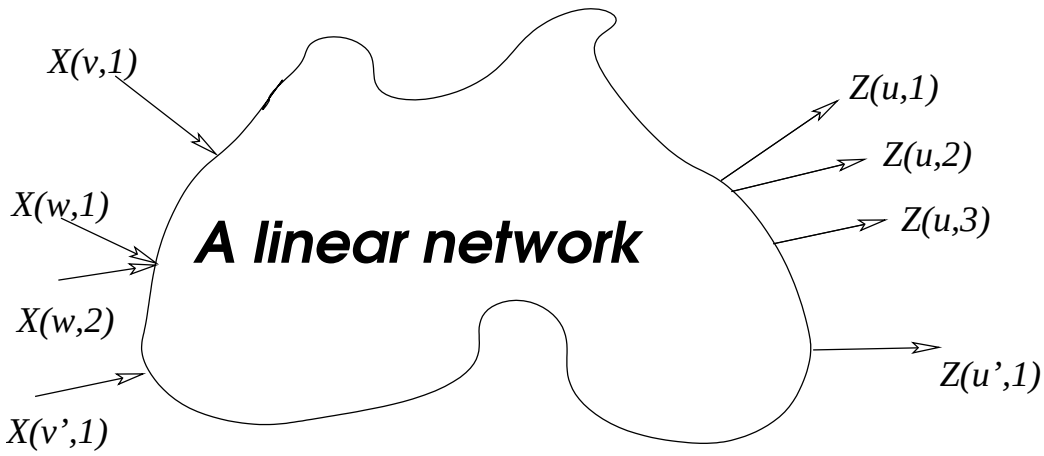
$$F = \begin{pmatrix} 0 & 0 & \beta_{e_1,e_3} & \beta_{e_1,e_4} & 0 \\ 0 & 0 & 0 & 0 & \beta_{e_2,e_5} \\ 0 & 0 & 0 & 0 & \beta_{e_3,e_5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

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

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

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

$$\underline{z} = M\underline{x} = B \cdot \underbrace{(I - F^T)^{-1}}_{G^T} \cdot A\ \underline{x}$$

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

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 $\mathbb{F}_2[\underline{\xi}]$

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

An innocent looking Lemma

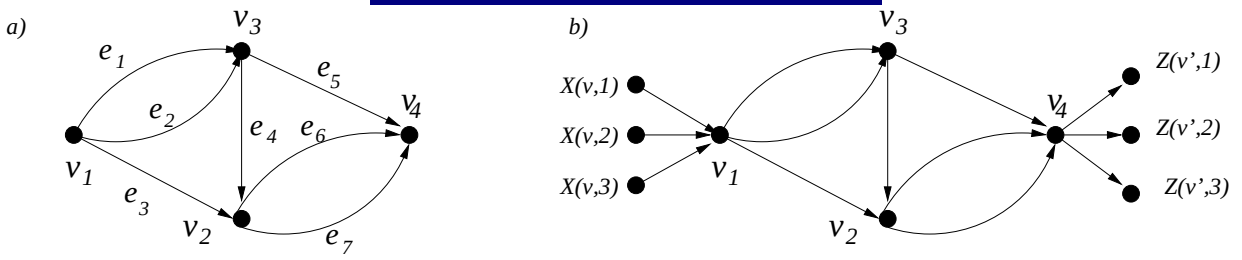
Let $\mathbb{F}[X_1, X_2, \ldots, X_n]$ be the ring of polynomials over an infinite field $\mathbb{F}$ in variables $X_1, X_2, \ldots, X_n$. For any non-zero element $f \in \mathbb{F}[X_1, X_2, \ldots, X_n]$ there exists an infinite set of n-tuples $(x_1, x_2, \ldots, x_n) \in \mathbb{F}^n$ such that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

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 $2^6 - 2^4 - 2^2 + 2 = 46 \equiv 1 (mod\ 5)$.

Another Example:



$$\mathscr{C} = (v_1, v_4, \{X(v_1,1), X(v,2), X(v_1,3)\})$$

$$A = \begin{pmatrix} \alpha_{e_1,1} & \alpha_{e_2,1} & \alpha_{e_3,1} \\ \alpha_{e_1,2} & \alpha_{e_2,2} & \alpha_{e_3,2} \\ \alpha_{e_1,3} & \alpha_{e_2,3} & \alpha_{e_3,3} \end{pmatrix}, \quad B = \begin{pmatrix} \varepsilon_{e_5,1} & \varepsilon_{e_5,2} & \varepsilon_{e_5,3} \\ \varepsilon_{e_6,1} & \varepsilon_{e_6,2} & \varepsilon_{e_6,3} \\ \varepsilon_{e_7,1} & \varepsilon_{e_7,2} & \varepsilon_{e_7,3} \end{pmatrix}.$$

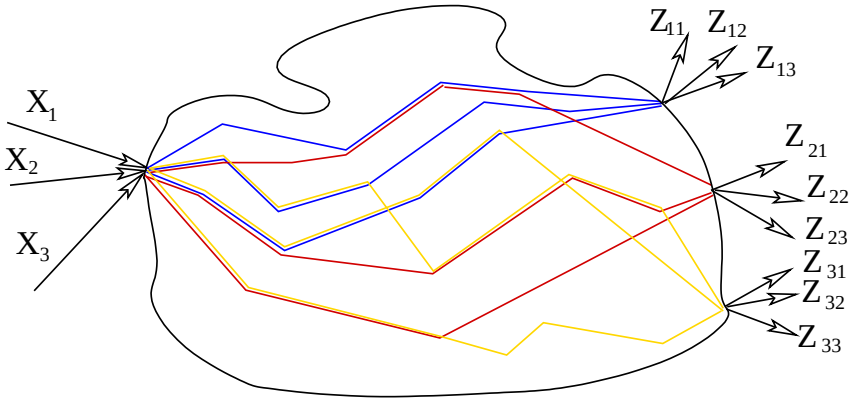
$$M = A \begin{pmatrix} \beta_{e_1,e_5} & \beta_{e_1,e_4}\beta_{e_4,e_6} & \beta_{e_1,e_4}\beta_{e_4,e_7} \\ \beta_{e_2,e_5} & \beta_{e_2,e_4}\beta_{e_4,e_6} & \beta_{e_2,e_4}\beta_{e_4,e_7} \\ 0 & \beta_{e_3,e_6} & \beta_{e_3,e_6} \end{pmatrix} B^T.$$

$$\begin{aligned} \det(M) &= \det(A)\det(B) \\ &\quad (\beta_{e_1,e_5}\beta_{e_2,e_4} - \beta_{e_2,e_5}\beta_{e_1,e_5})(\beta_{e_4,e_6}\beta_{e_3,e_7} - \beta_{e_4,e_7}\beta_{e_3,e_6}) \end{aligned}$$

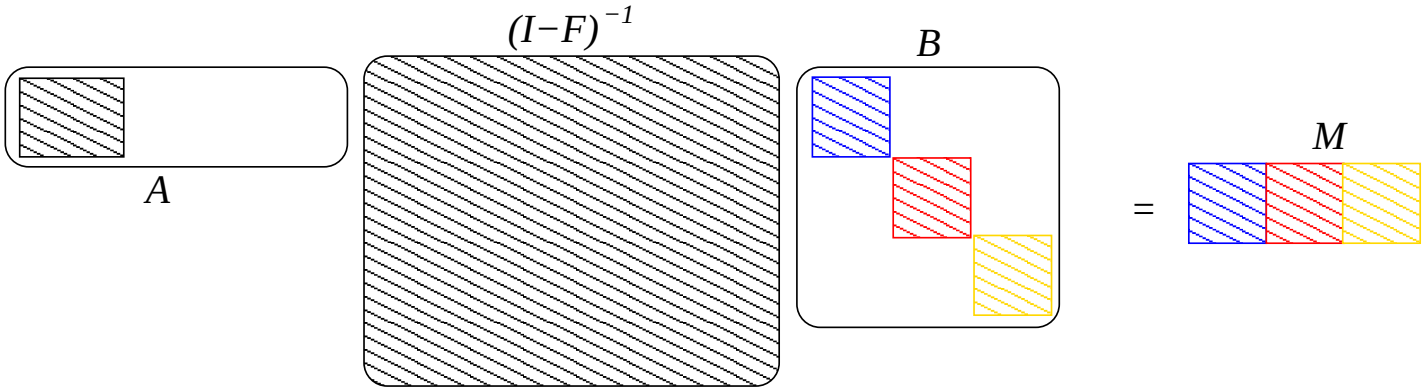
Choose the coefficients so that $\det(M) \neq 0$!

Multicast:

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

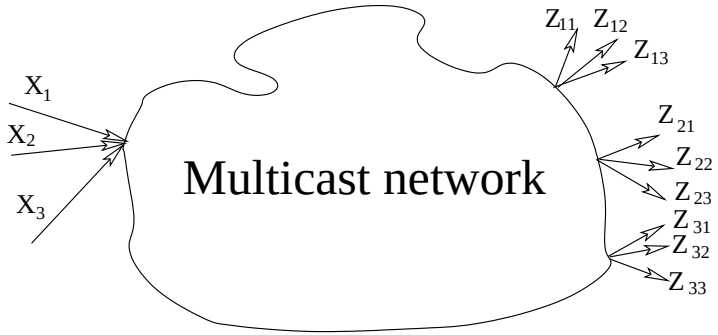


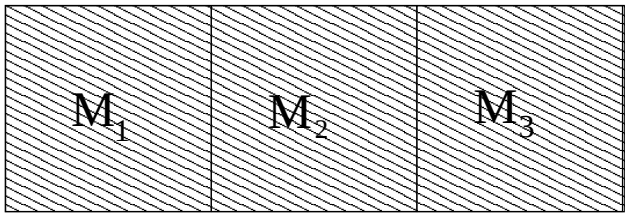
Multicast network



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

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 $\bar{\mathbb{F}}$ so that all $m_i(\underline{\xi})$ are unequal to zero.

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

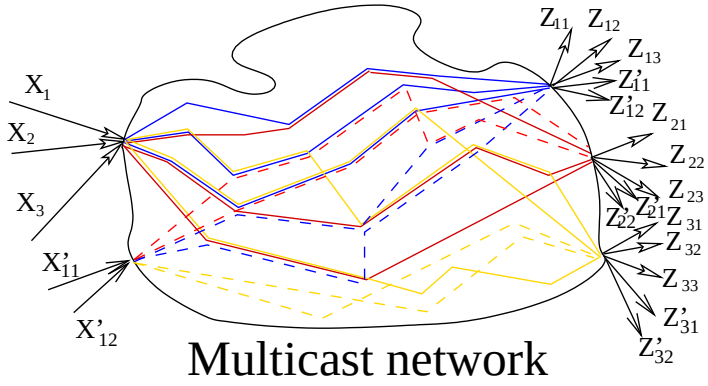
The main Multicast Theorem:

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.

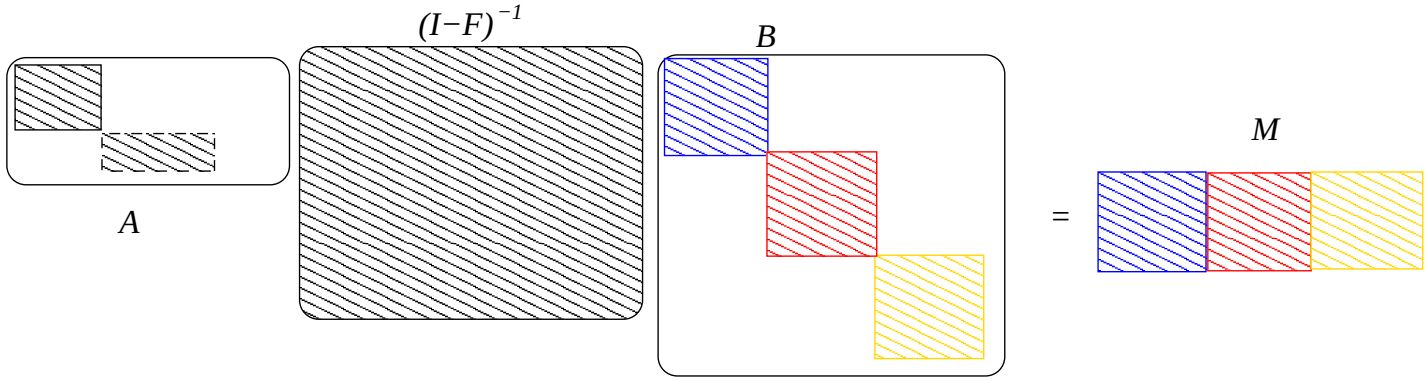
(We will see later how large m will have to be — it's not too bad)

Other (derived) problems: Multisource — Multicast

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



Multicast network

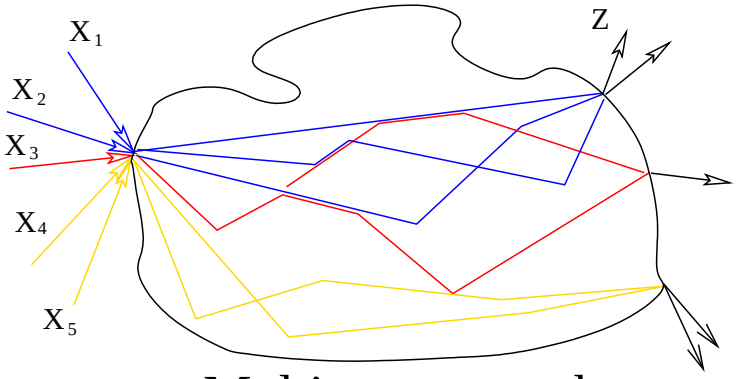


Other (derived) problems: Multisource — Multicast

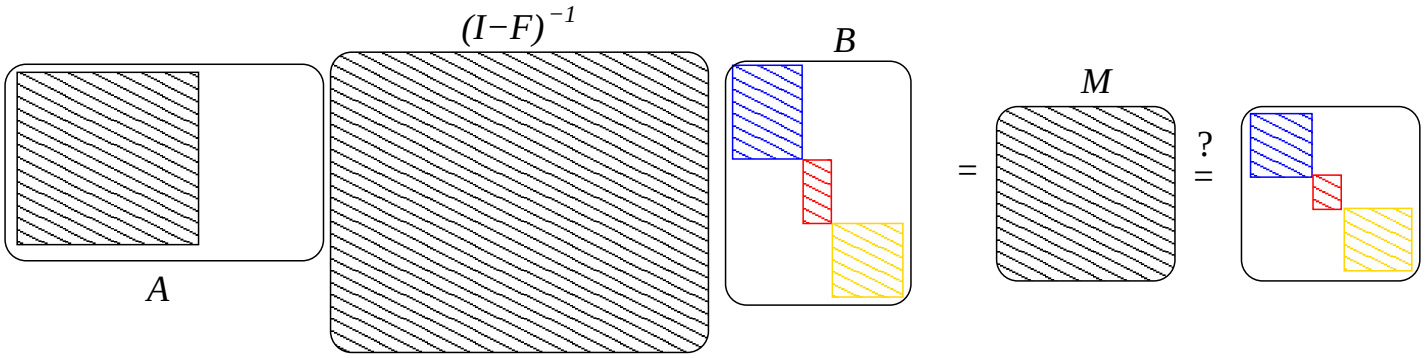
Theorem Let a linear, acyclic, delay-free network $\mathcal{G}$ be given with a set of desired connections $\mathscr{C} = \{(v_i, u_j, \mathscr{X}(v_i)) : i = 0, 1, \ldots N, j = 1, 2, \ldots K\}$. The network problem $(\mathcal{G}, \mathscr{C})$ is solvable if and only if the Min-Cut Max-Flow bound is satisfied for any cut between all source nodes $\{v_i : i = 0, 1, \ldots N\}$ and any sink node u_j.

Other (derived) problems: One source — Disjoint Muticasts

$\mathscr{C} = \{(v, u_j, \mathscr{X}(v, u_j)) : j = 1, 2, \ldots K\}$, $\mathscr{X}(v, u_j) \cap \mathscr{X}(v, u_i) = \emptyset$

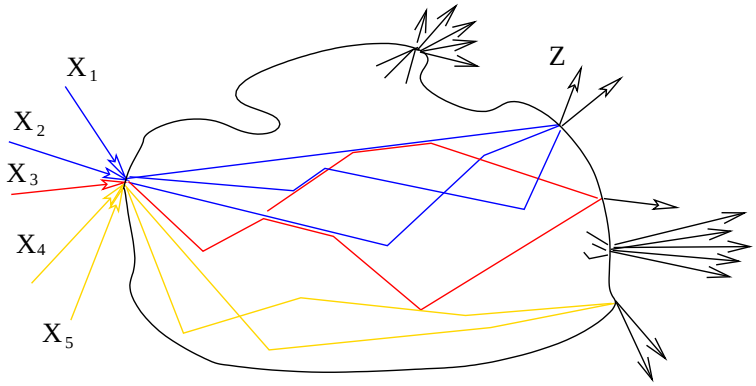


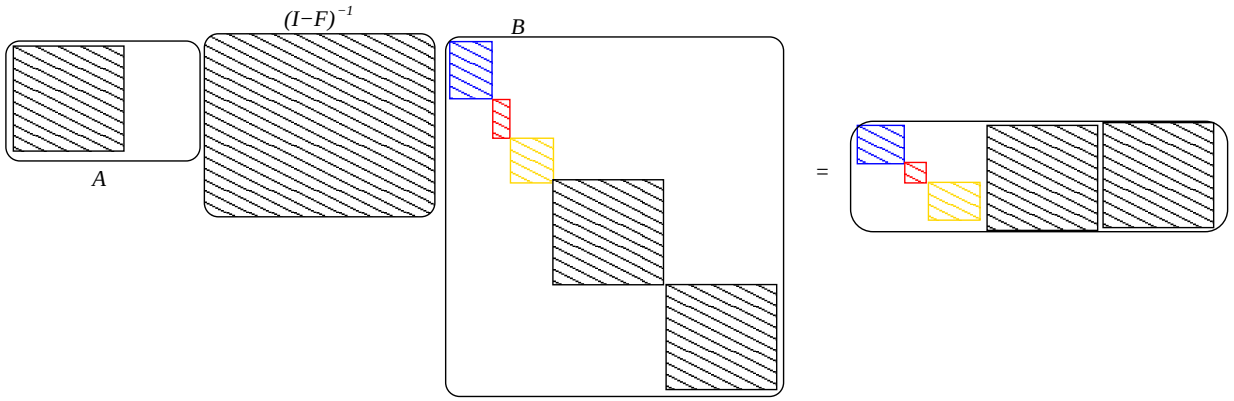
Multicast network



One source — Disjoint Muticasts + Multicasts

$\mathscr{C} = \{(v, u_j, \mathscr{X}(v, u_j)) : j = 1, 2, \ldots K\} \cup \{(v, u_\ell, \mathscr{X}(v)) : j = K+1, K+2, \ldots K+N\}$, $\mathscr{X}(v, u_j) \cap \mathscr{X}(v, u_i) = \emptyset$



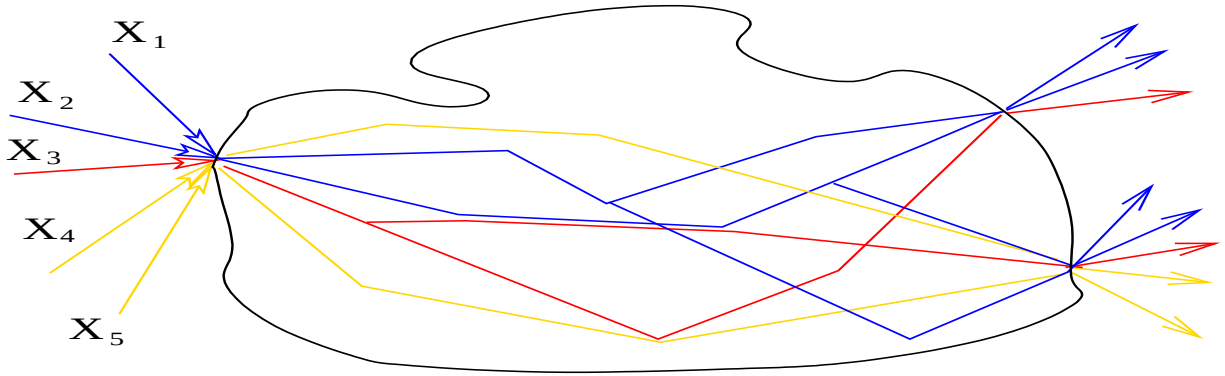


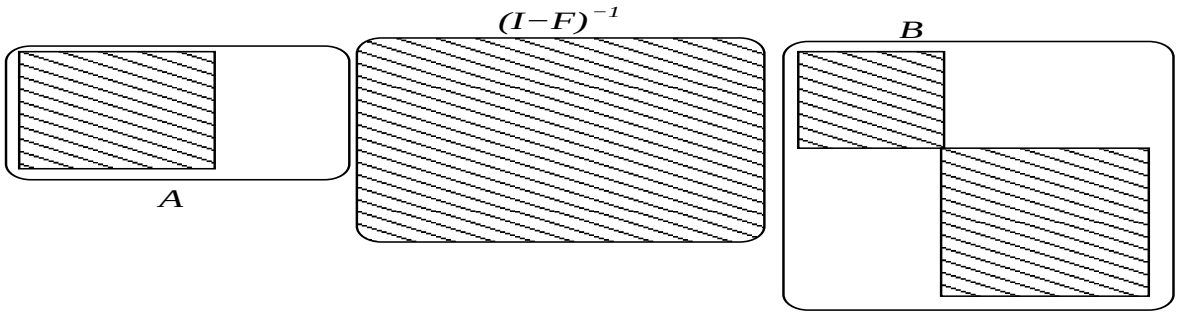
Multisource — Disjoint Muticasts + Multicast

Theorem Let a linear, acyclic, delay-free network $\mathcal{G}$ be given with a set of desired connections $\mathscr{C} = \{(v, u_j, \mathscr{X}(v, u_j)) : j = 1, 2, \ldots K\} \cup \{(v, u_\ell, \mathscr{X}(v)) : j = K+1, K+2, \ldots K+N\}$ such that collection of random processes $\mathscr{X}(v, u_j)$, $\mathscr{X}(v, u_j)$ are mutually disjoint for $i, j < K$, i.e. $\mathscr{X}(v, u_j) \cap \mathscr{X}(v, u_i) = \emptyset$ for $i \neq j$, $i, j \leq K$. The network problem is solvable if and only if the Min-Cut Max-Flow bound is satisfied between v and the set of sink nodes $\{u_1, u_2, \ldots, u_K\}$ at a rate $|\mathscr{X}(v)|$ and between v and u_ℓ, $\ell > K$ also at a rate $|\mathscr{X}(v)|$.

Other (derived) problems: Two level Multicasts

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





Other (derived) problems: Two Level Multicast

Theorem("Two-level multicast") Let an acyclic network $\mathcal{G}$ be given with a set of desired connections

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

The network problem is solvable if and only if the Min-Cut Max-Flow bound is satisfied between v and u_1 at a rate $|\mathscr{X}(v, u_1)|$ and between v and u_2 at a rate $|\mathscr{X}(v)|$.

So far so good!

What about networks with cycles?

What about networks with delays?

What about robustness?

Do we really need network coding for multicast?

So far so good!

What about networks with cycles?

What about networks with delays?

What about robustness?

Do we really need network coding for multicast? YES

Robust multicast:

Links in the network may fail. (non-ergodic). Set of failure patterns:$\mathscr{F}$

A network solution is static w.r.t. $\mathscr{F}$ if the operations in the network interior are oblivious to the particular failure in $\mathscr{F}$.

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem and let $\mathscr{F}$ be the set of failure patterns such that the problem is solvable. There exists a common static solution to all failure patterns in $\mathscr{F}$.

Proof sketch: All we have to do is to guarantee that the product of all determinants of all scenarios in $\mathscr{F}$ evaluates to a non zero value.

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem and let $\mathscr{F}$ be the set of failure patterns such that the problem is solvable.. There exists a solution for $(\mathcal{G}, \mathscr{C})$ over a finite field $\mathbb{F}_{2^m}$ with $m \leq \lceil \log_2(|\mathscr{F}|NR + 1) \rceil$.

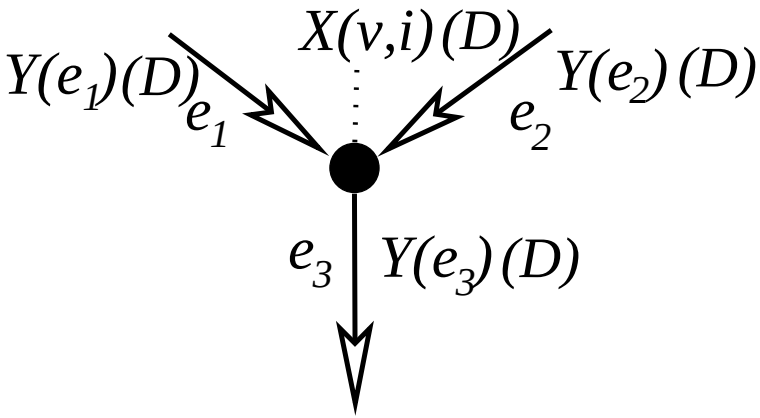
$\vdots$

Linear Networks with Delays

We transmit random processes in a delay variable D on links, i.e.

$$\begin{aligned} X(v,j)(D) &= \sum_{\ell=0}^{\infty} X_\ell(v,j) D^\ell, \\ Z(v,j)(D) &= \sum_{\ell=0}^{\infty} Z_\ell(v,j) D^\ell, \\ Y(e)(D) &= \sum_{\ell=0}^{\infty} Y_\ell(e) D^\ell. \end{aligned}$$

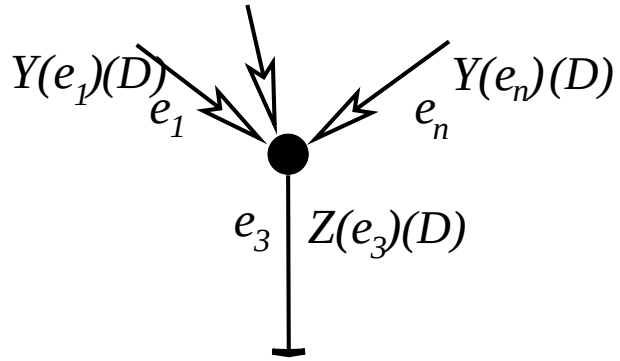
Conceptually, we consider an entire sequence in D as one symbol and work over the field of formal power series.



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

(other functions with memory are possible but not necessary)

At a receiver (terminal) node we have to allow for "rational" functions:

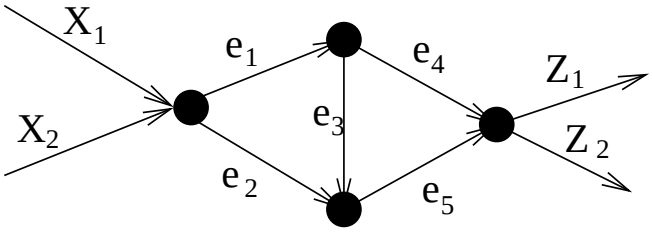


$$Y(e)(D) = \sum_{\ell=0}^{\infty} Y_\ell(e) D^\ell, \quad Z(v,j)(D) = \sum_{\ell=0}^{\infty} Z_\ell(v,j) D^\ell$$

$$Z_\ell(v,j) = \sum_{j=1}^{n} \sum_{k=0}^{\mu} \varepsilon_{j,k} Y_{\ell-k}(e_j) + \sum_{k=1}^{\mu} \lambda_k Z_{\ell-k}(v,j)$$

or $$Z(v,j)(D) = \sum_{j=1}^{n} \frac{\varepsilon_{j,k}(D)}{\lambda(D)} Y(e_j)(D)$$

The transfer matrix with delays



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

Summing the "path gains":

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

Observe that $G = (I - DF)^{-1}$ is polynomial over $\mathbb{F}_2(D)$.

An algebraic Min-Cut Max-Flow condition with delays

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 $\mathbb{F}_2(D)[\underline{\xi}]$ with coefficients from the field of rational functions.

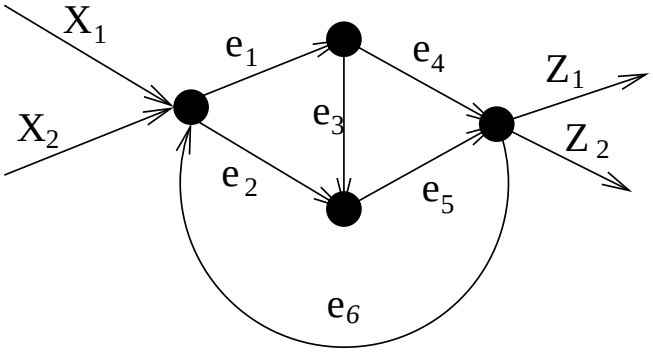
It is only that....

We have to study the solution sets of polynomial equations over $\mathbb{F}_2(D)$.

At receiver nodes we have to allow for memory and the possibility of implementing rational functions!

This is neccessary since now we have to invert a transfer matrix which has as elements polynomials over $\mathbb{F}_2(D)$.

The transfer matrix with delays and cycles



$$F = \begin{pmatrix} 0 & 0 & D\beta_{e_1,e_3} & D\beta_{e_1,e_4} & 0 & \\ 0 & 0 & 0 & 0 & D\beta_{e_2,e_5} & \\ 0 & 0 & 0 & 0 & D\beta_{e_3,e_5} & \\ 0 & 0 & 0 & 0 & 0 & D\beta \\ 0 & 0 & 0 & 0 & 0 & D\beta \\ D\beta_{e_6,e_1} & D\beta_{e_6,e_2} & 0 & 0 & 0 & \end{pmatrix}$$

Summing the "path gains":

$P = I + DF + D^2F^2 + \ldots = (I - DF)^{-1} =$ (6 × 6 matrix with rational coefficients)

Now $G = (I - DF)^{-1}$ is rational over $\mathbb{F}_2(D)$.

Delays and cycle - or really nothing has happened....

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

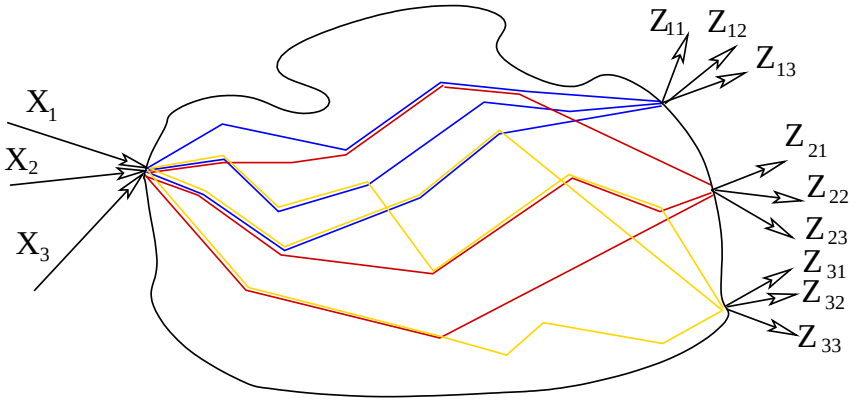
1. A point-to-point connection $c = (v, v', \mathscr{X}(v, v'))$ is possible.

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

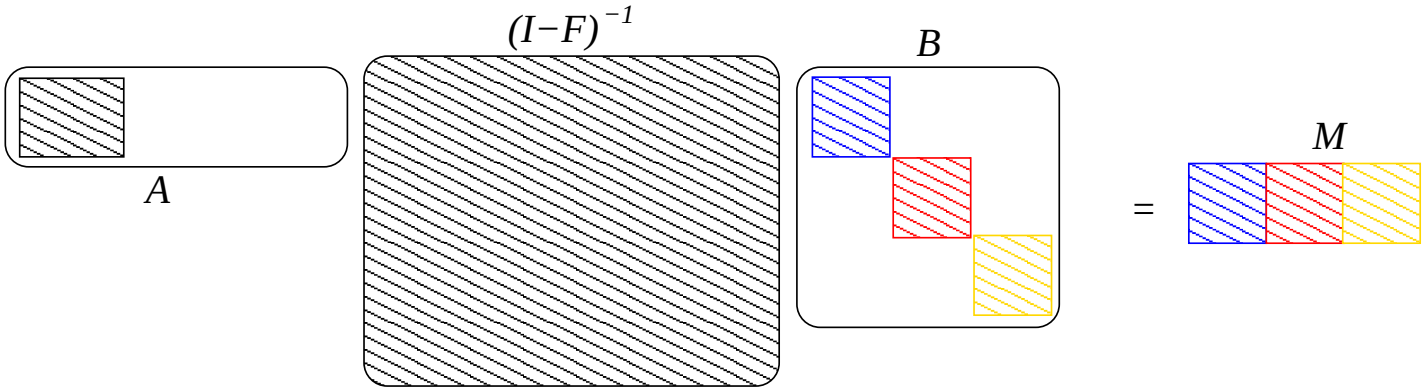
3. The determinant of the $R(c) \times R(c)$ transfer matrix M is nonzero over the ring of polynomials $\mathbb{F}_2(D)[\underline{\xi}]$ with coefficients from the field of rational functions.

Multicast:

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

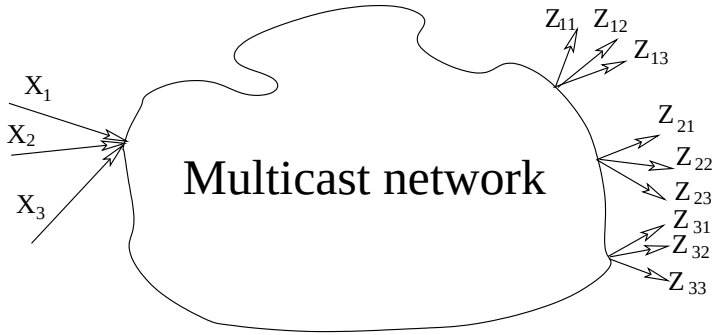


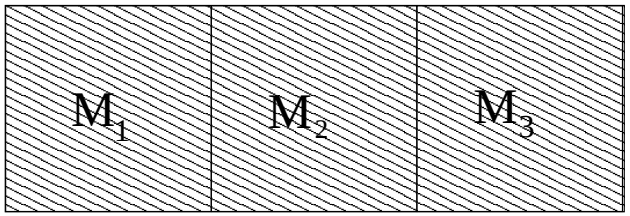
Multicast network



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

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 $\bar{\mathbb{F}}$ so that all $m_i(\underline{\xi})$ are unequal to zero.

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

The main Multicast Theorem:

Theorem Let $(\mathcal{G}, \mathscr{C})$ be a multicast network coding problem on a graph which may have a cyclic structure. 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.

Theorems, Theorems.....

Summary

- Connecting network information flow problems to algebraic equations yields powerful tools for analysis of networks.

- Multicast especially well suited for the approach since we have to find "non solutions" to equations, which can easily be accomplished in large fields.

- Many network scenarios can be derived from the multicast setup.

- The general non multicast setup will be treated later (much less is known).

- Field size?
- How do we find solutions?
- Is network coding really helpful or just a singular occurrence?