Course: Networking II

Topic: Network Coding – an introduction

Eurecom 2014-2015

L. Sassatelli

sassatelli@i3s.unice.fr

About…

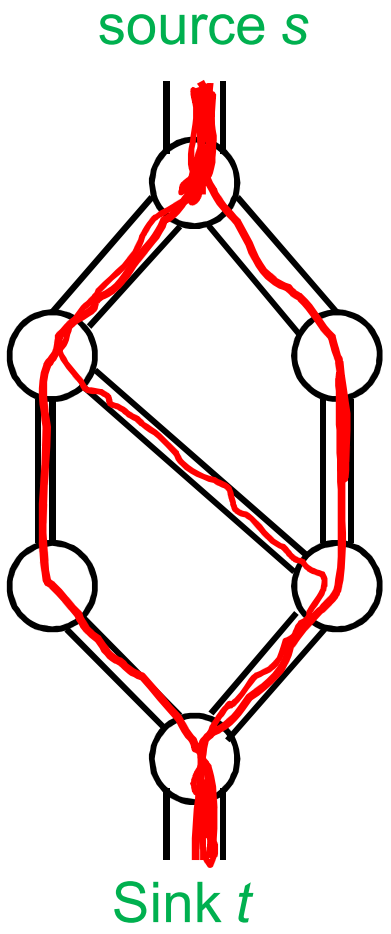
- This course is about:
  - Network coding as a new networking paradigm
    - Fundamentals
  - Application of NC to:
    - Distributed storage systems
    - Peer-to-peer networks
    - WiFi networks

Network coding

Network coding: generalization of routing – the intermediate nodes can modify the payload of packets they have to transfer

Transport networks

- Oriented valued graph *G = (V, E, c)*
- *c(p, q):* capacity of edge *(p, q)*
- *f(p, q)*: rate or flow of edge *(p, q)*
- Examples:
  - Water pipes
  - Pipelines
  - Transportation lanes
  - Freight traffic
  - Communication networks



Conditions

Capacity c : $V \times V \rightarrow \mathbf{R}$ with $c(p, q) \geq 0$ and if (p, q) not in E then $c(p, q) = 0$

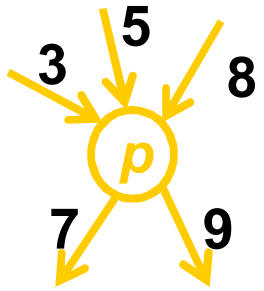
Flow f : $V \times V \rightarrow \mathbf{R}$

Source s ∈ V, sink (terminal) t ∈ V

- Accessiblity: all the vertices are on a path from *s* to *t*
- Constraint of capacity:
  - for all $p, q \in V$, $f(p, q) \leq c(p, q)$

- Anti-symmetry:
  - for *all* $p, q \in V$, $f(q, p) = -f(p, q)$

- Flow conservation:
  - for all $p \in V \backslash \{s,t\}$, $\Sigma (f(p, q) \mid q \in V) = 0$



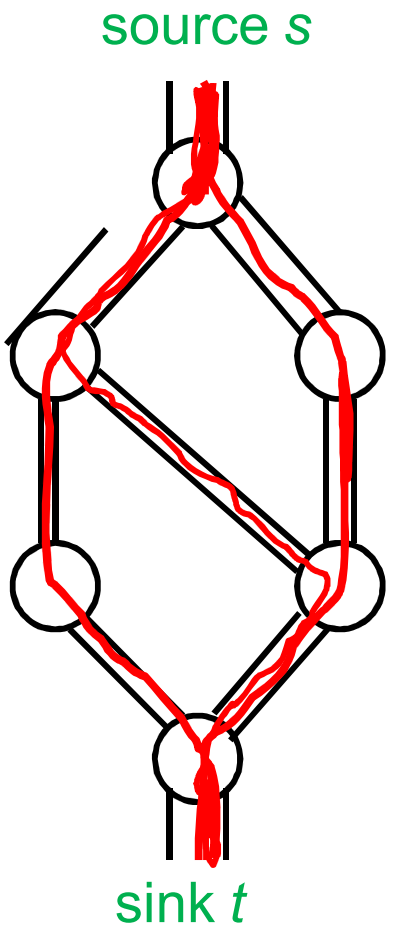
Flow

- Value of the flow

  $| f | = \Sigma (f (p, t) \mid p \in V)$

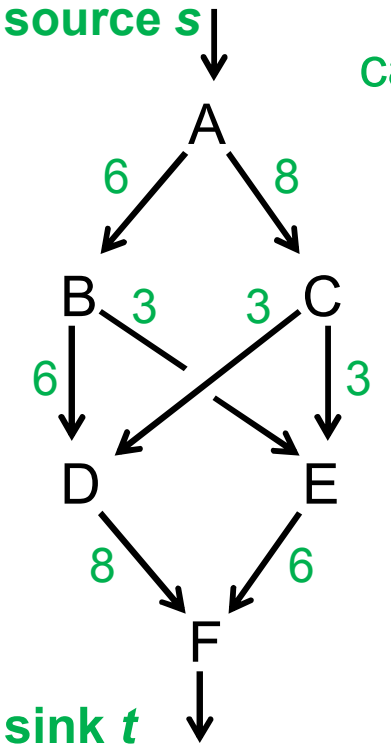
  that arrives to the sink

- Properties
  - for all $p \in V$, $f (p, p) = 0$
  - for all $q \in V$, $\Sigma (f (p, q) \mid p \in V) = 0$



Problem of max flow

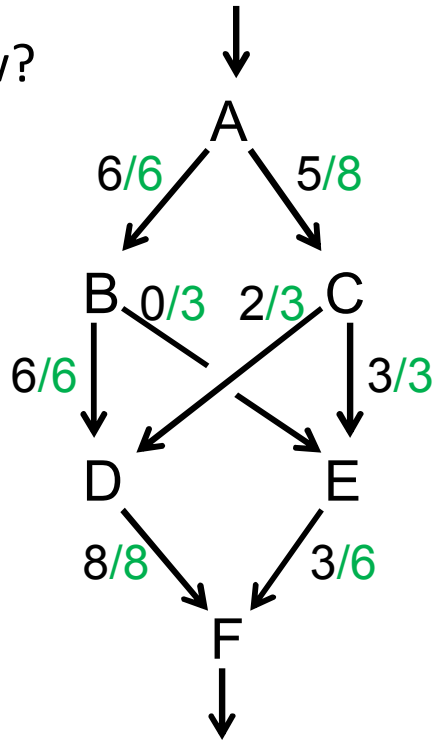
- Given an oriented valued graph: *G = (V, E, c)*
- Compute a maximum flow allocation:
  - set *f(p,q)*, for all *p,q ϵ E*, such that |f| is maximum



capacities

A maximum flow?

| *f* | = 11



Cut

- *(X, Y)* is a cut of *G* iif:

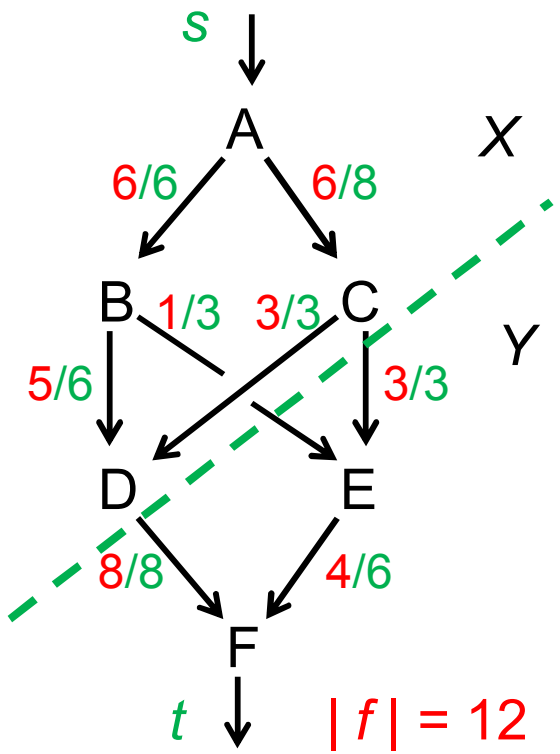
  (X, Y) partition of *V* with $s \in X$, $t \in Y$

- Capacity of the cut:

  $c(X, Y) = \sum (c(x, y) \mid x \in X, y \in Y)$

- Flow of the cut

  $f(X, Y) = \sum (f(x, y) \mid x \in X, y \in Y)$



X = {A,B,C,D} Y = {E,F} c(X,Y) = 14 f (X,Y) = 12

Properties

- Properties:
  1. For any cut *(X,Y)*, we have $f(X,Y) = |f|$
  2. For any cut *(X,Y)*, $f(X,Y) \leq c(X,Y)$
  3. *f* is a maximum flow (allocation) iif
     there exists a cut (X_0,Y_0) such that $|f| = c(X_0,Y_0)$

->Theorem of min-cut max-flow:

The maximum flow is the minimum cut capacity.

Min-cut

$X_0 = \{A, C\}$

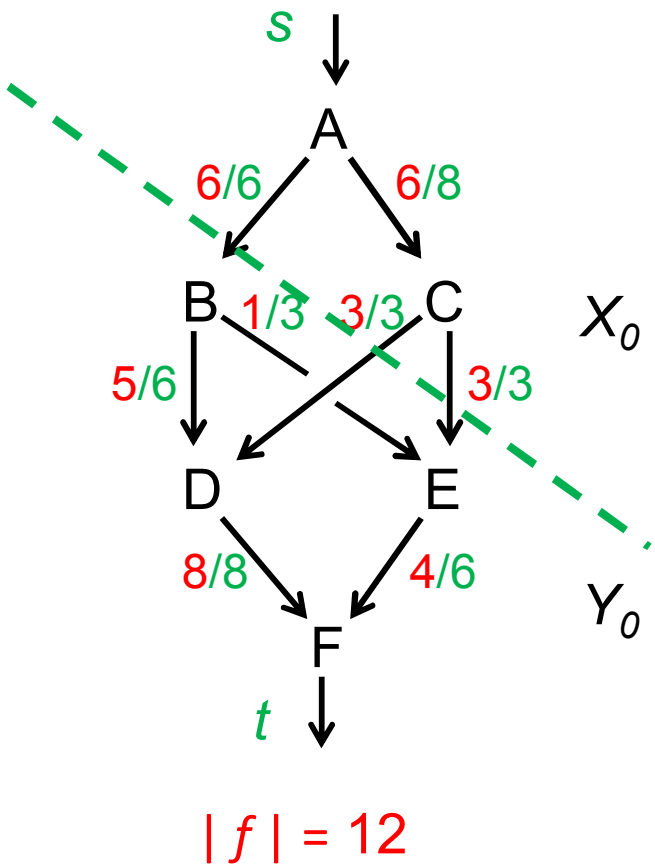
$Y_0 = \{B, D, E, F\}$

$c(X_0, Y_0) = 12$

(X_0, Y_0) of min capacity

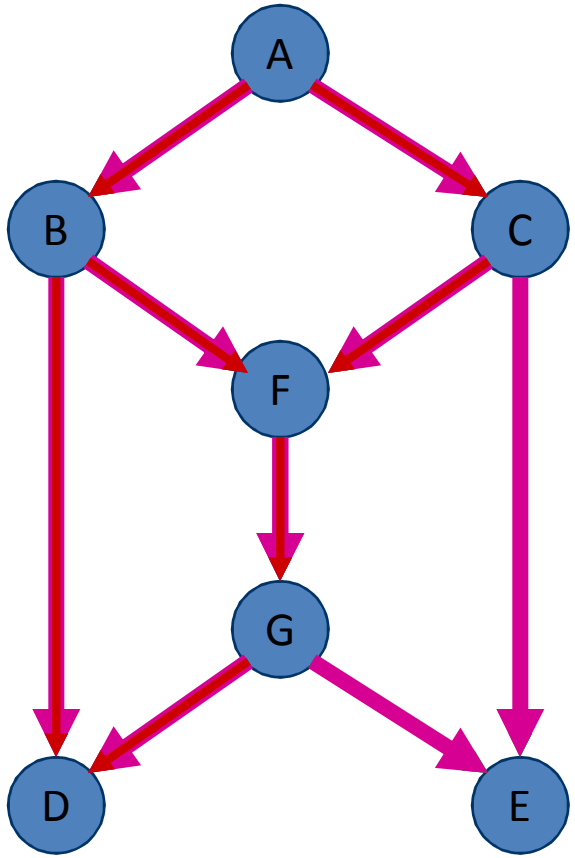
$f(X_0, Y_0) = 12$

Max flow



Multicast Problem

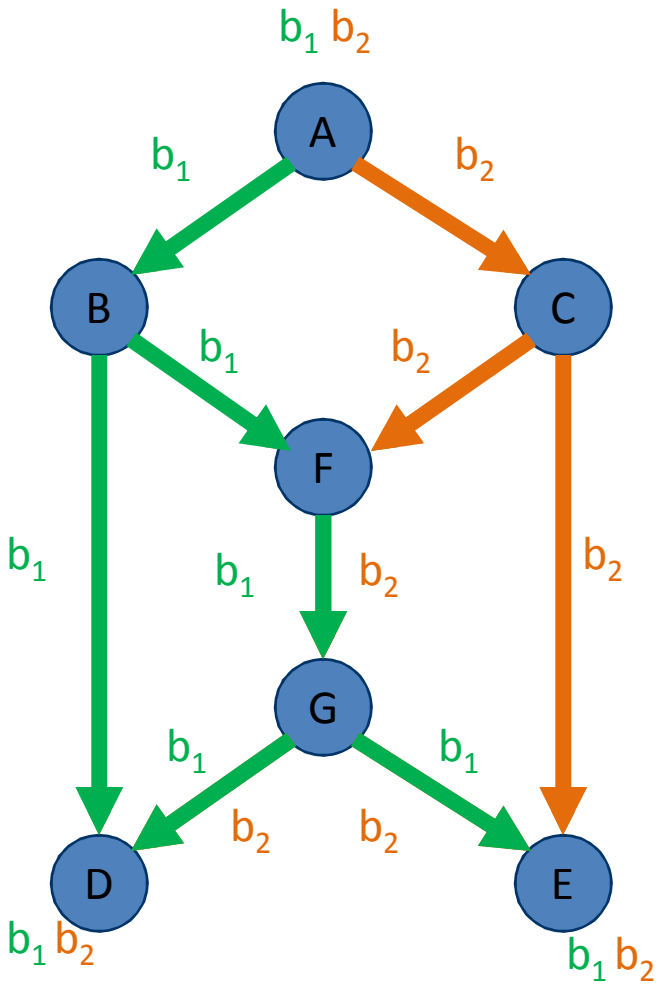
- Butterfly Networks: Each edge's capacity is 1.
- Max-Flow from A to D = 2
- Max-Flow from A to E = 2
- Multicast Max-Flow from A to D and E = 1.5
- Max-Flow for each individual connection is not achieved.



Network Coding

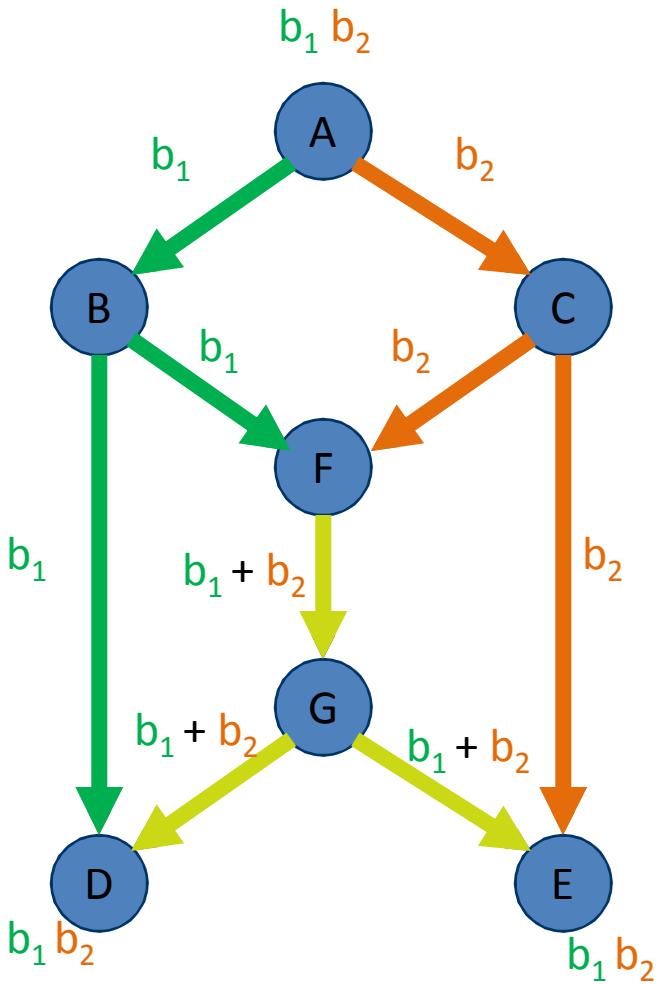
- Introduction
- Linear Network Coding
- Transfer Matrix
- Network Coding Solution
- Connection between an Algebraic Quantity and a Graph Theoretic Tool
- Finding Network Coding Solution

Introduction: multicast with routing



Introduction: multicast with network coding

- Ahlswede et al. (2000)
  - With network coding, every sink obtains the maximum flow.
- Li et al. (2003)
  - Linear network coding is enough to achieve the maximum flow, for multicast in directed networks



R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," IEEE Transactions on Information Theory, vol. 46, no. 4, July 2000.

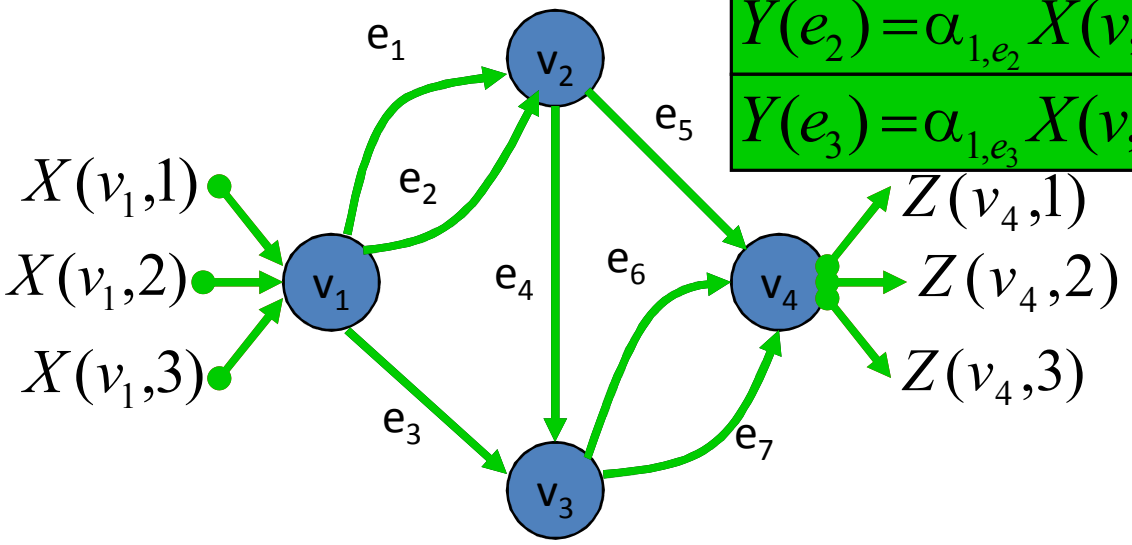
Linear Network Coding

- Random Processes in a Linear Network
  - Source Input: $X(v,l)=\{x_0(v,l),x_1(v,l),\ldots\}$
  - Info. along Edges: $Y(e)=\{y_0(e),y_1(e),\ldots\}$
  - Sink Output: $Z(v,l)=\{z_0(v,l),z_1(v,l),\ldots\}$
- Relationship between them

$$Y(e)=\sum_{l=1}^{\mu(v)}\alpha_{l,e}X(v,l)+\sum_{e':head(e')=tail(e)}\beta_{e',e}Y(e')$$

$$Z(v,j)=\sum_{e':head(e')=v}\varepsilon_{e',j}Y(e')$$

Transfer Matrix



$$Y(e_1)=\alpha_{1,e_1}X(v,1)+\alpha_{2,e_1}X(v,2)+\alpha_{3,e_1}X(v,3)$$

$$Y(e_2)=\alpha_{1,e_2}X(v,1)+\alpha_{2,e_2}X(v,2)+\alpha_{3,e_2}X(v,3)$$

$$Y(e_3)=\alpha_{1,e_3}X(v,1)+\alpha_{2,e_3}X(v,2)+\alpha_{3,e_3}X(v,3)$$

$$Y(e_4)=\beta_{e_1,e_4}Y(e_1)+\beta_{e_2,e_4}Y(e_2)$$

$$Y(e_5)=\beta_{e_1,e_5}Y(e_1)+\beta_{e_2,e_5}Y(e_2)$$

$$Y(e_6)=\beta_{e_3,e_6}Y(e_3)+\beta_{e_4,e_6}Y(e_4)$$

$$Y(e_7)=\beta_{e_3,e_7}Y(e_3)+\beta_{e_4,e_7}Y(e_4)$$

$$Z(v_4,1)=\varepsilon_{e_5,1}Y(e_5)+\varepsilon_{e_6,1}Y(e_6)+\varepsilon_{e_7,1}Y(e_7)$$

$$Z(v_4,2)=\varepsilon_{e_5,2}Y(e_5)+\varepsilon_{e_6,2}Y(e_6)+\varepsilon_{e_7,2}Y(e_7)$$

$$Z(v_4,3)=\varepsilon_{e_5,3}Y(e_5)+\varepsilon_{e_6,3}Y(e_6)+\varepsilon_{e_7,3}Y(e_7)$$

Transfer Matrix

$$Let \quad \bar{x} = (X(v_1,1), X(v_1,2), X(v_1,3))$$
$$\bar{z} = (Z(v_4,1), Z(v_4,2), Z(v_4,3))$$
$$\bar{z} = \bar{x} \cdot_{e_2} M$$

$$M = A \cdot \begin{bmatrix} \beta_{e_1,e_5} & \beta_{e_1,e_4}\beta_{e_4,e_6} & \beta_{e_1,e_4}\beta_{e_4,e_7} \\ \beta_{e_2,e_5} & \beta_{e_2,e_4}\beta_{e_4,e_6} & \beta_{e_2,e_4}\beta_{e_4,e_7} \\ 0 & \beta_{e_3,e_6} & \beta_{e_3,e_7} \end{bmatrix} \cdot B$$

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

$$Y(e_1) = \alpha_{1,e_1} X(v,1) + \alpha_{2,e_1} X(v,2) + \alpha_{3,e_1} X(v,3)$$
$$Y(e_2) = \alpha_{1,e_2} X(v,1) + \alpha_{2,e_2} X(v,2) + \alpha_{3,e_2} X(v,3)$$
$$Y(e_3) = \alpha_{1,e_3} X(v,1) + \alpha_{2,e_3} X(v,2) + \alpha_{3,e_3} X(v,3)$$
$$Y(e_4) = \beta_{e_1,e_4} Y(e_1) + \beta_{e_2,e_4} Y(e_2)$$
$$Y(e_5) = \beta_{e_1,e_5} Y(e_1) + \beta_{e_2,e_5} Y(e_2)$$
$$Y(e_6) = \beta_{e_3,e_6} Y(e_3) + \beta_{e_4,e_6} Y(e_4)$$
$$Y(e_7) = \beta_{e_3,e_7} Y(e_3) + \beta_{e_4,e_7} Y(e_4)$$
$$Z(v_4,1) = \varepsilon_{e_5,1} Y(e_5) + \varepsilon_{e_6,1} Y(e_6) + \varepsilon_{e_7,1} Y(e_7)$$
$$Z(v_4,2) = \varepsilon_{e_5,2} Y(e_5) + \varepsilon_{e_6,2} Y(e_6) + \varepsilon_{e_7,2} Y(e_7)$$
$$Z(v_4,3) = \varepsilon_{e_5,3} Y(e_5) + \varepsilon_{e_6,3} Y(e_6) + \varepsilon_{e_7,3} Y(e_7)$$

Network Coding Solution

$$\bar{z} = \bar{x} \cdot M$$

$$M = A \cdot \begin{bmatrix} \beta_{e_1,e_5} & \beta_{e_1,e_4}\beta_{e_4,e_6} & \beta_{e_1,e_4}\beta_{e_4,e_7} \\ \beta_{e_2,e_5} & \beta_{e_2,e_4}\beta_{e_4,e_6} & \beta_{e_2,e_4}\beta_{e_4,e_7} \\ 0 & \beta_{e_3,e_6} & \beta_{e_3,e_7} \end{bmatrix} \cdot B$$

$$A = \begin{bmatrix} \alpha_{1,e_1} & \alpha_{1,e_2} & \alpha_{1,e_3} \\ \alpha_{2,e_1} & \alpha_{2,e_2} & \alpha_{2,e_3} \\ \alpha_{3,e_1} & \alpha_{3,e_2} & \alpha_{3,e_3} \end{bmatrix},$$

$$B = \begin{bmatrix} \varepsilon_{e_5,1} & \varepsilon_{e_5,2} & \varepsilon_{e_5,3} \\ \varepsilon_{e_6,1} & \varepsilon_{e_6,2} & \varepsilon_{e_6,3} \\ \varepsilon_{e_7,1} & \varepsilon_{e_7,2} & \varepsilon_{e_7,3} \end{bmatrix}$$

- We want $\bar{z} = \bar{x}$
- Choose A to be an identity matrix.
- Choose B to be the inverse of

$$\begin{bmatrix} \beta_{e_1,e_5} & \beta_{e_1,e_4}\beta_{e_4,e_6} & \beta_{e_1,e_4}\beta_{e_4,e_7} \\ \beta_{e_2,e_5} & \beta_{e_2,e_4}\beta_{e_4,e_6} & \beta_{e_2,e_4}\beta_{e_4,e_7} \\ 0 & \beta_{e_3,e_6} & \beta_{e_3,e_7} \end{bmatrix}$$

NETWORK CODING

SOLUTION EXISTS IF DETERMINANT OF M IS NON-ZERO

Connection between an Algebraic Quantity and a Graph Theoretic Tool

- Let a linear network be given with source node ν, sink node ν', and a desired connection $c = (\nu, \nu', \chi(\nu, \nu'))$ of rate $R(c)$. The following three statements are equivalent.
  - 1. The connection $c = (\nu, \nu', \chi(\nu, \nu'))$ is possible.
  - 2. The Min-Cut Max-Flow bound is satisfied
  - 3. The determinant of the $R(c) \times R(c)$ transfer matrix M is non-zero over the ring $F_2[\ldots, \alpha_{l,e}, \ldots, \beta_{e,e'}, \ldots, \varepsilon_{e',j}, \ldots]$.

Finding Network Coding Solution

- Koetter and Medard (2003): Greedy Algorithm
- Let a delay-free communication network G and a solvable multicast problem be given with one source and N receivers. Let R be the rate at which the source generates information. There exists a solution to the network coding problem in a finite field F_{2^m} with

$$m \leq \lceil \log_2(NR+1) \rceil$$

Random Network Coding

- **η** is the number of edges and ***d*** is the number of sink nodes

Lemma 2.5 *Let P be a nonzero polynomial in $\mathbb{F}[\xi_1, \xi_2, \ldots]$ of degree less than or equal to $d\eta$, in which the largest exponent of any variable ξ_i is at most d. Values for $\xi_1, \xi_2, \ldots$ are chosen independently and uniformly at random from $\mathbb{F}_q \subseteq \mathbb{F}$. The probability that P equals zero is at most $1 - (1 - d/q)^\eta$ for $d < q$.*

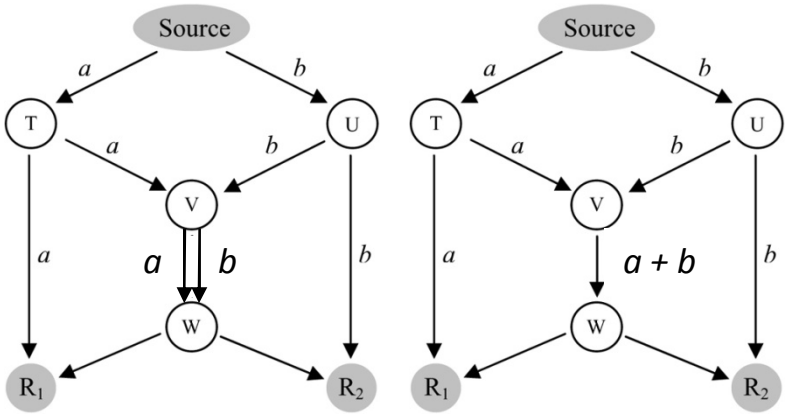
--> Choosing the coding coefficients uniformly at random in $\mathbb{F}_q$, with *q* large enough, is sufficient to ensure high probability of decoding at the sink(s)
Or: For a fixed success probability, the field size needs to be on the order of the number of links **η** multiplied by the number of receivers ***d***.
--> Fully decentralized

RLNC achieves robustness to link failures

T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in IEEE International Symposium on Information Theory, 2003.
T. Ho and D.S. Lun, *Network Coding: An Introduction*, Cambridge University Press, 2008

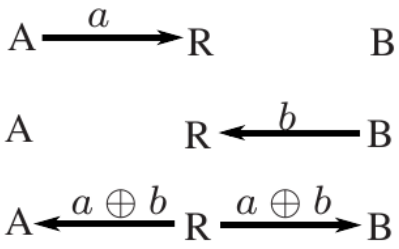
Inter-session network coding

- Intra-session network coding

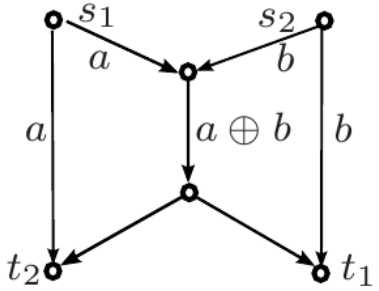


- improves the throughput of lossless multicast sessions, and of lossy unicast or multicast sessions

- Inter-session network coding



The COPE example



The butterfly example

- is necessary to achieve optimal throughput in general

On lossy channels: NC for unicast improves rate AND delay

ε_{12}: Erasure probability on link $(1, 2)$.
ε_{23}: Erasure probability on link $(2, 3)$.

- To cope with erasures on a channel with **ε**, we need to send ***N***=***K***+***M*** pkts to retrieve ***K*** info pkts (***M*** such that (***K***+***M***)(1- **ε**)=***K***).
- The additional ***M*** pkts are not replications, but function of all ***K*** info pkts.
- The ***K*** info pkts must be retrieved from the ***N'***≤***N*** received pkts. Then there are 3 solutions for the 2 links problem :

On lossy channels: NC for unicast improves rate AND delay

- S sends N pkts such that $N(1-\varepsilon_{12})(1-\varepsilon_{23})=K$. Then the end-to-end rate is $K/N=(1-\varepsilon_{12})(1-\varepsilon_{23})$
- If the relay is able to decode (retrieve all K info pkts from the N_1' received) then
  - S can send N_1 such that $N_1(1-\varepsilon_{12})=K$
  - R can send N_2 such that $N_2(1-\varepsilon_{23})=K$

  Then the E2E rate is $\min(1-\varepsilon_{12}, 1-\varepsilon_{23}) \geq N(1-\varepsilon_{12})(1-\varepsilon_{23})$

  BUT: such block coding entails delay
- RLNC at relay: R generates pkts on the fly by mixing those received from S, w/o decoding

  --> no delay and max rate

A.G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright and K. Ramchandran

Network Coding for Distributed Storage Systems, IEEE Transactions on Information Theory, Vol. 56(9), 4539-4551, Sept. 2010.

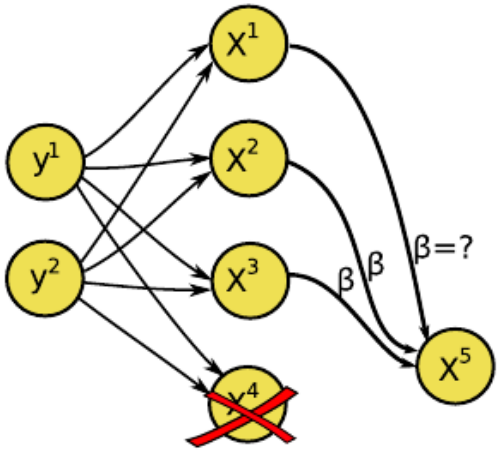
NC FOR DISTRIBUTED STORAGE SYSTEMS (DSS)

NC for Distributed Storage Systems (DSSs)

- DSSs provide reliable access to data through redundancy spread over individually unreliable nodes
- Applications: data centers, P2P storage
- Storing fragments encoded with an erasure code requires less redundancy than replication, for the same level of reliability
- Fragments must be periodically replaced as nodes fail
- Problem: How to generate encoded fragments in a distributed way while transferring as little data as possible across the network?
- -> *regenerating codes* can significantly reduce the repair bandwidth
- -> there is a fundamental tradeoff between storage and repair bandwidth

NC for Distributed Storage Systems (DSSs)

- The simplest form of redundancy is replication.
- But erasure coding offers better storage efficiency:



- Consider a file of size ***M*** divided into ***k*** fragments (of size *M/k*), encode them into ***n*** encoded fragments to store them at ***n*** nodes
- Then the original file can be recovered from any set of ***k*** coded fragments (using an *(n,k)* MDS)
  -> Here we store ***n*** fragments of size ***M/k***

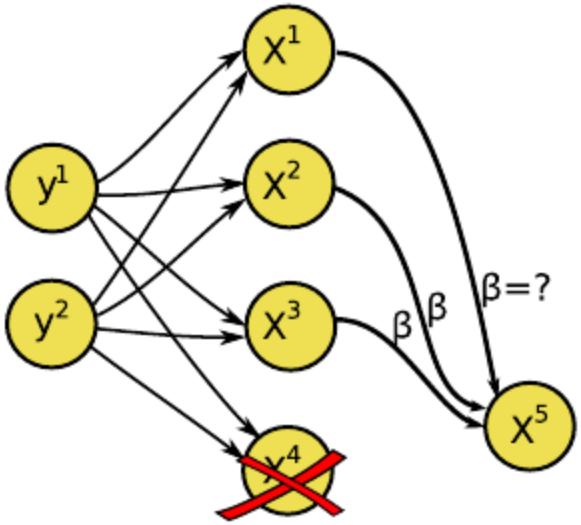
- Storing encoded fragments (generated with MDS) is optimal in terms of redundancy/reliability tradeoff because ***k*** pieces of size ***M/k*** provide the minimum data to recover the file of size ***M***
- If replication were used, we would store ***n*** replicas of size ***M*** each
  -> Then what *n-k* fragments to replicate?

NC for Distributed Storage Systems (DSS)

- Problem: in DSS redundancy must be continually refreshed as nodes fail or leave

-> large data transfers across the network

- File of 2Mb. 2 fragments y^1 and y^2, encoded into 4 x^i
- Any 2 of x^i allow to get back y^1 and y^2
- Assume x^4 fails and a new node x^5 arrives



- The newcomer needs to communicate with existing nodes to create a new encoded packet
- Receiving any 2 encoded fragments is sufficient
- But requires repair bw of ***M*** to generate a ***M/k***-sized fragment at x^5
- If replication is used instead: bw of ***M*** to generate a ***M***-sized replica, so no overhead

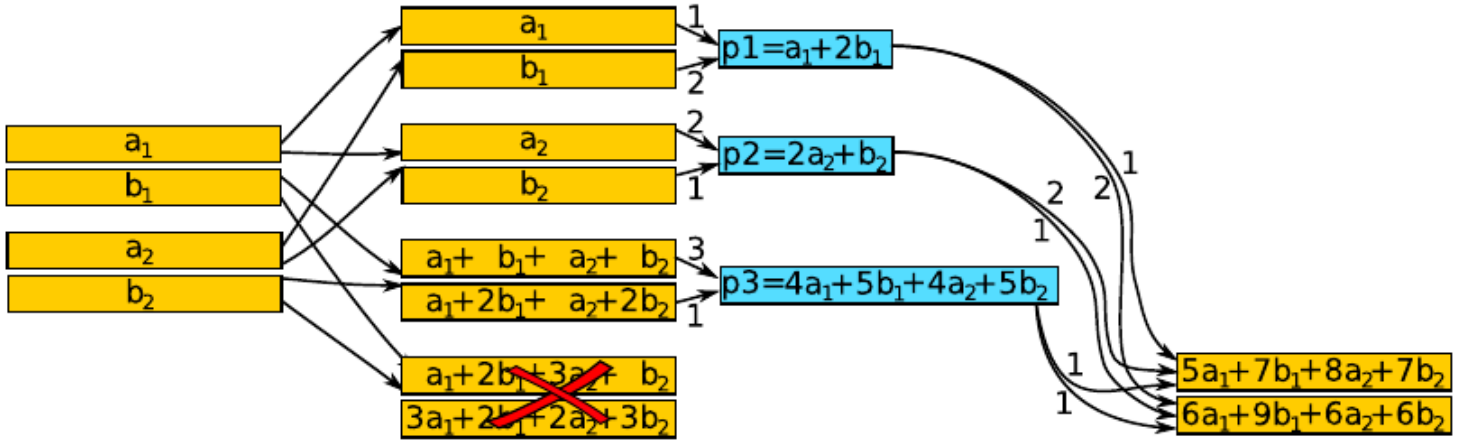
- It was commonly believed that this ***k***-factor overhead in repair bandwidth is unavoidable

-> it has been shown there exist erasure codes that can be repaired without communicating the whole data object

NC for Distributed Storage Systems (DSS)

- A tradeoff between storage and repair bandwidth, and there are codes that achieve every point on this optimal tradeoff curve

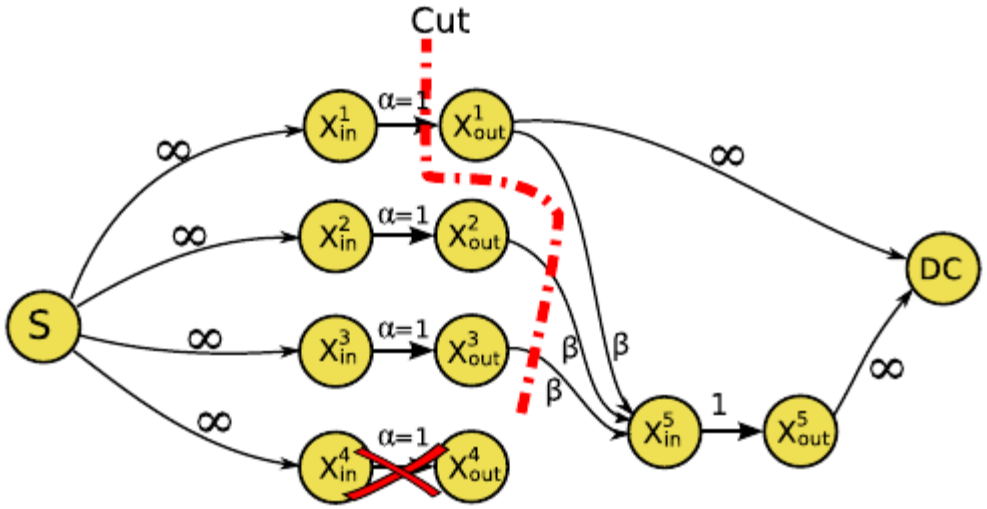
- Example: the previous newcomer can communicate 1.5Mb to repair a failure and this is the information theoretic minimum



- Key point: nodes do not send their information but generate smaller parity packets of their data, and forward them to the newcomer, who further mixes them to generate two new packets.

NC for Distributed Storage Systems (DSS)

- A DSS with a (4,2) erasure code where any 2 fragments suffice



- To reconstruct a new fragment, node x^5 connects to the ***d*** = 3 active storage nodes.
- Assuming **β** bits communicated from each active storage node: what is the minimum **β**?
- -> The min-cut separating the source and the data collector must be larger than ***M*** = 2Mb for reconstruction to be possible.
- For this graph, the min-cut value is given by 1 + 2β, implying that β ≥0.5Mb is sufficient and necessary.

NC for Distributed Storage Systems (DSS)

- Achievable storage-repair bandwidth points
- Setup:
  - maintain ***n*** active storage nodes
  - each storing **α** bits
  - a newcomer is sent **β** bits each from any **d** surviving nodes => total repair bandwidth is **γ**= **dβ**
- A (n, k, d, α, γ)-tuple is feasible, if a code with storage **α** and repair bandwidth **γ** exists.

NC for Distributed Storage Systems (DSS)

- *Theorem*: For any $\alpha \geq \alpha^*(n, k, d, \gamma)$, the points $(n, k, d, \alpha, \gamma)$ are feasible, and linear network codes suffice to achieve them. It is information theoretically impossible to achieve points with $\alpha < \alpha^*(n, k, d, \gamma)$. The threshold function $\alpha^*(n, k, d, \gamma)$ is the following:

$$\alpha^*(n,k,d,\gamma) = \begin{cases} \frac{\mathcal{M}}{k}, & \gamma \in [f(0), +\infty) \\ \frac{\mathcal{M}-g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)), \end{cases}$$

where

$$f(i) \triangleq \frac{2\mathcal{M}d}{(2k-i-1)i+2k(d-k+1)},$$
$$g(i) \triangleq \frac{(2d-2k+i+1)i}{2d},$$

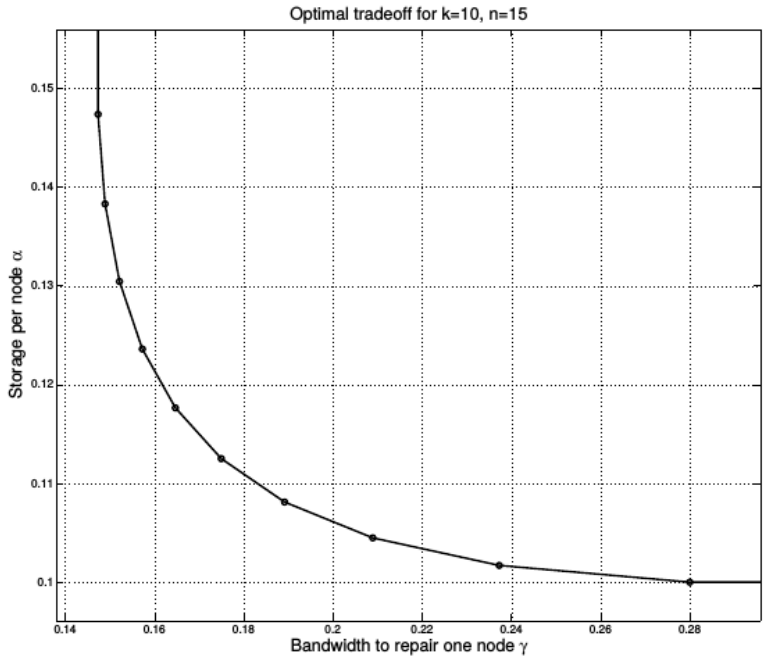
where $d \leq n-1$. For d, n, k given, the minimum repair bandwidth γ is

$$\gamma_{\min} = f(k-1) = \frac{2\mathcal{M}d}{2kd-k^2+k}.$$

NC for Distributed Storage Systems (DSS)

- Gist of the proof: the code repair problem is mapped to a multicasting problem on the information flow graph.
- One important observation is that the minimum repair bandwidth $\gamma = d\beta$ is a decreasing function of the number d of nodes that participate in the repair.

- $M = 1$ and $d = n - 1$
- Traditional erasure coding corresponds to the point $(\gamma = 1, \alpha = 0.1)$.



Baochun Li and Di Niu, "Random Network Coding in Peer-to-Peer Networks: From Theory to Practice," Proceedings of the IEEE , vol.99, no.3, March 2011

NETWORK CODING IN PEER-TO-PEER NETWORKS

NC in P2P networks

- Applications such as file sharing and video streaming in P2P networks is a very promising area for using NC.

- P2P bulk content distribution systems, such as BitTorrent, allow peers to collaborate with one another so that large files can be distributed from one peer to a large number of subscribing receivers, without the aid of dedicated servers.

- Random gossiping: studies the spread of a single or multiple blocks across a group of participating peers
- Random NC is shown to maximize the information diffusion efficiency given limited bandwidth in network topologies.

Random NC in P2P Content Distribution

- The objectives is to distribute some bulk content, such as a large file.
- End hosts connect using overlay links.

- Formally, consider ***n*** peers, the file is divided into ***k*** blocks.
- Assume time is measured in rounds. Each peer uploads a block to ***B*** random neighbors
- -> How many rounds are needed for all peers to receive a copy of the file?

- Random gossiping w/o NC: at least $k+\log_2 n$ rounds are needed

- Will the use of random network coding reduce the number of rounds needed to distribute ***k*** blocks with random gossiping?

Random Gossiping with NC

- Will the use of random network coding reduce the number of rounds needed to distribute k blocks with random gossiping?

- It has been proven that, with NC, optimal broadcast times are achieved, regardless of the network topology and transmission schedule.

- Example:
  - w/o coding: peer B may transmit to peer F block B_1 it already has
  - unless accurate and frequent buffer comparison is performed.
  - w/ NC: B transmits RLCs to F

- Intuitively, the use of RNC has increased the diversity of blocks being transmitted

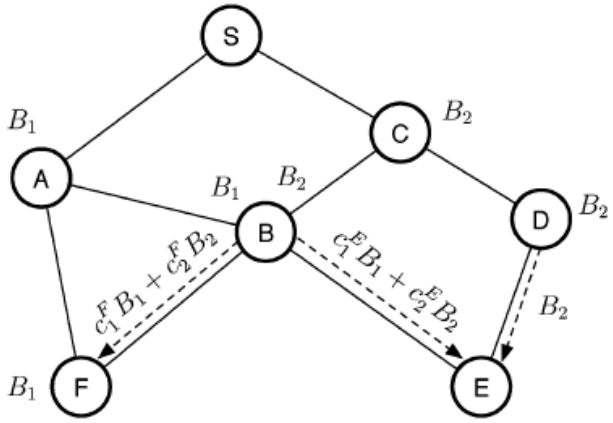


Fig. 1. *An example of distributing two blocks B_1 and B_2 with network coding. S is the source peer.*

Random Gossiping with NC

- In practice, the computational complexity of NC escalates with an increasing number of blocks
- -> blocks in a file are divided into multiple *generations*, and NC only performed within the same generation

- The original file with ***F*** bytes is divided into ***G*** generations, each further divided into ***m*** blocks

- ***M***=***Gm*** original blocks, each of size ***k***=***F***/***M*** bytes

Practical Limitations of NC

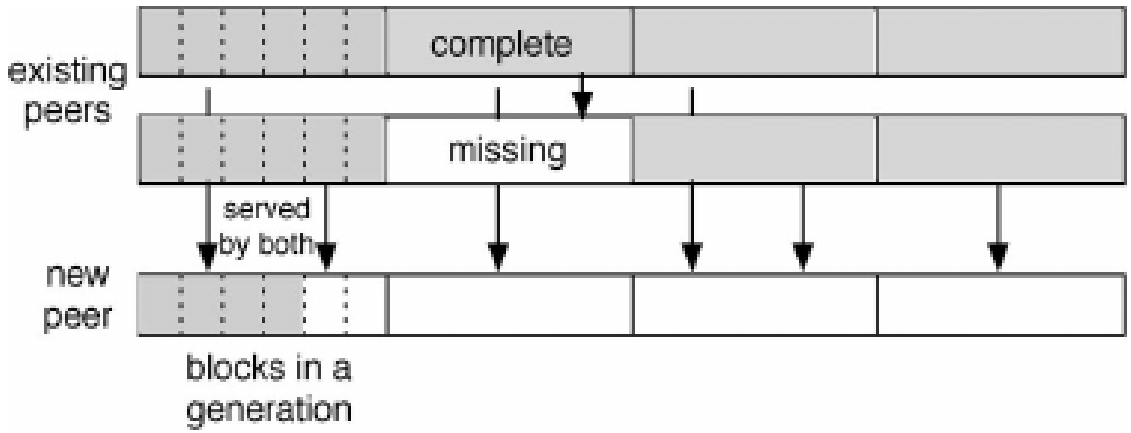
- The need for reconciliation even at the coarser granularity of generations, may negatively affect the advantage of random NC.
- RNC is shown to achieve the best possible performance, but the performance gap between the use of network coding and a well-designed block selection protocol is not clear.
- Such a performance gap may be quite small, as practical block selection protocols without coding, such as BitTorrent, are usually designed to download the rarest blocks first (with the aid of frequent exchanges of buffer states among peers), in the hope of mitigating some of the adverse effects of locating rare blocks as the file download approaches completion.

Random NC in P2P streaming

- NC-based random gossiping is equally useful in P2P streaming, despite the timing of block delivery.
- In P2P live streaming: the only fundamental difference is that a dynamic sliding window of blocks over time needs to be distributed in a streaming fashion, rather than a fixed number of blocks in a static file.
- Performance metrics: initial buffering delay, server bandwidth costs, smooth playback

Live P2P streaming with NC

- P2P live streaming with the use of random network coding: multiple existing peers are able to collaborate and serve coded blocks within the same generation to a new peer joining the session, minimizing its initial buffer delay.



Implementation issues in modern hardware

- Will modern off-the-shelf computer hardware (including dedicated streaming servers, notebook computers, and mobile phones) be able to perform a software implementation of random NC?
- Intel Core Duo 1.83 GHz: can process around 5 MB/s, with 128 blocks of 4 KB each in a generation (saturates a typical DSL uplink)
- Mobile phone: ARM Cortex-A8 CPU - an iPhone 3GS with its CPU operated at 600 MHz is able to achieve 1 MB/s with 128 blocks in a generation
- -> Decoding a high-quality video stream at a streaming bit rate of 768 kb/s, for example, will increase CPU usage by no more than 10%

On-demand streaming with random NC

- Can it be similarly applied to P2P on-demand streaming systems, also called video-on-demand (VoD) systems?
- More challenging: requires not only smooth sequential playback, but also the shortest possible restarting latency after an interactive random seek request
- Random NC has been shown to achieve higher throughput, compared to a block selection algorithm that downloads globally rarest blocks in the generation first, with simulations and a small prototype.
- Not clear whether these benefits can easily be carried over to real-world on-demand streaming systems operating at a large scale

- Practical Challenges:
  - preferable to use a smaller block size (1 block = 1 IP packet)
  - each coefficient occupies a byte, and such overhead depends on the number of blocks
  - However, we cannot afford to use too few blocks in a generation: communication overhead with braking acknowledgments
  - In addition, the need for exchanging buffer availability information between neighboring peers calls for a larger number of blocks in a generation

S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard and J. Crowcroft. *XORs in the air: Practical wireless network coding*. In Proceedings of SIGCOMM 2006.

XORS IN THE AIR: PRACTICAL WIRELESS NETWORK CODING

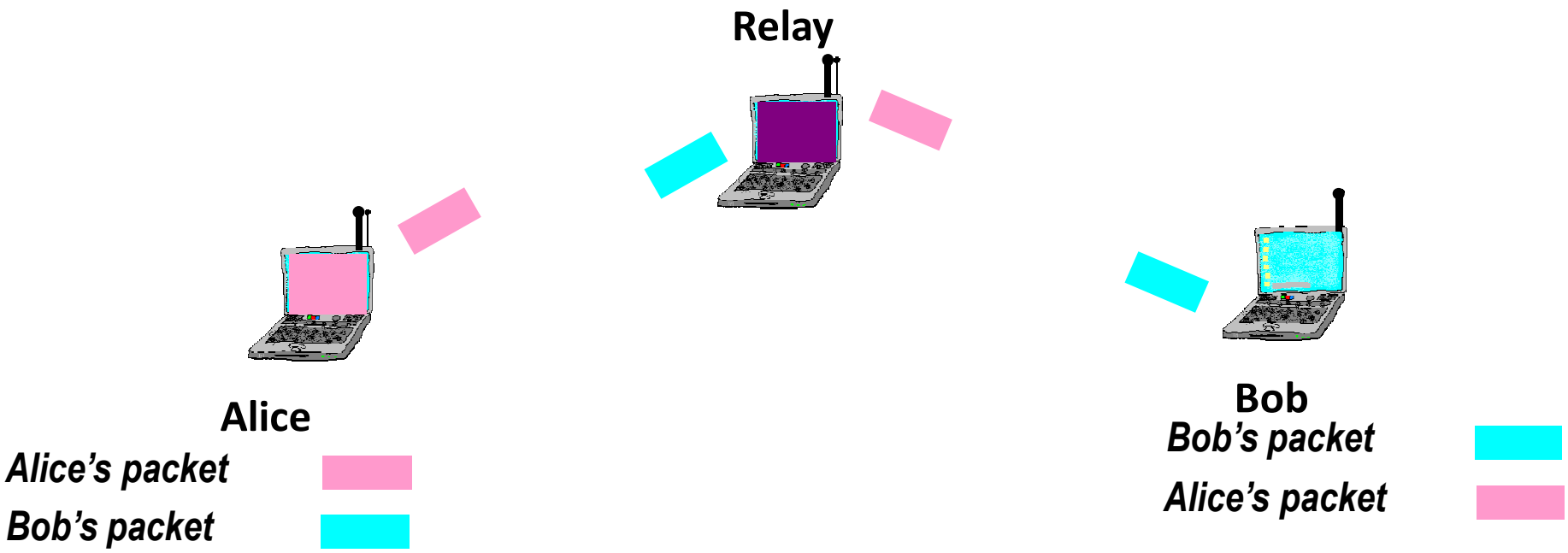
The problem

- Wireless networks are highly resource constrained
  - Bandwidth is the most expensive
  - Power is sometimes an issue too

    --> Serious problems for mesh networks
- How to optimize throughput?
  - Can we send more information?
  - Can we reduce bandwidth requirement?

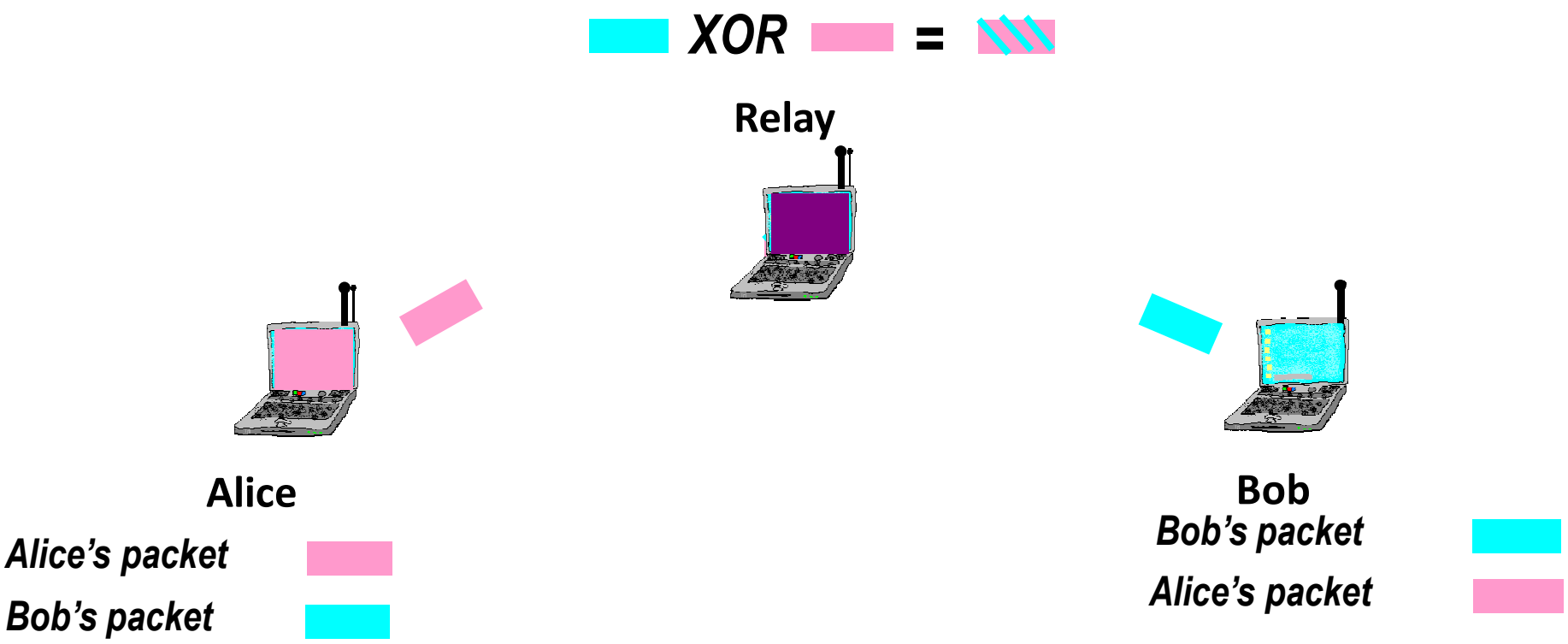
    --> Do both at the same time?

An information exchange scenario



- Multi-hop unicast requires 4 transmissions
- Can we do better?

Can Network Coding help? - An idea



3 transmissions instead of 4

- Saves bandwidth & power
- 33% throughput increase

The COPE approach

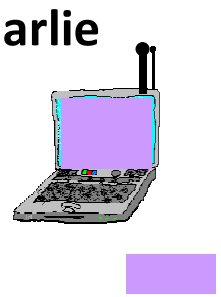
- Considers multiple unicast flows
  - Generalizes the duplex flow scenario
- Opportunistic coding using local info
  - Overhear packets to increase coding gain
  - Online, distributed and deployable
- Emulation and testbed results
  - First real-world implementation

COPE: Opportunistic Coding Protocol

Alice → Bob

Bob → Charlie

Charlie → Alice

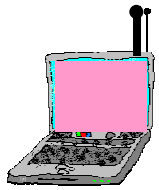


Charlie's packet

Alice's packet

Bob's packet

XOR *XOR* = Relay



Alice

Alice's packet

Bob's packet

Charlie's packet

Bob

Bob's packet

Charlie's packet

Alice's packet

How it works…(Cont.)

- Relay – Encoding
  - Checks packets in queue
  - Combines packets traversing the same three hops in opposite directions
  - Metadata in a header between MAC and IP
  - Broadcast encoded packets
- Alice/Bob – Decoding
  - Keep copies of sent packets
  - Detect the extra header (decoding info)
  - Retrieve the right packet to decode
- Distributed and local action only!

Generalize to COPE

- Nodes snoop on the medium
  - Reception reports to neighbours
- When encoding
  - Identify what packets neighbours have
    - Reception reports and guesses
  - Encode as many packets as possible
    - Provided intended recipients can decode them
- Still distributed and local action only!

The importance of being opportunistic

- Opportunistic coding
  - Only encode if packets in queue
  - No delay penalty
  - Insensitive to flow characteristics
- Opportunistic listening
  - Helps create more coding opportunities

‘Pseudo-broadcast’

- COPE gain is from broadcast medium
- But 802.11 broadcast doesn’t work!
  - No reliability scheme to mask collision loss
  - Send packets at lowest bit rate
  - May actually reduce throughput!
- Pseudo-broadcast
  - Send encoded packets as if unicast
  - Other neighbours overhear
  - Benefit as a unicast packet

Implementation

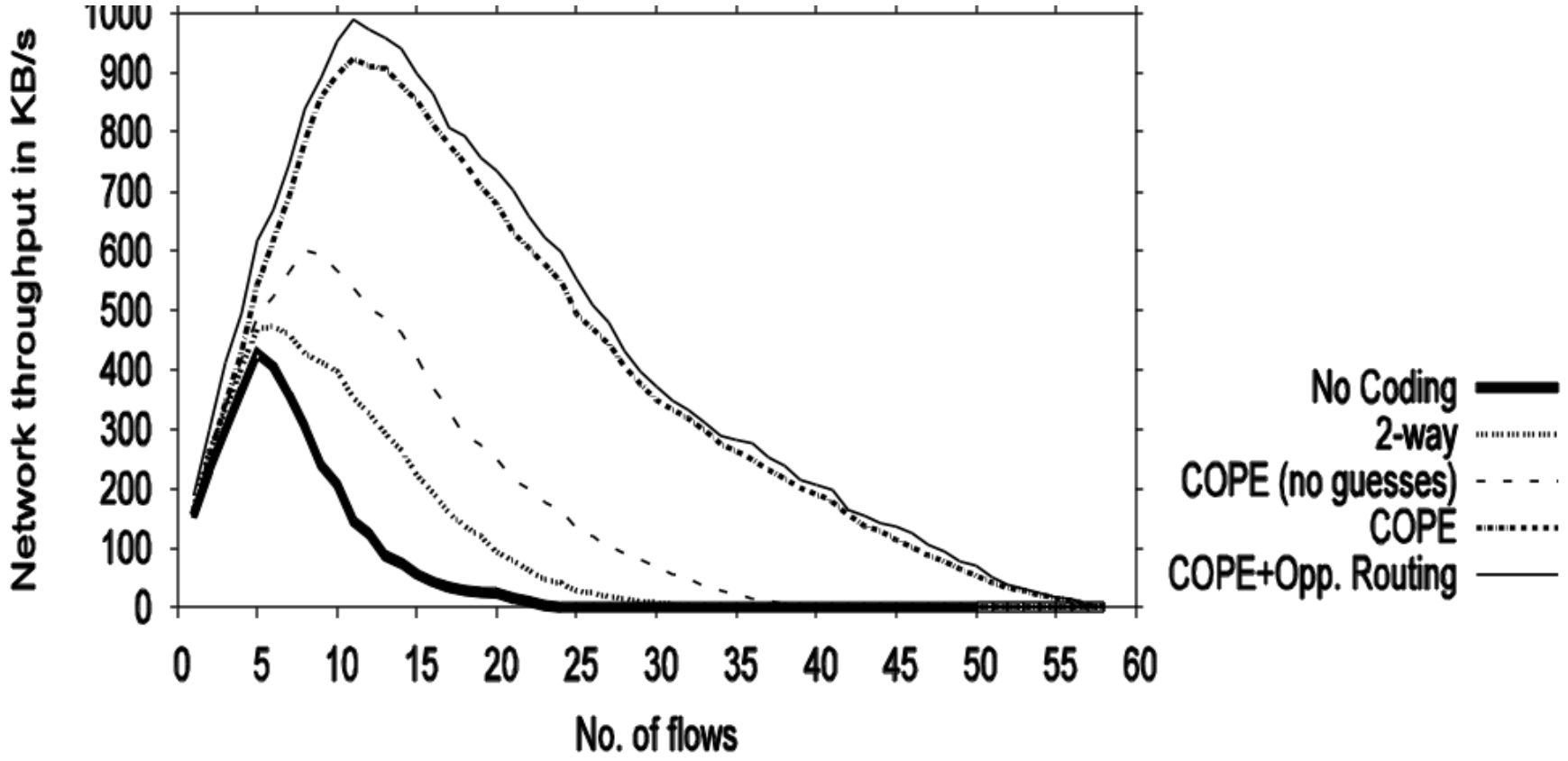
- A shim between MAC and IP
  - Agnostic to protocols above/below
- Emulations
  - General COPE
  - Emsim (part of Emstar) environment
- Testbed
  - Based on the Alice/Bob scenario
  - Extension to Roofnet code (in Click)

Emulation Scenario

- 100 nodes in 800m x 800m
  - Consider range ~50m
- Random senders/receivers
  - Senders always backlogged
  - Bit rate at 11 Mb/s
- Geographic routing
- Metric: end-to-end data traffic throughput over all flows

Emulation performance

Throughput (KB/s)



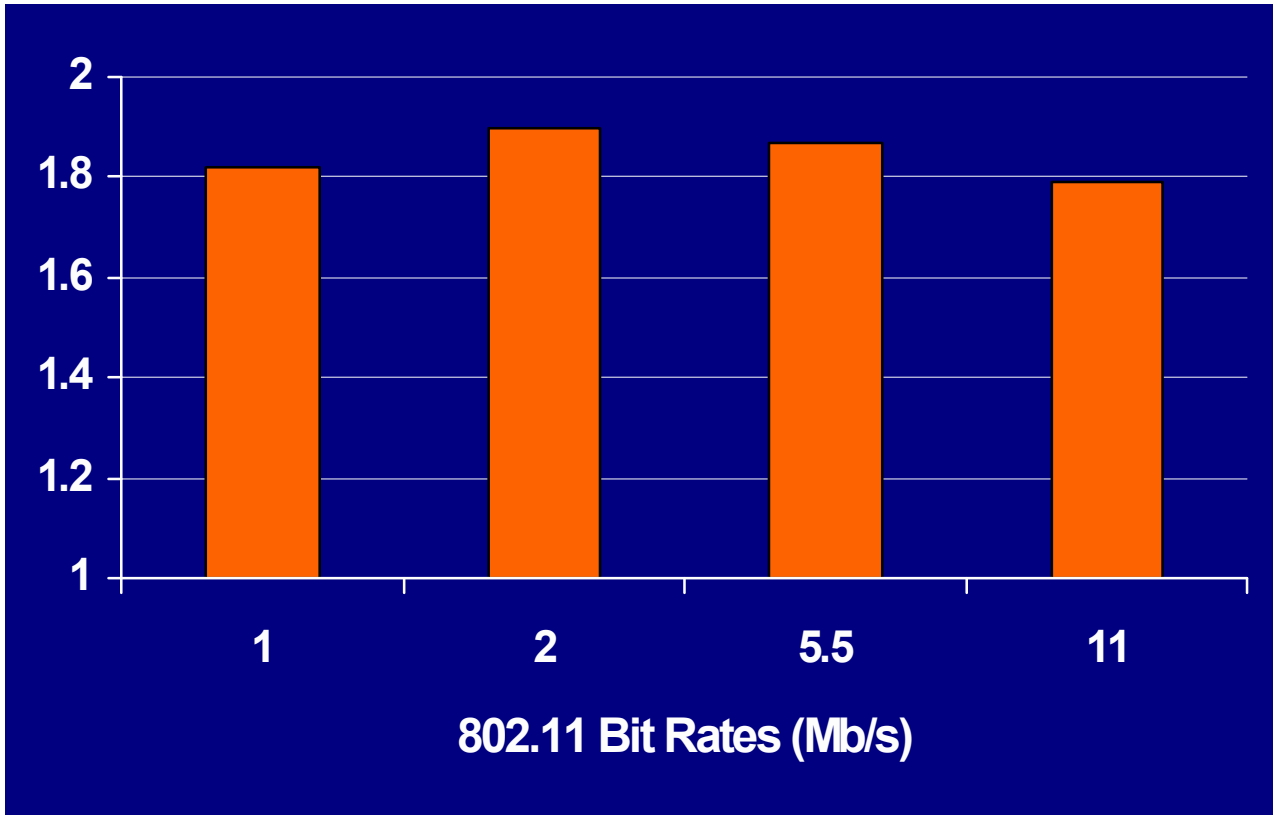
Coding always outperforms no-coding

Testbed setup

- Indoor PCs with 802.11b cards
  - Intersil Prism 2.5 802.11b chipset
  - Connected to omni-directional antenna
  - RTS/CTS disabled
  - 802.11 ad hoc mode
- Randomly chosen 3 nodes from testbed
  - Static routes
  - End nodes send UDP traffic to each other

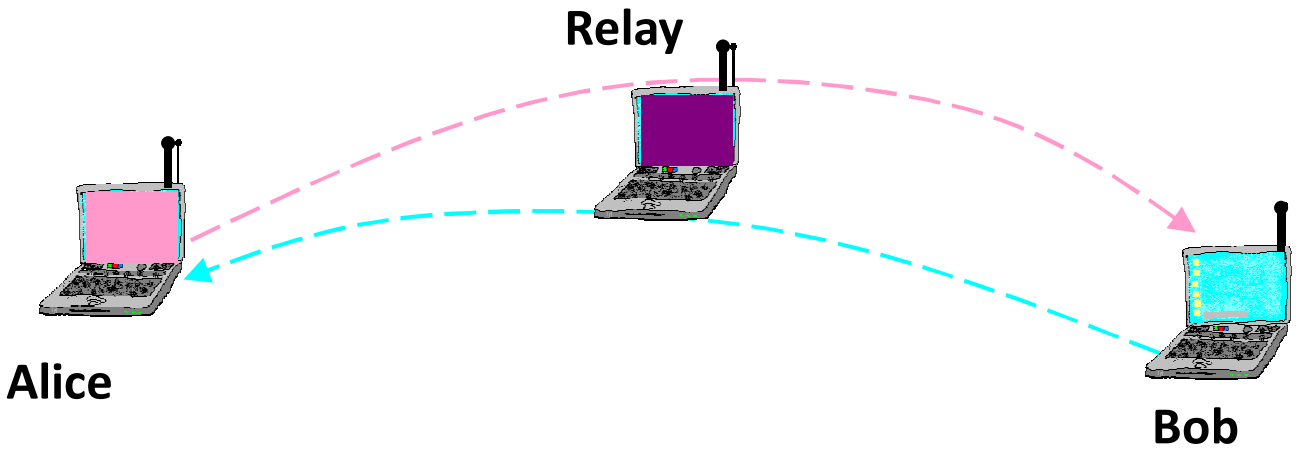
Testbed results

Ratio of Throughput with Coding to No-Coding



Encoding almost doubles the throughput

Why more than 33%?



MAC is fair -> 1/3 BW for each node

- Without coding, relay needs twice as much bandwidth as Alice or Bob
- With coding, all nodes need equal bandwidth

Summary

- Opportunistic approach allows practical integration of network coding into current stack
- Throughput can double in practice
  - Cross-layer effects
  - Congestion plays in our favour
- First implementation of network coding in a wireless environment