Network Coding

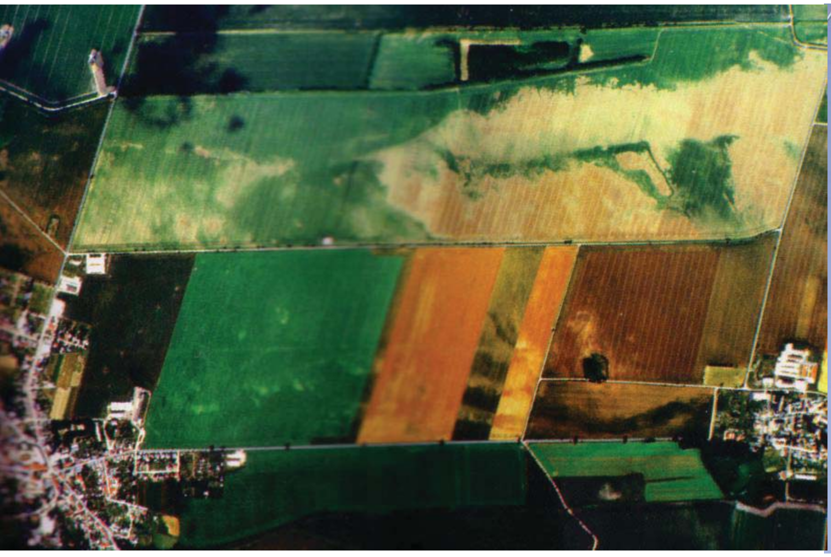
Chapter 5

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Agriculture (precision farming)

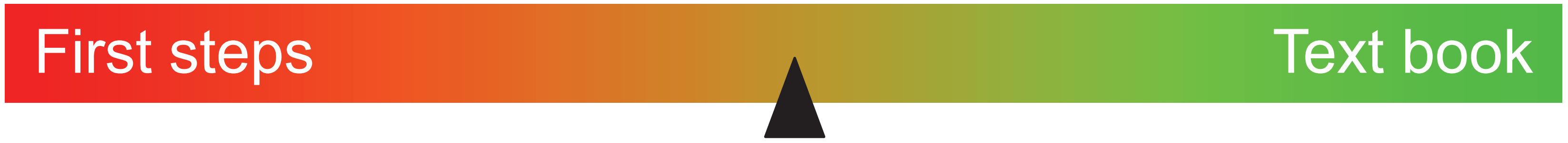
- Farming decision support system based on recent local environmental data
  - High accuracy: GPS tractors
  - Irrigation, fertilization, pest control, etc. are output of function of sunlight, temp., humidity, soil moisture, etc.

[Technology Review, EPFL, IIT]

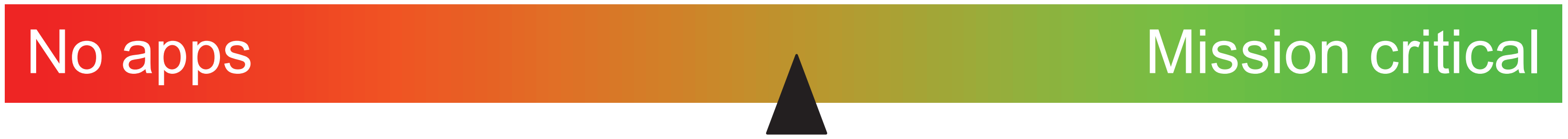


Rating

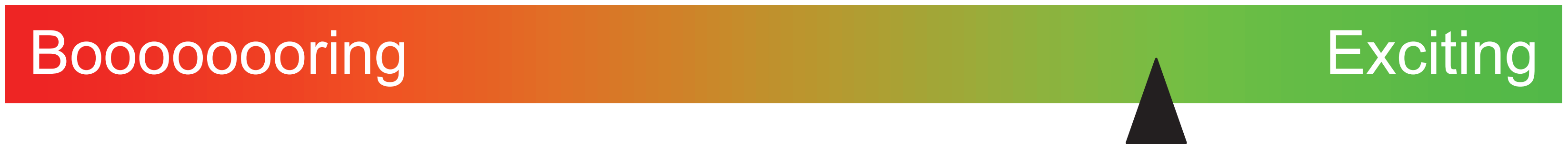
- Area maturity



- Practical importance



- Theory appeal



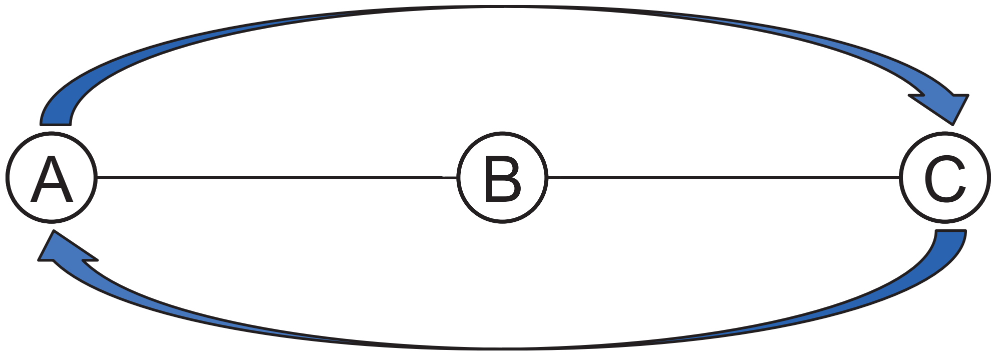
Overview

- Motivation
- Some bounds
- Examples

- Case Study: Data Gathering
  - Self-coding
    - Excursion: Shallow Light Tree
  - Foreign coding
  - Multi-coding

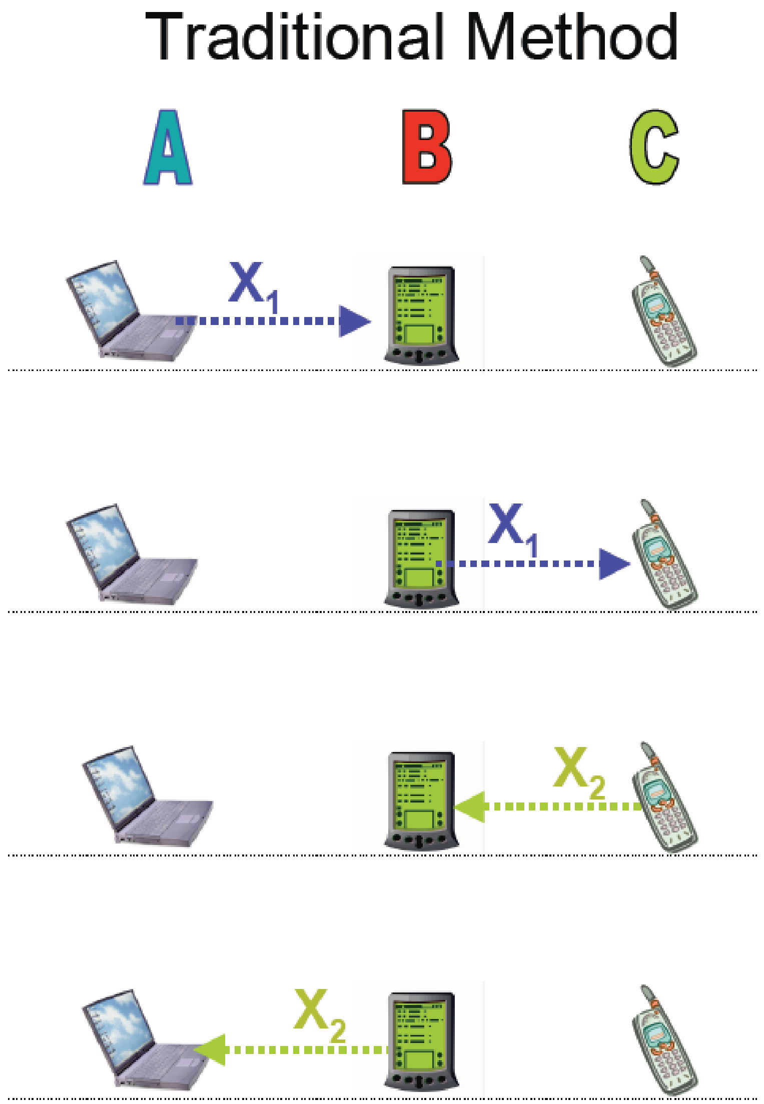
Motivation

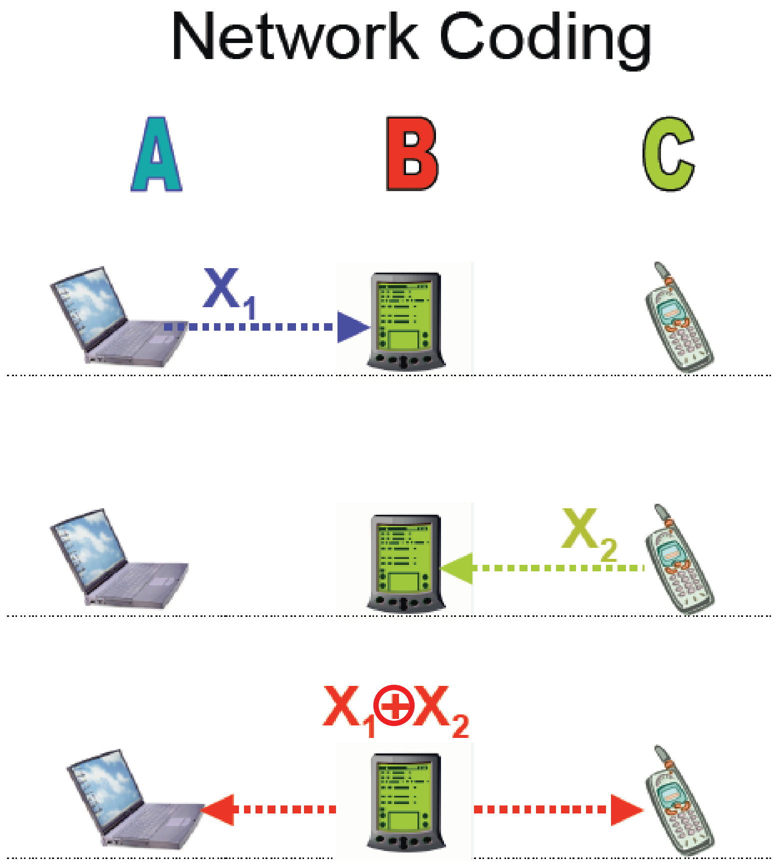
- Given the wireless network as below, where two nodes A and C are too far away to communicate directly. If transmitting one packet costs 1 time unit, how many time units do we need to transmit one packet from A to C and one packet from C to A?



- Traditionally, intermediate nodes in networks just forward data. Network coding deviates from this paradigm, in the sense that intermediate nodes are allowed to process data before forwarding!

Network Coding Saves Transmissions

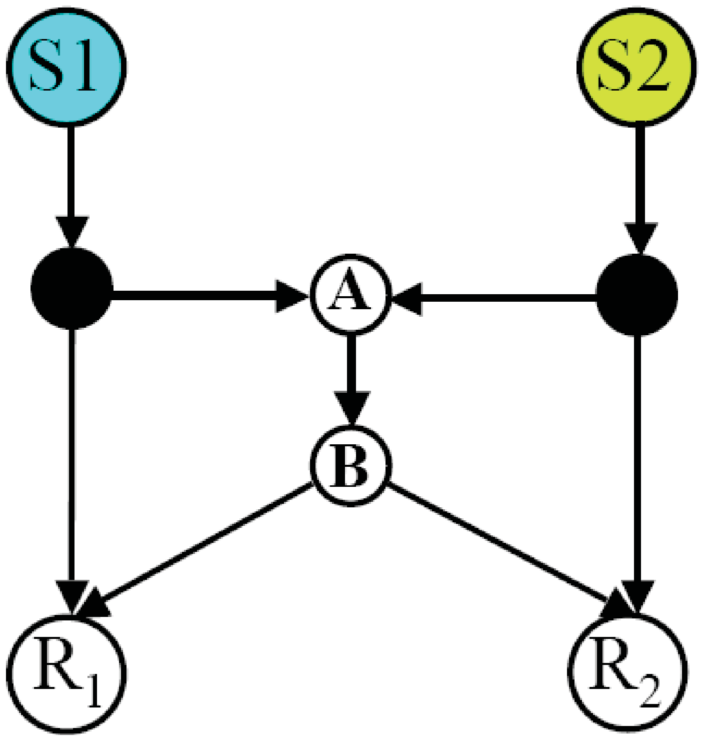




[Christina Fragouli, EPFL]

The Classic Example

- Given two sources, each with a 1 GB file, and two receivers.
- Each directed (wire-line!) link can forward 1 MB/s.
- How long does it take until both receivers have received both files?

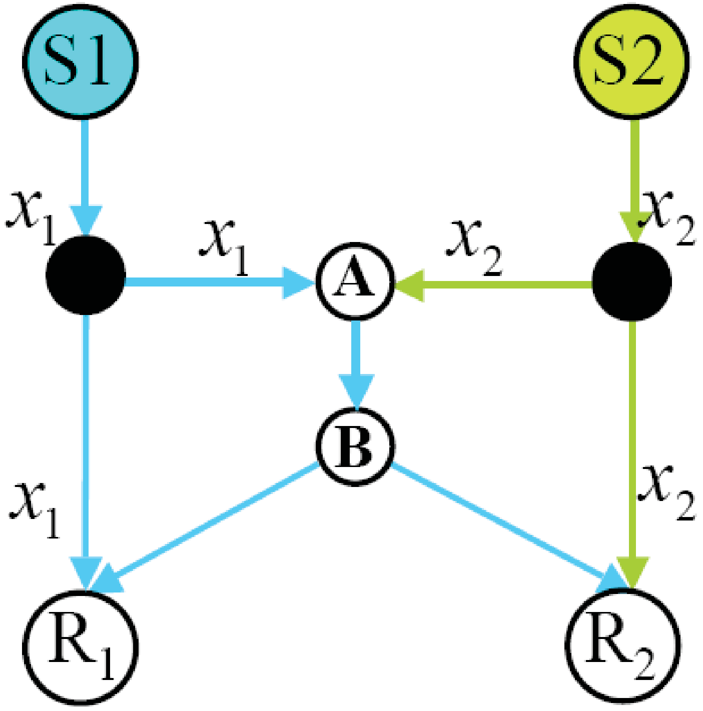


Without Network Coding?

- Well, the naïve solution would first deliver the first file to both receivers, then the second. The total time needed is 2000s. Can we do better (without network coding)?

- First it seems that there is a better "forwarding-only" solution. The picture shows that we can deliver a total of 3MB/s. However, this is not true. Indeed "crossing" traffic must go through the bottleneck link A-B; to deliver the 2GB information through this 1MB/s link, we need 2000s…

- What about with network coding?



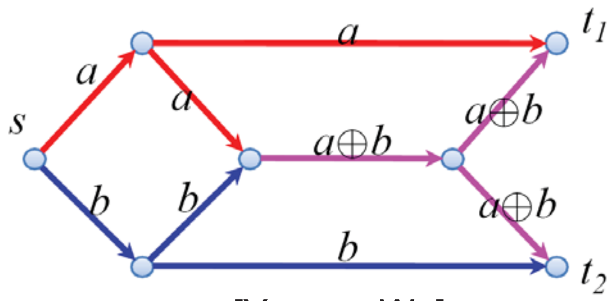
[Christina Fragouli, EPFL]

With Network Coding

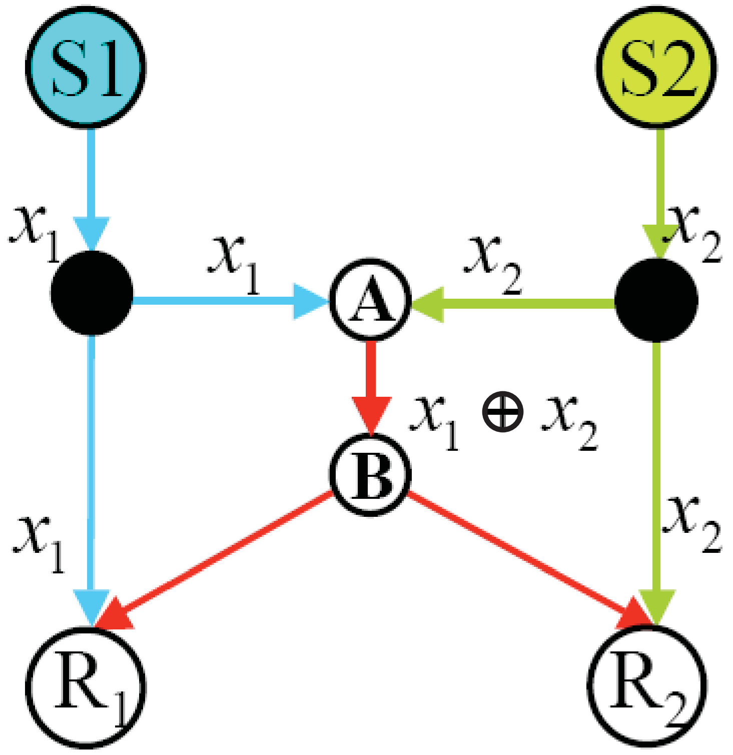
- With network coding, we can indeed deliver all the data in 1000s. Simply let the bottleneck link transmit the XOR of the two packets (or bits), and reconstruct everything at the receivers.

- Network coding saves a factor 2! In this example this is optimal. In general?

- BTW: Same example with one source only is known better:



[Yunnan Wu]



Max-Flow Min-Cut Theorem [Ford-Fulkerson]

We can transmit a flow at rate *r* from source *s* to receiver *t*

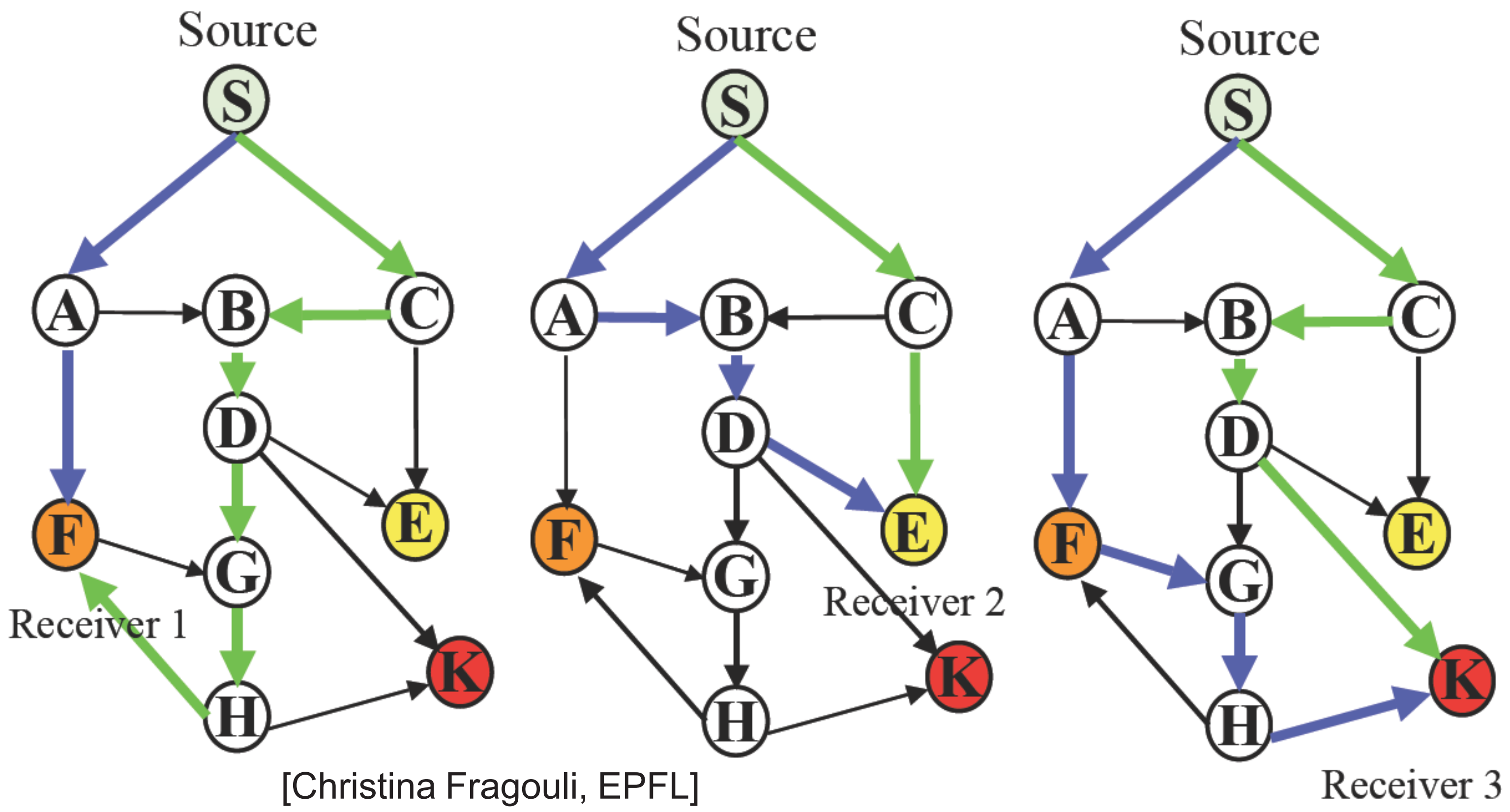
$$\Longleftrightarrow$$

Between source *s* and receiver *t*, the minimum cut is *r*

- Assumes splittable flows
- Can we achieve the max-flow min-cut rate even when multicasting?!

Multicasting

- Consider a network, where the source *S* wants to multicast to three receivers *E*, *F*, and *K*. The min-cut between *S* and each individual receiver is 2. However, some edges (e.g. *SA* and *BD*) are used in conflicting ways! We have to make sure that green paths don't share edges with blue paths…



[Christina Fragouli, EPFL]

Steiner Tree Packing

- To optimize multicasting (without network coding), we need to solve the Steiner tree packing problem (How can you connect source and all destinations by edge-disjoint Steiner trees?). This is known to be notoriously difficult (NP-complete, there are approximations).

- Even if we could solve this, we might end up with a solution which is inferior to the best solution using network coding.
  - Indeed, all previous examples showed that the best Steiner tree packing is a factor 2 worse than the min-cut.

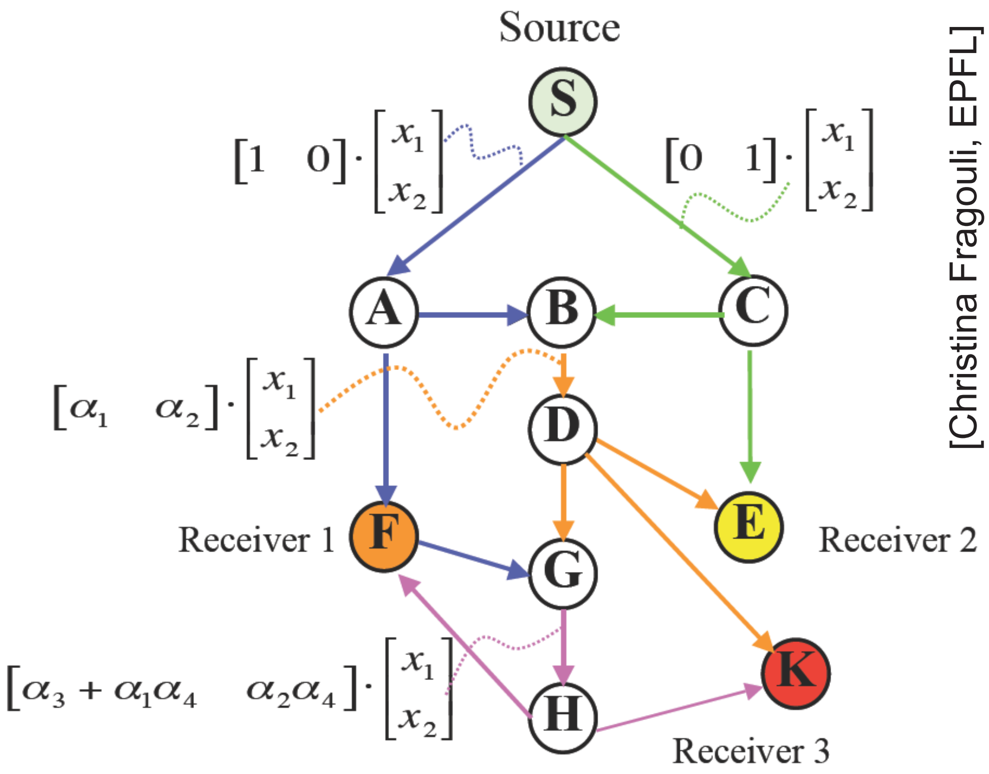
Multicasting w/ Network Coding [Ahlswede, Cai, Li, Yeung]

We can transmit a flow at rate *r* from source *s* to each receiver t_i

$$\Longleftrightarrow$$

Between source *s* and each receiver t_i, the minimum cut is *r*

- This works with various coding schemes
- Indeed, the factor 2 was no coincidence. For undirected networks, it can be shown that network coding can at most improve multicasting by a factor 2.

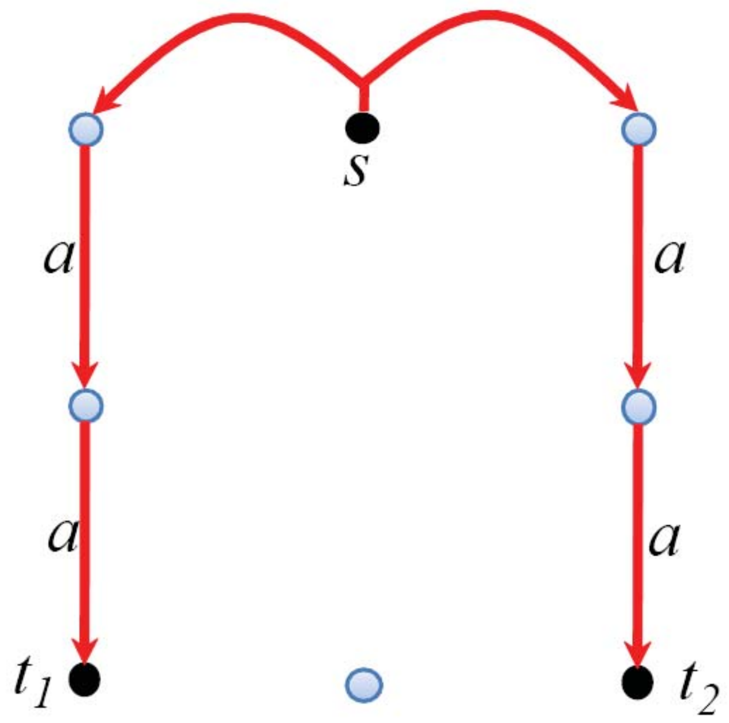


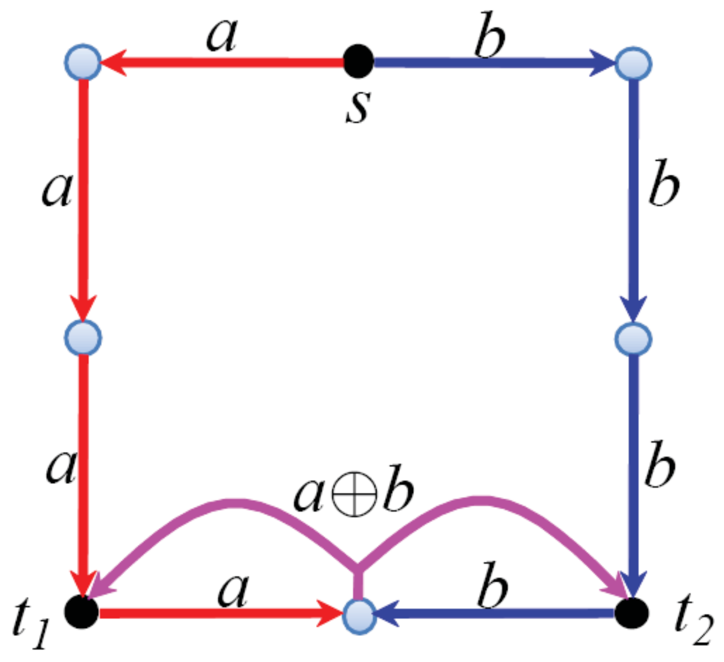
Source
S
$[1 \quad 0] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
$[0 \quad 1] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
A
B
C
$[\alpha_1 \quad \alpha_2] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
D
Receiver 1
F
E
Receiver 2
G
K
$[\alpha_3 + \alpha_1\alpha_4 \quad \alpha_2\alpha_4] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
H
Receiver 3

[Christina Fragouli, EPFL]

Multicast: Saving Transmissions?

- Can we construct examples, where we can save transmissions?
- Yes, for instance with 8 nodes, square topology:



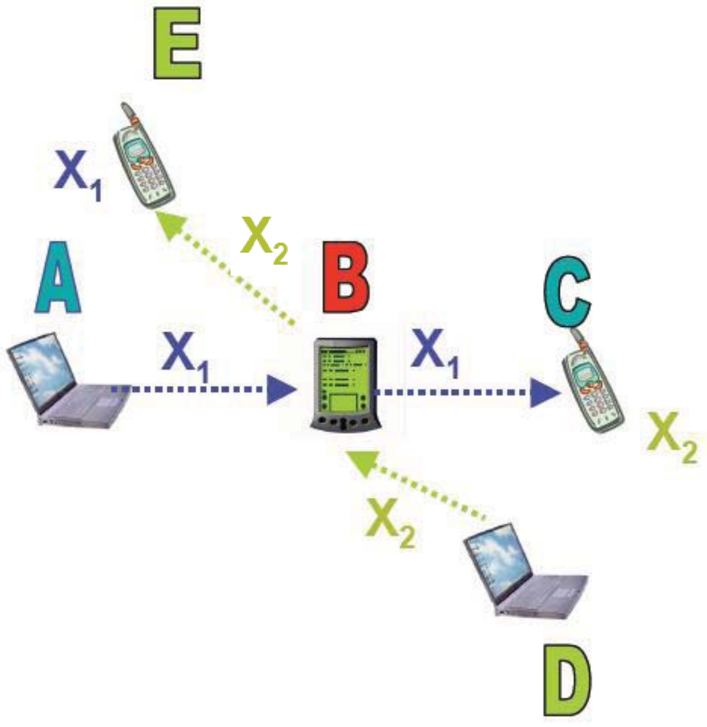


[Yunnan Wu]

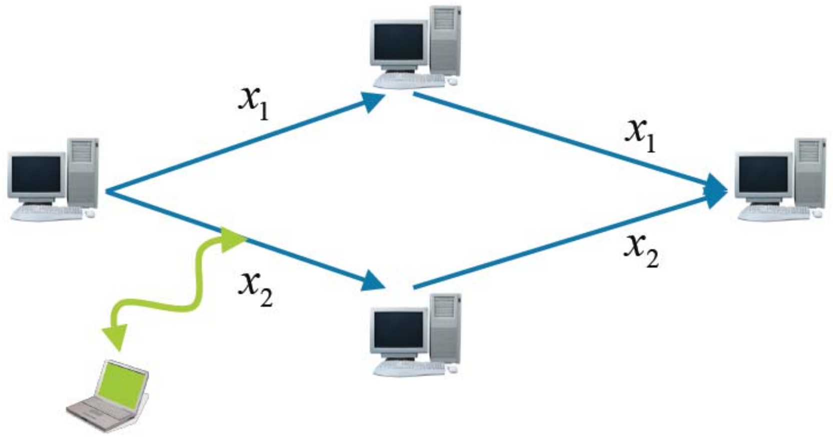
Applications: Network Bottlenecks

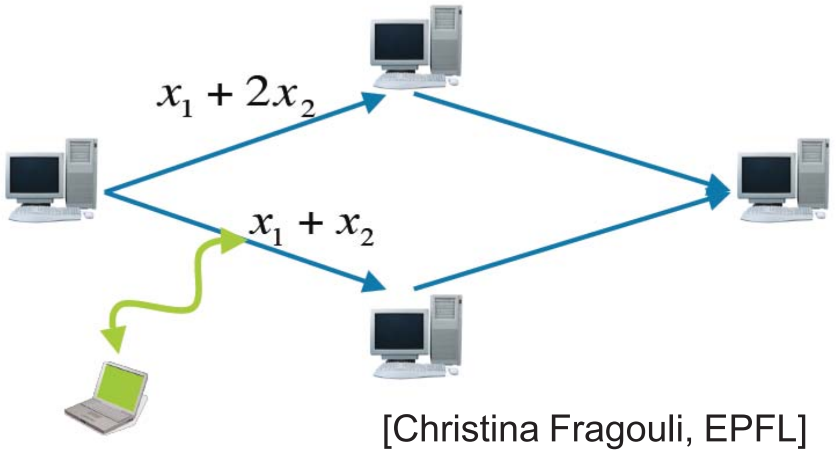
- Node B in the network below is a “bottleneck” because it will need to forward traffic for two flows (A to C and D to E).

- However, thanks to overhearing, it is enough if B transmits the XOR. In this example, all nodes have the same amount of traffic.



Applications: Security



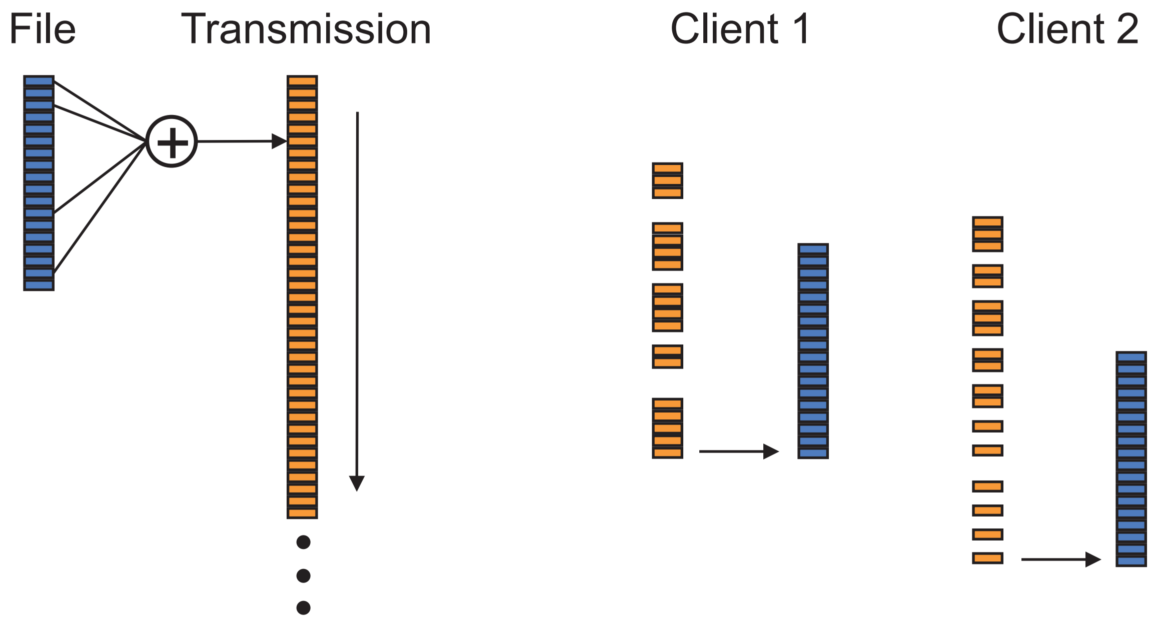


[Christina Fragouli, EPFL]

- Without network coding, an eavesdropper may get half of the information.
- With network coding, getting useful information is harder.

Digital Fountain

- A "digital fountain" streams data continuously and consumers get the full content after a fixed number of received packets.



Digital Fountain Discussion

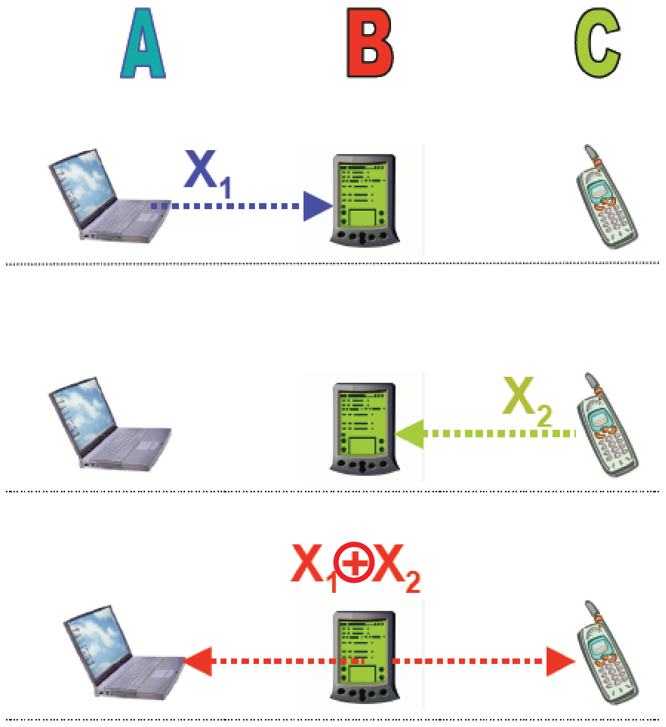
- With the right codes, arbitrary $n + o(n)$ out of n packets are sufficient to reconstruct the complete file.

- The digital fountain idea is slightly older than the other network coding applications, and may be seen as the original work on network coding.
  - However, in both the digital fountain and the security examples, intermediate nodes simply forward the data, without modification. As such, it may be outside the new scope of network coding.

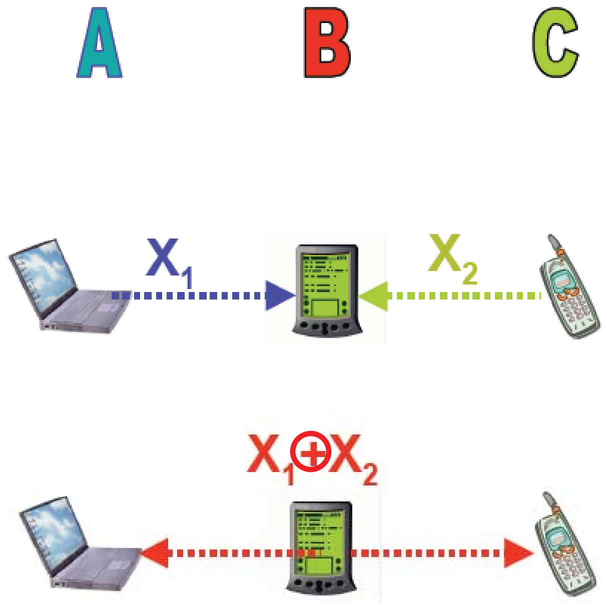
- Digital fountains may also be used to make data more available. Indeed, in peer-to-peer networks, thanks to coding, data may be available long after the source died.

Physical Layer Network Coding

- Remind 3-station example:



- Instead, node B may just repeat the received physical signal, saving one more slot:



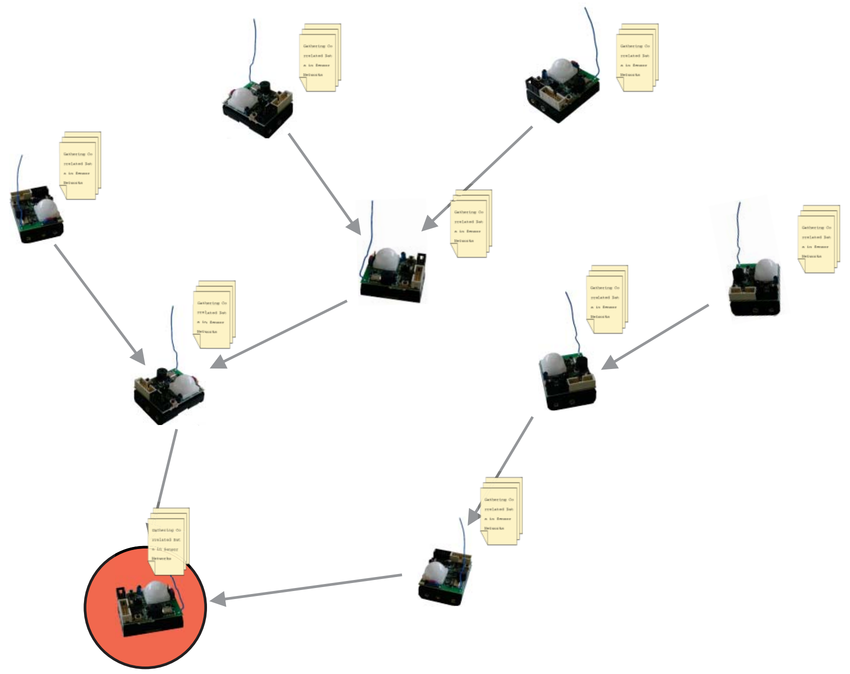
[Christina Fragouli, EPFL]

Case Study: Data Gathering (with Network Coding)

- All nodes produce relevant information about their vicinity periodically.
- Data is conveyed to an information sink for further processing.

➡ Coding scheme

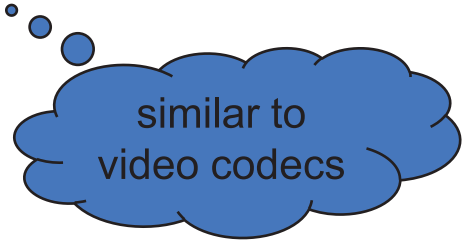
How do we minimize the amount of transmitted data?

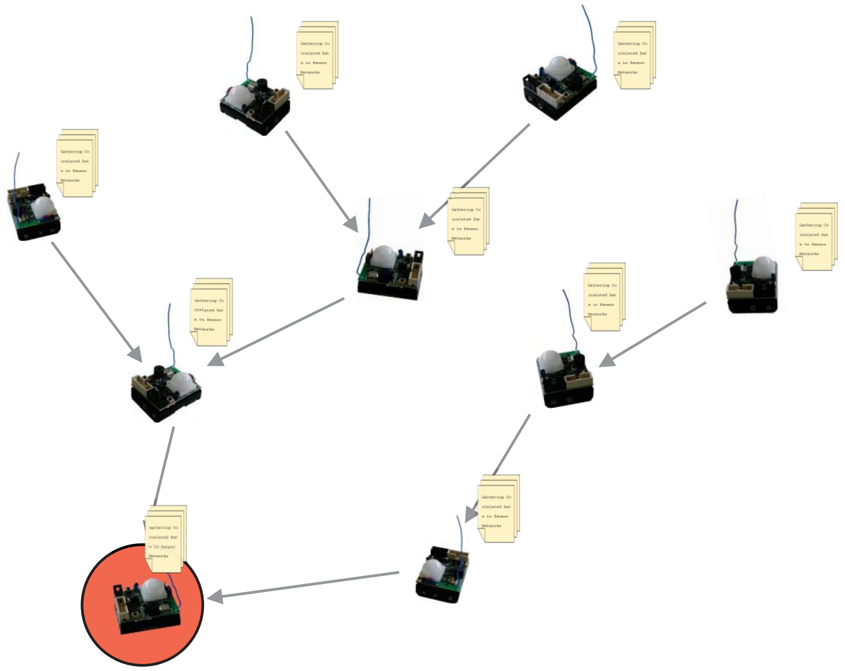


Time coding

- The simplest trick in the book: If the sensed data of a node changes not too often (e.g. temperature), the node only needs to send a new message when its data (significantly) changes.

- Improvement: Only send change of data, not actual data.





Correlated Data

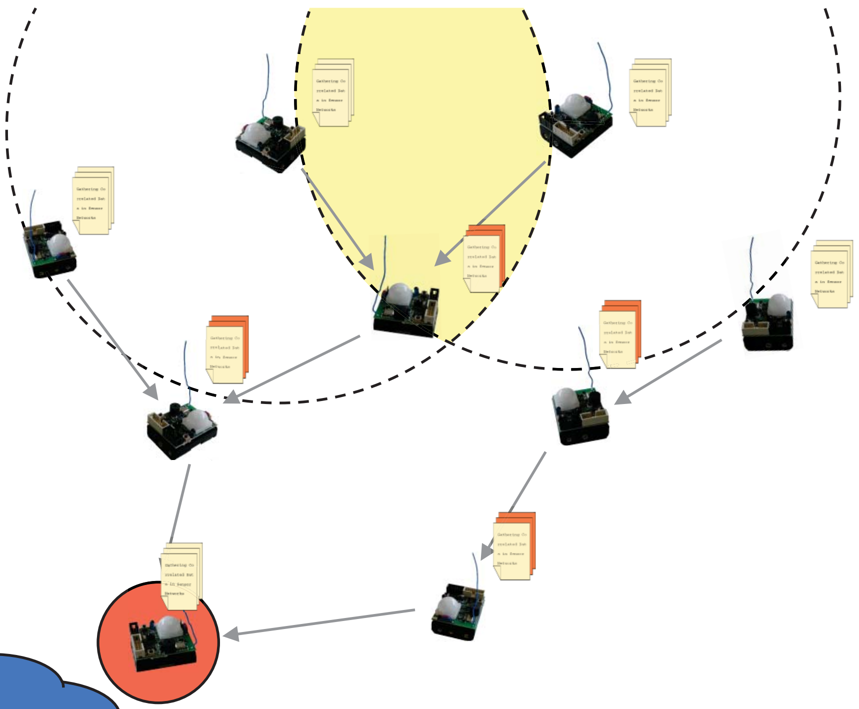
- Different sensor nodes partially monitor the same spatial region.

  ➡ Data correlation

- Data might be processed as it is routed to the information sink.

  ➡ Network coding

At which node is node *u*'s data encoded?



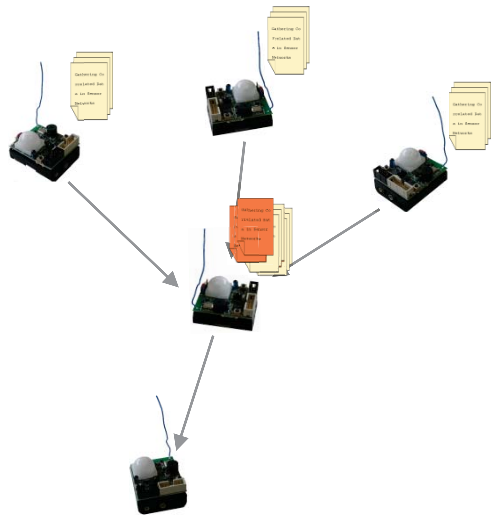
Find a routing scheme and a coding scheme to deliver data packets from all nodes to the sink such that the overall energy consumption is minimal.

Coding strategies

- Multi-input coding
  - Exploit correlation among several nodes.
  - Combined aggregation of all incoming data.

➡ Recoding at intermediate nodes

➡ Synchronous communication model



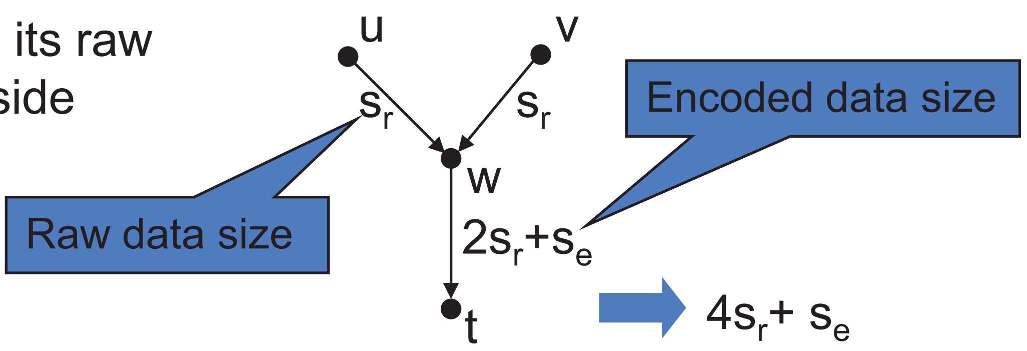
- Single-input coding
  - Encoding of a nodes data only depends on the side information of one other node.

➡ No recoding at intermediate nodes

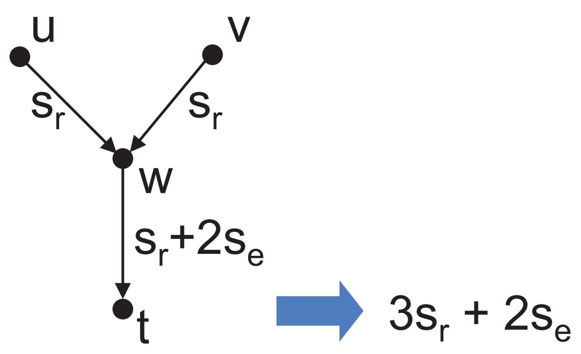
➡ No waiting for belated information at intermediate nodes

Single-input coding

- Self-coding
  - A node can only encode its raw data in the presence of side information.



- Foreign coding
  - A node can use its raw data to encode data it is relaying.



Self-coding

- The cost of an optimal topology

$$c_{opt} = \min \sum_{u \in B} \left(s_r \cdot \mathsf{ST}(S_u, u, t) + \sum_{v \in S_u} s_e \cdot \mathsf{SP}(v, t) \right) .$$

Set of nodes that encode with data from u

Set of nodes with no side information

Steiner tree

Shortest path

- Two ways to lower-bound this equation:
  - $c_{opt} \geq \sum_{u \in V} s_e \cdot \mathsf{SP}(u, t)$
  - $c_{opt} \geq s_r \cdot c(\mathsf{MST})$

Algorithm

- LEGA (Low Energy Gathering Algorithm)
- Based on the shallow light tree (SLT)
- Compute SLT rooted at the sink *t*.
- The sink *t* transmits its packet p_t
- Upon reception of a data packet p_j at node v_i
  - Encode p_i with $p_j \rightarrow p_i^j$
  - Transmit p_i^j to the sink *t*
  - Transmit p_i to all children

Size = s_r

Size = s_e

Excursion: Shallow-Light Tree (SLT)

- Introduced by [Awerbuch, Baratz, Peleg, PODC 1990]
- Improved by [Khuller, Raghavachari, Young, SODA 1993]
  - new name: Light-Approximate-Shortest-Path-Tree (LAST)

- Idea: Construct a spanning tree for a given root r that is both a MST-approximation as well as a SPT-approximation for the root r. In particular, for any $\gamma > 0$
  - $c(\text{SLT}) \leq (1 + \sqrt{2}/\gamma) \cdot c(\text{MST})$
  - $d_{SLT}(v_i, r) \leq (1 + \sqrt{2}\gamma) \cdot \text{SP}(v_i, r)$

- Remember:
  - MST: Easily computable with e.g. Prim's greedy edge picking algorithm
  - SPT: Easily computable with e.g. Dijkstra's shortest path algorithm

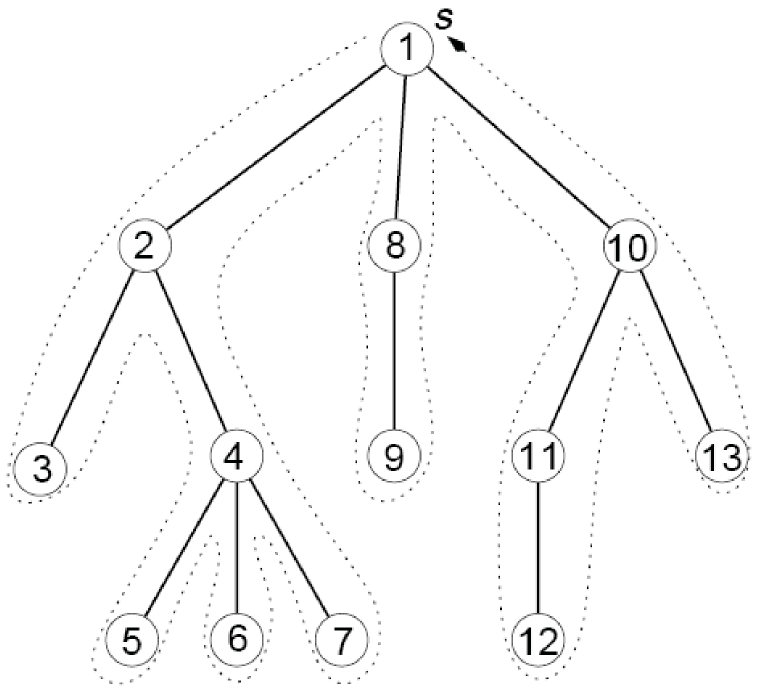
MST vs. SPT

Is a good SPT not automatically a good MST (or vice versa)?

Result & Preordering

- Main Theorem: Given an $\alpha > 1$, the algorithm returns a tree T rooted at r such that all shortest paths from r to u in T have cost at most α the shortest path from r to u in the original graph (for all nodes u). Moreover the total cost of T is at most $\beta = 1+2/(\alpha-1)$ the cost of the MST.

- We need an ingredient:
  A preordering of a rooted tree is generated when ordering the nodes of the tree as visited by a depth-first search algorithm.

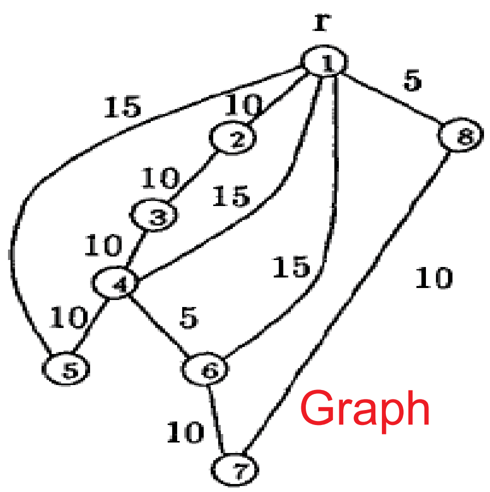


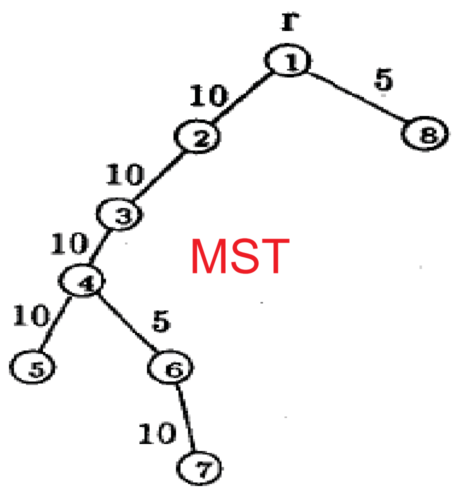
The SLT Algorithm

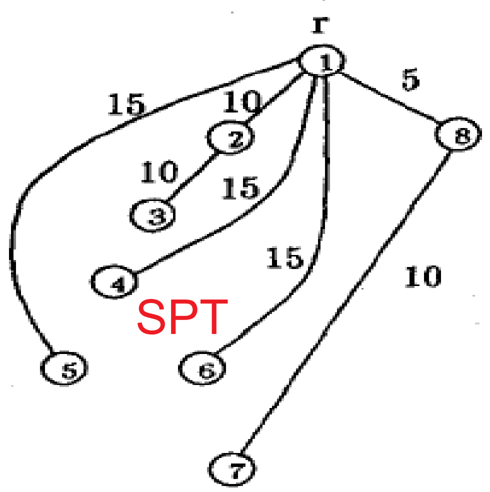
1. Compute MST *H* of Graph *G*;
2. Compute all shortest paths (SPT) from the root *r*.
3. Compute preordering of MST with root *r*.
4. For all nodes *v* in order of their preordering do
    - Compute shortest path from *r* to *v* in *H*. If the cost of this shortest path in *H* is more than a factor α more than the cost of the shortest path in *G*, then just add the shortest path in *G* to *H*.
    - Formally: **IF** $d_H(r,z_i) > \alpha d_G(r,z_i)$ **THEN** $H := H + d_G(r,z_i)$ **ENDIF**.
5. Now simply compute the SPT with root *r* in *H*.

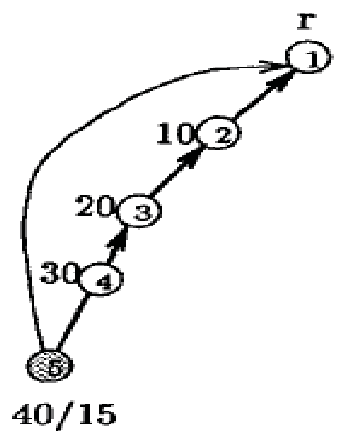
- Sounds crazy… but it works!

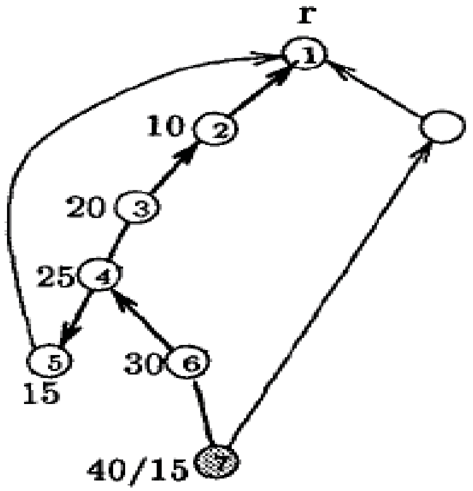
An example, $\alpha = 2$

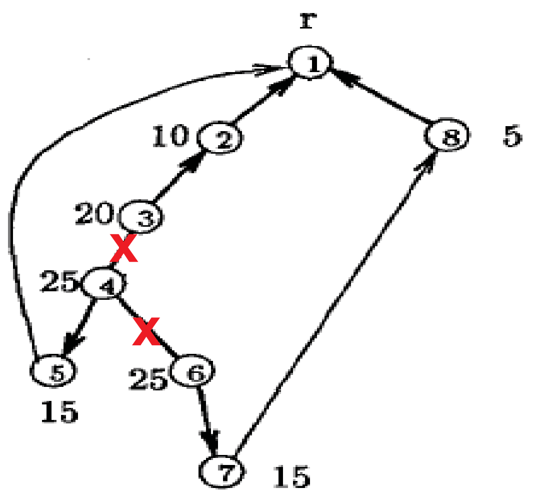












Proof of Main Theorem

- The SPT α-approximation is clearly given since we included all necessary paths during the construction and in step 5 only removed edges which were not in the SPT.

- We need to show that our final tree is a β-approximation of the MST. In fact we show that the graph H before step 5 is already a β-approximation!

- For this we need a little helper lemma first…

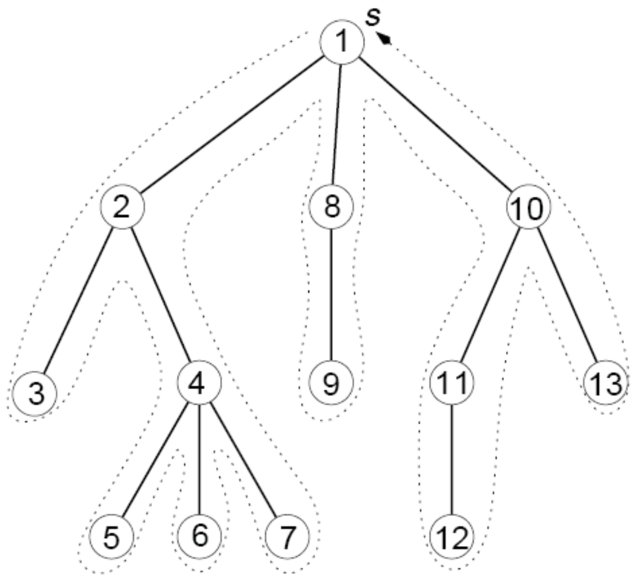
A preordering lemma

- Lemma: Let T be a rooted spanning tree, with root r, and let z_0, z_1, …, z_k be arbitrary nodes of T in preorder. Then,

$$\sum_{i=1}^{k} d_T(z_{i-1}, z_i) \leq 2 \cdot cost(T).$$

- "Proof by picture": Every edge is traversed at most twice.

- Remark: Exactly like the 2-approximation algorithm for metric TSP.



Proof of Main Theorem (2)

- Let $z_1, z_2, \ldots, z_k$ be the set of k nodes for which we added their shortest paths to the root r in the graph in step 4. In addition, let z_0 be the root r. The node z_i can only be in the set if (for example) $d_G(r,z_{i-1}) + d_{MST}(z_{i-1},z_i) > \alpha d_G(r,z_i)$, since the shortest path (r,z_{i-1}) and the path on the MST (z_{i-1},z_i) are already in H when we study z_i.

- We can rewrite this as $\alpha d_G(r,z_i) - d_G(r,z_{i-1}) < d_{MST}(z_{i-1},z_i)$. Summing up:

$$\begin{array}{lcll}
\alpha d_G(r,z_1) - \cancel{d_G(r,z_0)} & < & d_{MST}(z_0,z_1) & (i=1) \\
\alpha d_G(r,z_2) - d_G(r,z_1) & < & d_{MST}(z_1,z_2) & (i=2) \\
\ldots & \ldots & \ldots & \\
\alpha d_G(r,z_k) - d_G(r,z_{k-1}) & < & d_{MST}(z_{k-1},z_k) & (i=k) \\
\hline
\Sigma_{i=1\ldots k}(\alpha-1)\, d_G(r,z_i) \quad + \quad \alpha \cancel{d_G(r,z_k)} & < & \Sigma_{i=1\ldots k}\, d_{MST}(z_{i-1},z_i) &
\end{array}$$

Proof of Main Theorem (3)

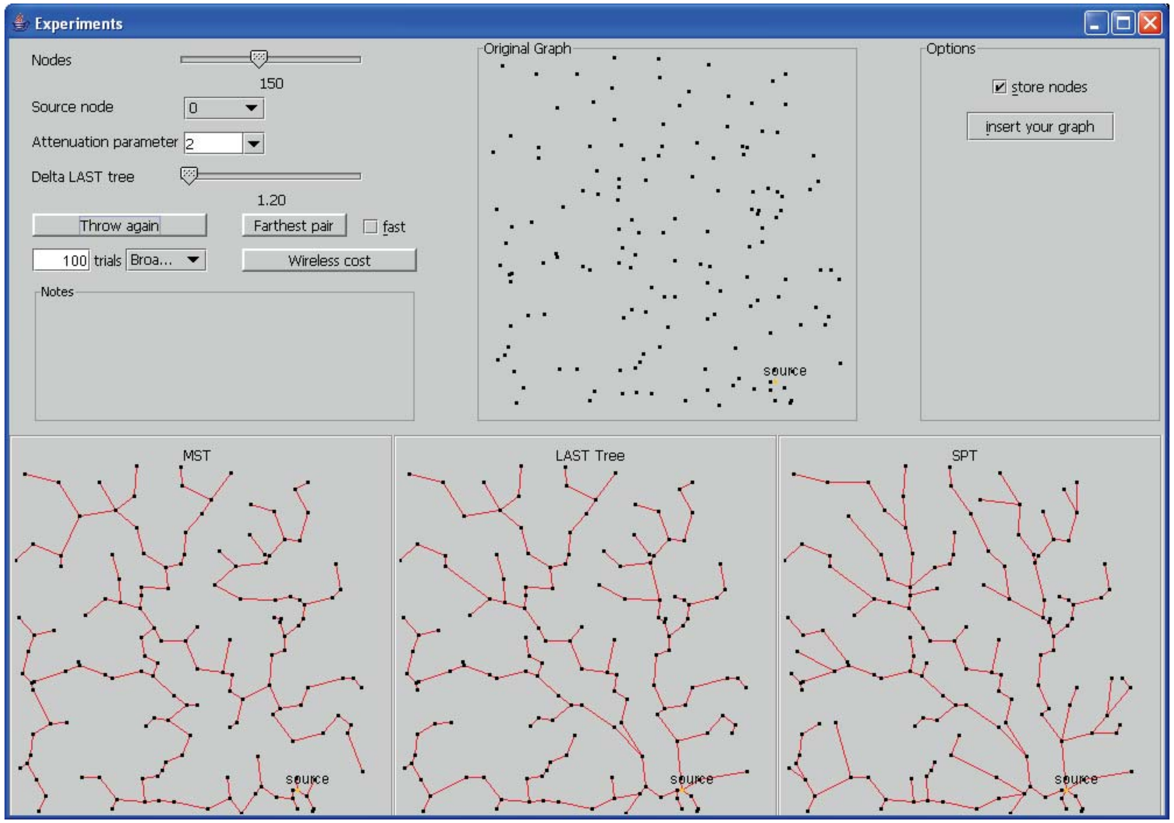
- Simplifying a bit: $(\alpha-1) \Sigma_{i=1\ldots k} d_G(r,z_i) < \Sigma_{i=1\ldots k} d_{MST}(z_{i-1},z_i)$
- All we did in our construction of H was to add exactly at most the cost $\Sigma_{i=1\ldots k} d_G(r,z_i)$ to the cost of the MST. In other words, $\text{cost}(H) \leq \text{cost}(MST) + \Sigma_{i=1\ldots k} d_G(r,z_i)$.
- Using the inequality at the top of this slide we have $\text{cost}(H) < \text{cost}(MST) + 1/(\alpha-1) \Sigma_{i=1\ldots k} d_{MST}(z_{i-1},z_i)$.
- Using our preordering lemma we have $\text{cost}(H) \leq \text{cost}(MST) + 1/(\alpha-1)\, 2\text{cost}(MST) = 1+2/(\alpha-1)\, \text{cost}(MST)$
- That's exactly what we needed: $\beta = 1+2/(\alpha-1)$.

How the SLT can be used

- The SLT has many applications in communication networks.

- Essentially, it bounds the cost of unicasting (using the SPT) and broadcasting (using the MST).

- Remark: If you use $\alpha = 1+\sqrt{2}$, then $\beta = 1+2/(\alpha-1) = \alpha$.


Experiments

Nodes 150

Source node 0

Attenuation parameter 2

Delta LAST tree 1.20

100 trials Broa...

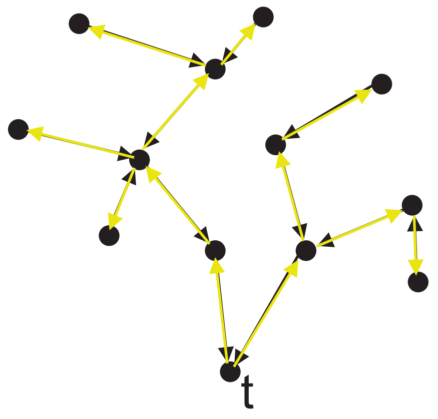
☐ fast

☑ store nodes

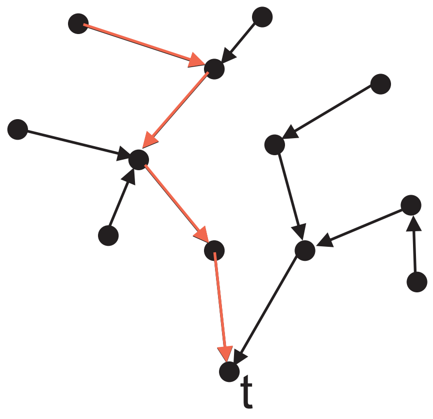
[www.dia.unisa.it/~ventre]

Analysis of LEGA

Theorem: LEGA achieves a $2(1+\sqrt{2})$-approximation of the optimal topology. (We use $\alpha = 1+\sqrt{2}$.)



$\Rightarrow s_r \cdot c(\mathsf{SLT})$



$\Rightarrow \sum_{v_i \in V} s_e \cdot d_{SLT}(v_i, t)$

$$\begin{aligned} c_{LEGA} &\le s_r \cdot (1+\sqrt{2})c(\mathsf{MST}) + (1+\sqrt{2}) \sum_{v_i \in V} s_e \cdot \mathsf{SP}(v_i, t) \\ &\le 2(1+\sqrt{2})c_{opt} \end{aligned}$$

Slide 5/25

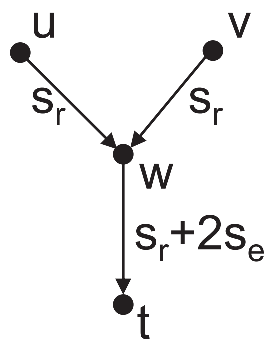
Foreign coding

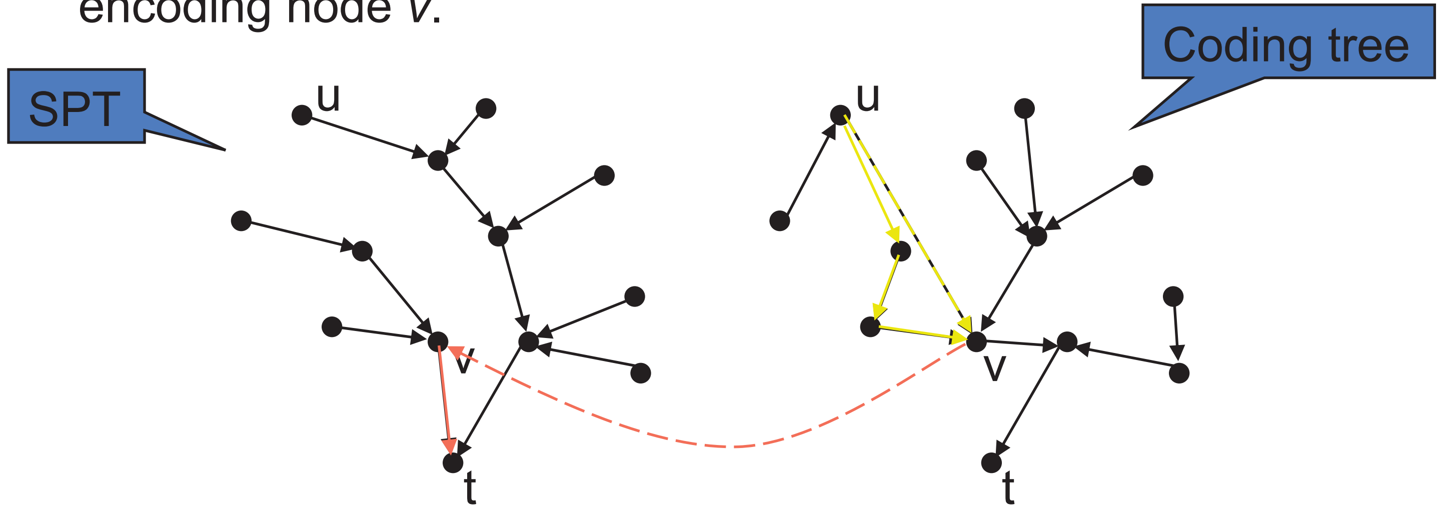
- MEGA (Minimum-Energy Gathering Algorithm)
  - Superposition of two tree constructions.

- Compute the shortest path tree (SPT) rooted at *t*.

- Compute a coding tree.
  - Determine for each node *u* a corresponding encoding node *v*.

Encoding must not result in cyclic dependencies.





Coding tree construction

- Build complete directed graph
- Weight of an edge *e*=(v_i,v_j):

Cost from v_j to the sink t.

$$w(e) = s_i \cdot \mathsf{SP}(v_i, v_j) + s_i^j \cdot \mathsf{SP}(v_j, t)$$

Cost from v_i to the encoding node v_j.

Number of bits when encoding v_i's info at v_j

- Compute a directed minimum spanning tree (arborescence) of this graph. (This is not trivial, but possible.)

Theorem: MEGA computes a minimum-energy data gathering topology for the given network.

All costs are summarized in the edge weights of the directed graph.

Summary

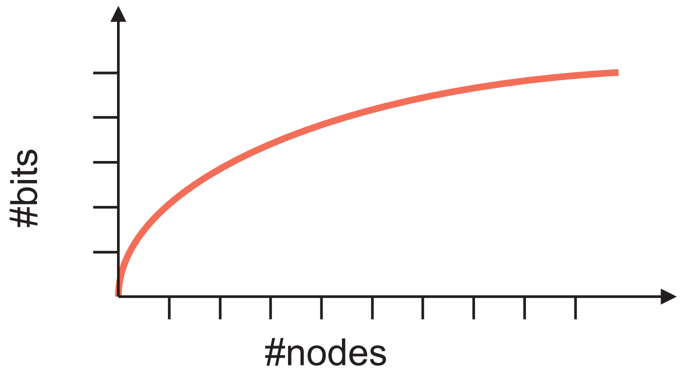
- Self-coding:
  - The problem is NP-hard [Cristescu et al, INFOCOM 2004]
  - LEGA uses the SLT and gives a $2(1+\sqrt{2})$-approximation.
  - Attention: We assumed that the raw data resp. the encoded data always needs s_r resp. s_e bits (no matter how far the encoding data is!). This is quite unrealistic as correlation is usually regional.

- Foreign coding
  - The problem can be solved optimally, with MEGA.

- What if we allow both coding strategies at the same time?
- What about a more accurate correlation model?
- What if multi-coding is allowed?

Multicoding

- Hierarchical matching algorithm [Goel & Estrin SODA 2003].
- We assume to have concave, non-decreasing aggregation functions. That is, to transmit data from k sources, we need f(k) bits with $f(0)=0$, $f(k) \geq f(k-1)$, and $f(k+1)/f(k) \leq f(k)/f(k-1)$.



- The nodes of the network must be a metric space*, that is, the cost of sending a bit over edge (u,v) is c(u,v), with
  - Non-negativity: $c(u,v) \geq 0$
  - Zero distance: $c(u,u) = 0$ (*we don't need the identity of indescernibles)
  - Symmetry: $c(u,v) = c(v,u)$
  - Triangle inequality: $c(u,w) \leq c(u,v) + c(v,w)$

The algorithm

- Remark: If the network is not a complete graph, or does not obey the triangle inequality, we only need to use the cost of the shortest path as the distance function, and we are fine.

- Let S be the set of source nodes. Assume that S is a power of 2. (If not, simply add copies of the sink node until you hit the power of 2.) Now do the following:

1. Find a min-cost perfect matching in S.
2. For each of the matching edges, remove one of the two nodes from S (throw a regular coin to choose which node).
3. If the set S still has more than one node, go back to step 1. Else connect the last remaining node with the sink.

The result

- Theorem: For any concave, non-decreasing aggregation function f, and for [optimal] total cost C[*], the hierarchical matching algorithm guarantees

$$E\left[\max_{\forall f} \frac{C(f)}{C^*(f)}\right] \leq 1 + \log k$$

- That is, the expectation of the worst cost overhead is logarithmically bounded by the number of sources.

Remarks

- For specific concave, non-decreasing aggregation functions, there are simpler solutions.
  - For f(x) = x the SPT is optimal.
  - For f(x) = const (with the exception of f(0) = 0), the MST is optimal.
  - For anything in between it seems that the SLT again is a good choice.
  - For any a priori known f one can use a deterministic solution by [Chekuri, Khanna, and Naor, SODA 2001]
  - If we only need to minimize the maximum expected ratio (instead of the expected maximum ratio), [Awerbuch and Azar, FOCS 1997] show how it works.

- Again, sources are considered to aggregate equally well with other sources. A correlation model is needed to resemble the reality better.

Other work using coding

- LEACH [Heinzelman et al. HICSS 2000]: randomized clustering with data aggregation at the clusterheads.
  - Heuristic and simulation only.
  - For provably good clustering, see chapter on clustering.

- Correlated data gathering [Cristescu et al. INFOCOM 2004]:
  - Coding with Slepian-Wolf
  - Distance independent correlation among nodes.
  - Encoding only at the producing node in presence of side information.
  - Same model as LEGA: NP-hardness proof.

Open problem

- Future applications incorporating network coding may not try to optimize network throughput but utilize other side effects. In peer-to-peer networks for example, network coding is used to increase the longevity of a file inside the network.
  - Concretely, so far peers store pieces of a file. If all peers storing a certain piece leave the network, the file cannot be reconstructed anymore. Instead, if peers store combinations of pieces, the file will be more available.

- Goal: Find a new application that exploits the reliability aspect of network coding.