
Network Coding

Jie Gao

Existing network

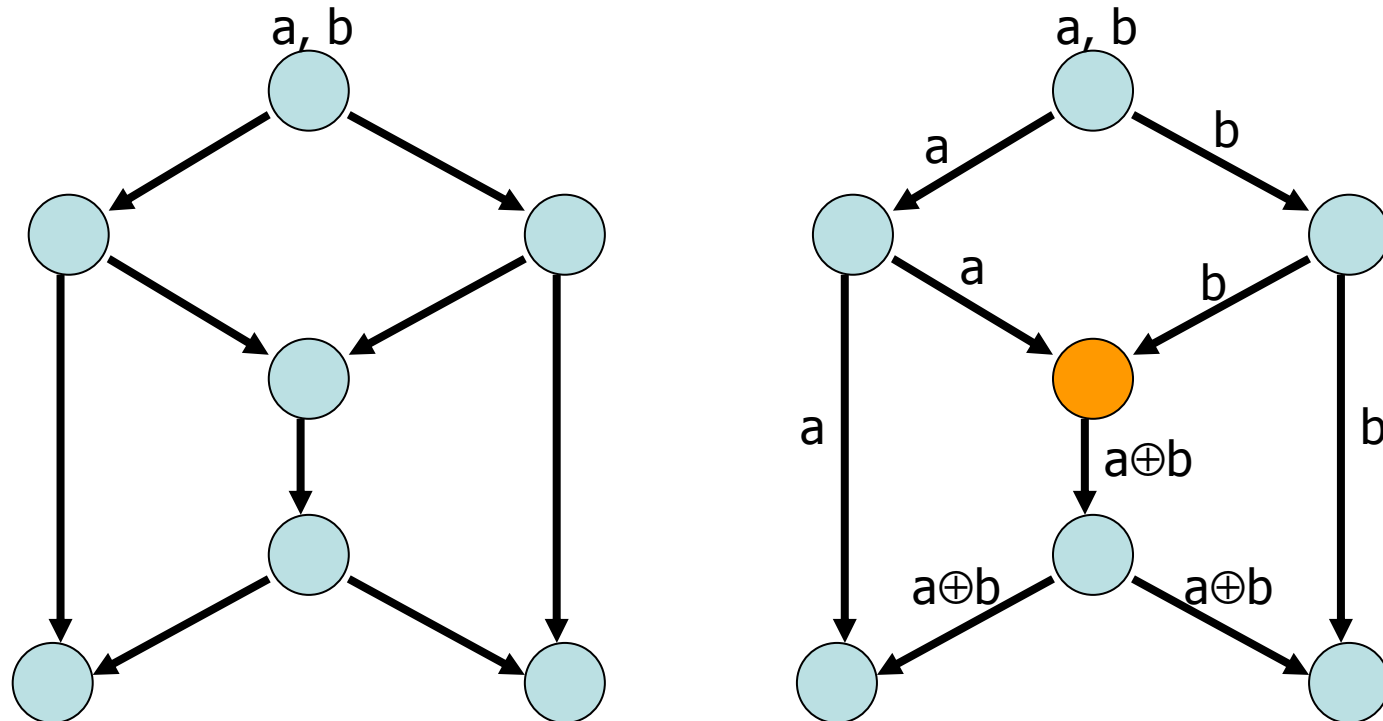
- Independent data stream sharing the same network resources
 - Packets over the Internet
 - Signals in a phone network
 - An analog: cars sharing a highway.
- Information flows are separated.
- What about we mix them?

Why do we want to mix information flows?

- The core notion of network coding is to allow and encourage mixing of data at intermediate network nodes.
- R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", *IEEE Transactions on Information Theory*, IT-46, pp. 1204-1216, 2000.

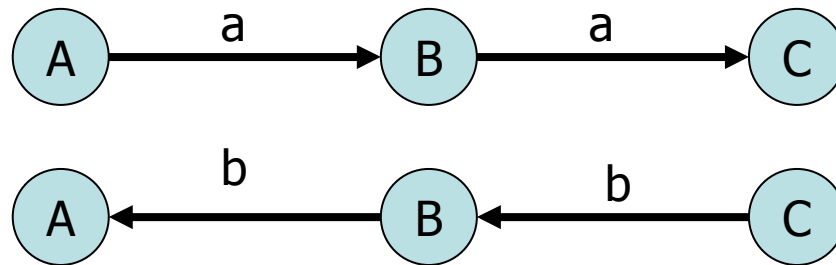
Network coding increases throughput

- Butterfly network
- Multi-cast: throughput increases from 1 to 2.

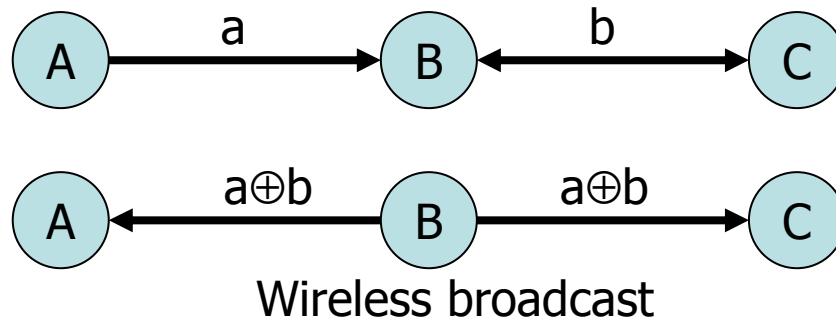


Network coding saves energy & delay in wireless networks

- A wants to send packet a to C.
- C wants to send packet b to A.



- B performs coding



Linear coding is enough

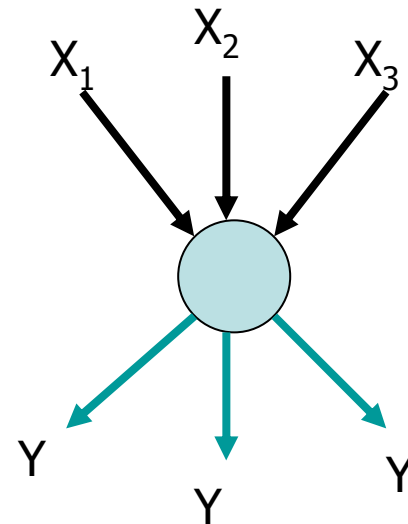
- Linear code: basically take linear combinations of input packets.
 - Not concatenation!
 - $3a+5b$: has the same length as a , b .
 - $+$ is xor in a field of 2.
- Even better: **random** linear coding is enough.
 - Choose coding coefficients randomly.

Encode

- Original packets: M_1, M_2, \dots, M_n .
- An incoming packet is a linear combination of the original packets
- $X = g_1 M_1 + g_2 M_2 + \dots + g_n M_n$.
- $g = (g_1, g_2, \dots, g_n)$ is the **encoding vector**.
- Encoding can be done recursively.

An example

- At each node: do linear encoding of the incoming packets.
- $Y = h_1 X_1 + h_2 X_2 + h_3 X_3$
- Encoding vector is attached with the packet.



Decode

- To recover the original packets M_1, M_2, \dots, M_n .
- Receive m (scrambled) packets.
- How to recover the n unknowns?
 - First, $m \geq n$.
 - The good thing is, $m=n$ is sufficient.
- Received packets: Y_1, Y_2, \dots, Y_n .

Coding scheme

- To decode, we have the linear system:
- $Y_i = a_{i1}M_1 + a_{i2}M_2 + \dots + a_{in}M_n$
- As long as the coefficients are independent \rightarrow we can solve the linear system.

- Theorem: (1) There is a deterministic encoding algorithm; (2) Random linear coding is good, with high probability.

Practical considerations (1)

- Decoding: receiver keeps a **decoding matrix** recording the packets it received so far.
- When a new packet comes in, its **coding vector** is inserted at the bottom of the matrix, then perform Gaussian elimination.
- When the matrix is solvable, we are done.

Practical consideration (2)

- **Control the decoding effort** (size of the matrix).
- Group packets into **generations**. Packets in the same generation are encoded.
- **Delay**: in practice typically not much higher.

Implication of network coding

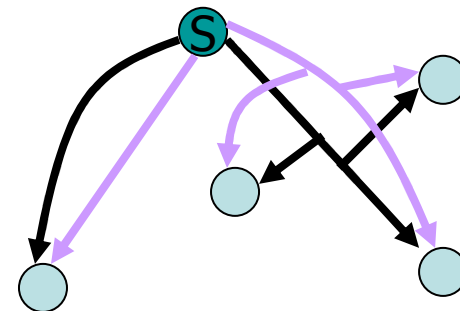
- Successful reception of information
 - **does not** depend on the receiving packet content.
 - But rather depend on receiving a **sufficient** number of independent packets.

What next?

- Benefits of network coding
- Applications of network coding

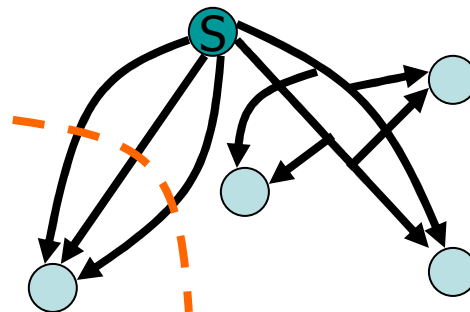
Throughput (capacity) with coding

- Multi-cast problem: one source, N receivers in a **directed** network. Each edge has a maximum capacity.
- Without coding, the maximum throughput routing is NP-hard.
- Proof: reduction from Steiner tree packing.
- Nodes **share** network resources.



Throughput (capacity) with coding

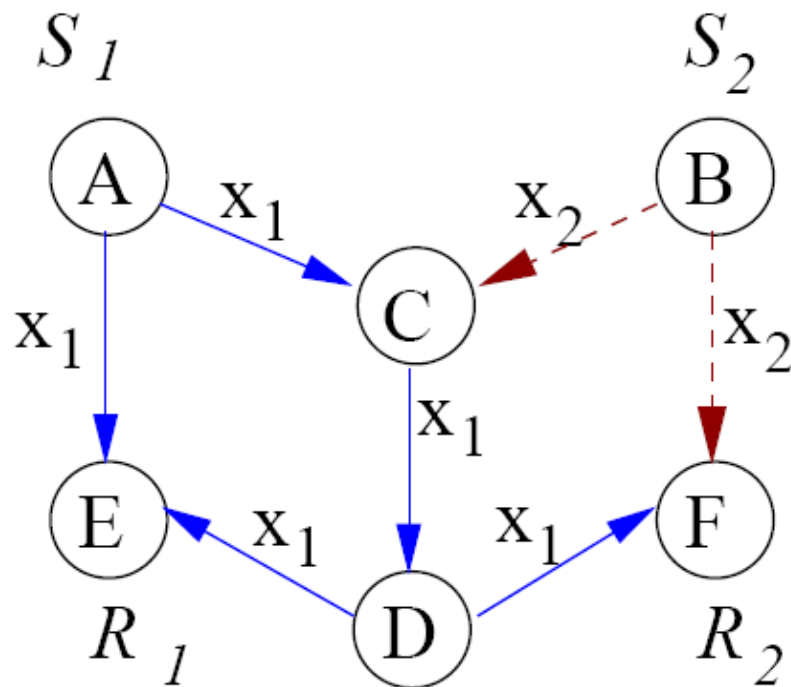
- With coding each destination can ignore the other destinations.
- Receiving data rate = min-cut.
- As if the user is using the entire network by itself.
- Offer throughput benefit for unicast as well.



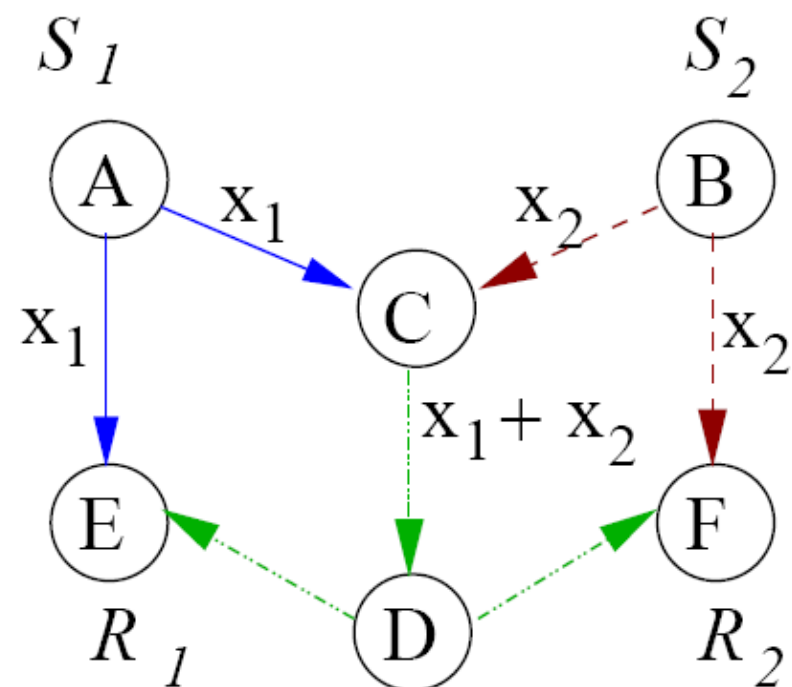
Example: butterfly network

- 2 multi-cast flow. Data rate=2.

Traditional Method



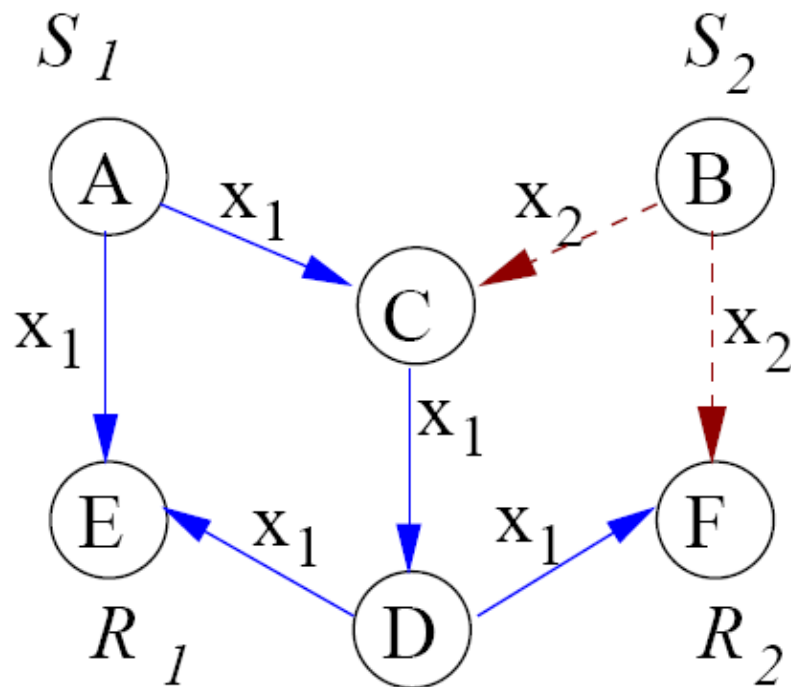
Network Coding



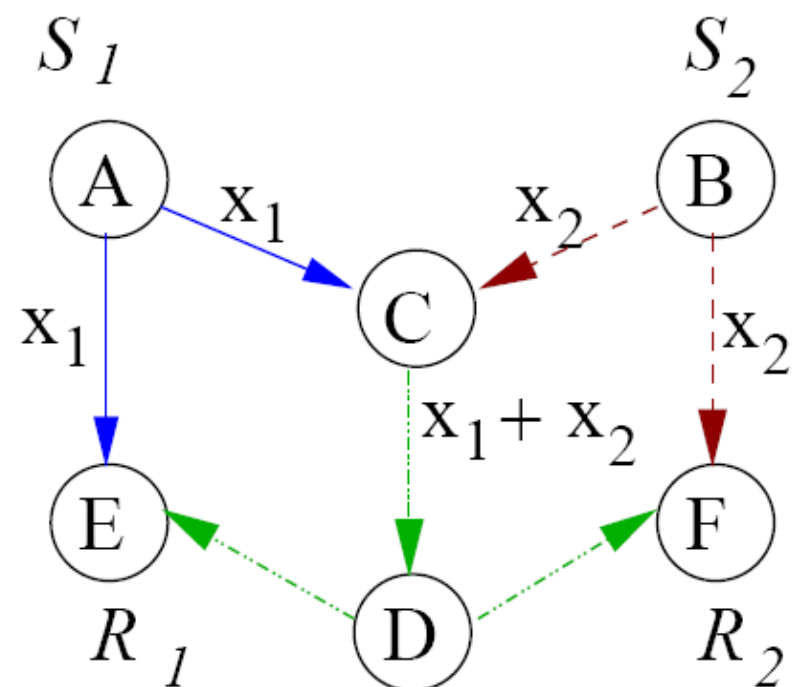
Example: butterfly network

- $S_1 \rightarrow R_2$, $S_2 \rightarrow R_1$. Data rate=1.

Traditional Method



Network Coding

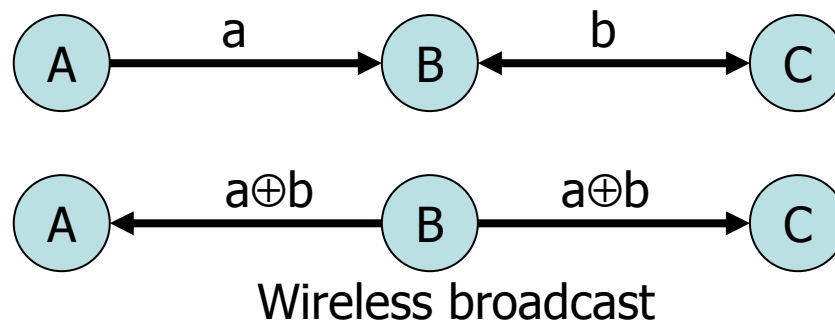


Summary on throughput gain

- In **directed** graph, the throughput gain by network coding can be **arbitrarily large**.
- In **undirected** graph, the throughput gain is at most 2.
- Without coding, the max throughput routing is NP-hard.
- With coding, the max throughput coding is achievable by linear programming.
 - Just decide on the rate of each edge.
 - Ignore the content.

Robustness and stability

- Each encoded packet is “equally important”
→ opportunistic routing.
- A, C may go sleep randomly without telling B. B sends $a \oplus b$, so whoever wakes up can get new information.



Application in gossip algorithm

- Assume n nodes each holding a packet, all of them want all packets.
- Gossip algorithm: in each round each node picks **randomly** another node and exchange **1** message.
- Question: what is the number of rounds (in expectation)?
- Answer: $O(n \log n)$.
- Why? --- coupon collection problem, once again

Gossip with coding

- Aka. algebraic gossip.
- Each node encodes with random linear combination of incoming (received) messages.
- $O(n)$ is enough with high probability.
- This is optimal.

Another example: packet erasure networks

- Want 2 things: low delay and high rate.
- But, packets get dropped in the middle.
- Two main approaches in the literature:
 - Repeat: low delay, **low rate**.
 - Error correction: max rate, **high** delay.
- With coding, we can achieve OPT rate & delay.

An example

- Error rate $e(AB)$, $e(BC)$: probability that packet is dropped on the link.



- Approach 1:

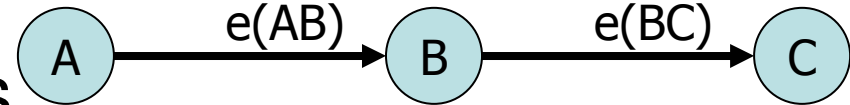
- Send data packets blindly with max sending rate.
- End-to-end rate= $\{1-e(AB)\}\{1-e(BC)\}$.

- Approach 2:

- Do error checking at B. Make sure B recovers the message (and then send to C). Higher delay.
- End-to-end rate= $\min\{1-e(AB), 1-e(BC)\}$. Optimal.

An example

- Error rate $e(AB)$, $e(BC)$: probability that packet is dropped on the link.



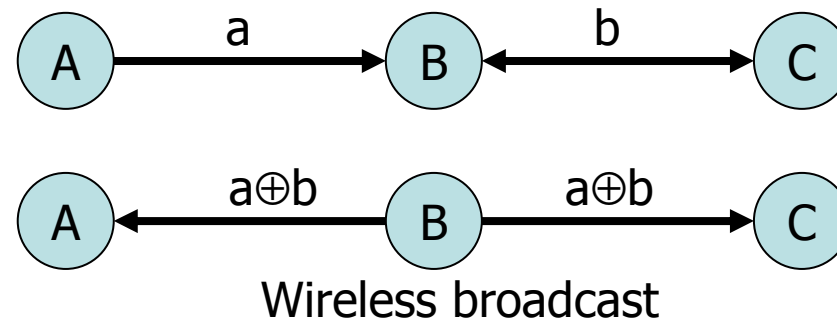
- Send a stream of packets
- With network coding:
 - Node B forms random linear combinations of encoded packets it received so far.
 - Achieves the optimal rate $=\min\{1-e(AB), 1-e(BC)\}$.
 - With no delay --- node B does not have to wait.

Applications of network coding: P2P

- **Avalanche** (<http://research.microsoft.com/~pablo/avalanche.aspx>)
- **BitTorrent-style P2P sharing with network coding.**
 - Big file gets chopped into small pieces.
 - Randomly coded.
 - Participants share their **coded** pieces.
- **Why use coding?**
 - Topology of the P2P users is hard to know.
 - Optimal packet scheduling for large files is difficult.
 - Robustness to user join/leave.
 - Easy to incorporate incentive mechanisms (prevent free-riding).

Wireless networks

- Wireless links are broadcast nature.



- Bidirectional traffic for a path:
 - First alternate for two directions.
 - Then use coding/wireless broadcast.
 - Double the capacity.

A perfect match: Wireless networks + network coding

- Wireless channels are lousy. Make use of overhearing for opportunistic routing.
 - Residential wireless mesh network
 - Many-to-many broadcast (or gossip algorithm with broadcast).
- Network coding is good with
 - Large network
 - No global topology information
 - Unreliable links/nodes.

Sensor networks + network coding

- Radio
 - Tuning them to the same channel is energy costly.
 - Channel assignment to maximize throughput is highly non-trivial.
- Untuned (non-calibrated) radios
 - Two devices may not be able to communicate.
 - With a group of them, the chance that there exists two with the same channel is high. (Birthday paradox)
 - But we don't know which pair can communicate.
 - Send coded packets blindly.
 - Even multi-hop works (without discovering the path).

Birthday paradox

- In a room of n people, Prob{No two people have the same birthday}.

$$\bar{p}(n) = 1 \cdot \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{n-1}{365}\right) = \frac{365 \cdot 364 \cdots (365 - n + 1)}{365^n} = \frac{365!}{365^n (365 - n)!}$$

$$\bar{p}(n) \approx 1 \cdot e^{-1/365} \cdot e^{-2/365} \cdots e^{-(n-1)/365}$$

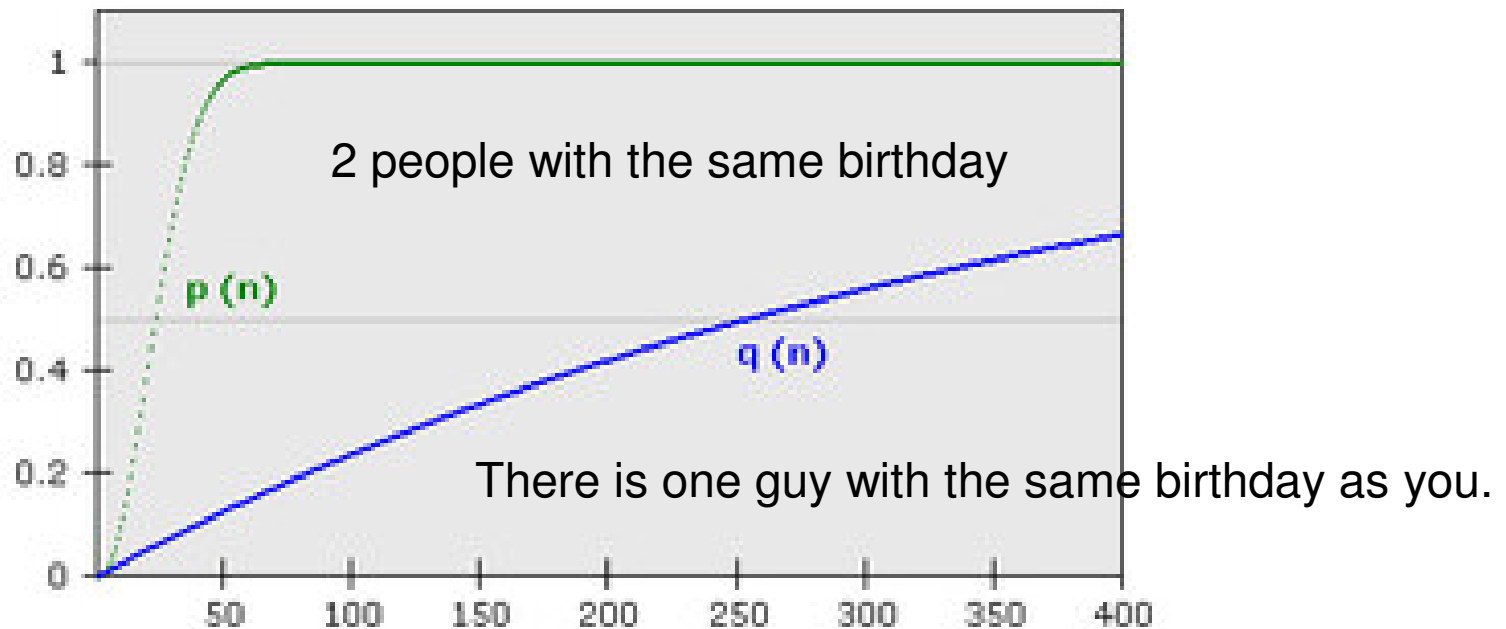
$$= 1 \cdot e^{-(1+2+\cdots+(n-1))/365}$$

$$= e^{-(n(n-1))/2 \cdot 365} \quad e^x = 1 + x + \frac{x^2}{2!} + \cdots$$

$$p(n) = 1 - \bar{p}(n) \approx 1 - e^{-(n(n-1))/2 \cdot 365}$$

Birthday paradox

$$p(n) = 1 - \bar{p}(n) \approx 1 - e^{-(n(n-1))/2 \cdot 365}$$



$$q(n) = 1 - \left(\frac{365 - 1}{365}\right)^n$$

Network Tomography

- Network diagnosis of loss rate of links.
- In a multi-cast tree, the receivers that miss the same packet can derive the failed link.
- With coding one can get more detailed information about the failure links from the **pattern of the received codes**.
 - Active diagnosis
 - Passive network monitoring.

Security

- Information gets smashed.
- Protection from eavesdroppers.
 - It is difficult to interpret and short-term overhear does not work.
- Packet modification is harder too.
 - Need to fake data that makes sense
 - Challenge: no idea about the original data packets.
- Jamming
 - Less of a problem. Jamming a few packets does not affect a large set of data packets much.

Summary

- Network coding is good for
 - Scalability
 - Limited topological information
 - Highly dynamic network
- Key insights
 - Treat the packets equally
 - No need to read the content, just do counting
 - Anything helps.
 - Don't need to know what is where.

Discussions

- More application scenarios?
 - Vehicle network (taxi network)
 - Moving vehicle with access points in proximity.
- New ideas?
- When network coding is **NOT** appropriate?

References

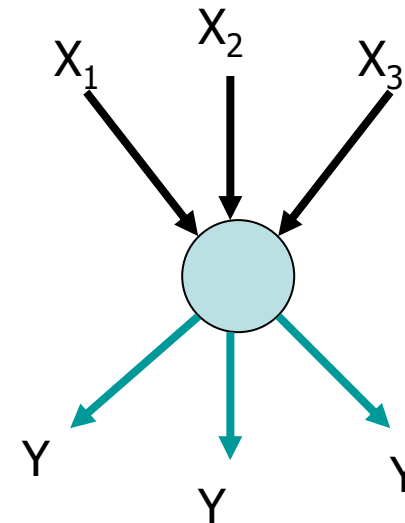
- A 6-page network coding introduction.
C. Fragouli, J. Le Boudec, Jorg Widmer,
[Network coding: an instant primer.](#)
- Network coding webpage:
<http://tesla.csl.uiuc.edu/~koetter/NWC/>
- A book: “Network coding theory”.

Presentations 11/28 (Tuesday next week)

- **[Katti06]** S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, J. Crowcroft, [XORs in The Air: Practical Wireless Network Coding](#), Sigcomm'06.
- **[Zhang06]** Shengli Zhang, Soung Chang Liew, Patrick Lam, [Hot Topic: Physical-Layer Network Coding](#), Mobicom'06.

Network coding

- The core notion of network coding is to allow and encourage mixing of data at intermediate network nodes.
- Each internal node performs **random linear coding**.
- $Y_i = h_1 X_1 + h_2 X_2 + h_3 X_3$

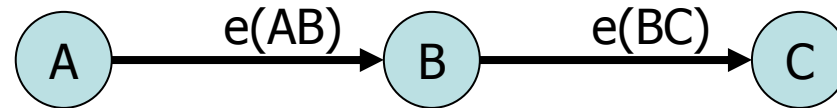


Packet erasure networks

- Model packet loss
 - Congestion, buffer overflow, fading in wireless channels.
- 1. End-to-end acknowledgement & Retransmission (TCP style)
 - Retransmissions use up resources.
 - Multicast: possibly only a subset of nodes need retransmission.
- 2. Erasure error correction codes
 - First code the original k data items into n pieces.
 - Recover the original data from any k pieces.

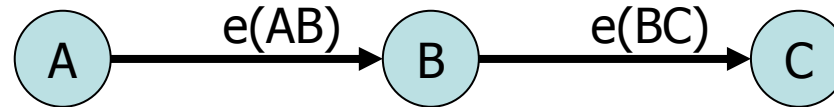
3-node example

- Erasure channel: with probability $e(AB)$, $e(BC)$ packet **disappears** on the link.



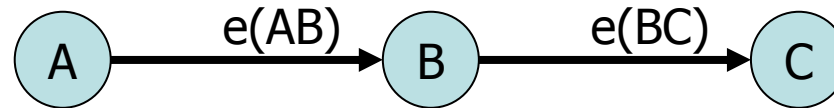
- A retransmit, B simply forward.
 - Throughput: # (re)-transmissions for a packet to reach C = $1/[(1-e(AB))(1-e(BC))]$.
 - Why?

3-node example



- Erasure coding on each link **separately**.
 - Node A encode k data into $k/(1-e(AB))$ pieces.
 - Roughly B gets k data pieces, reconstruct.
 - Do the same at link BC.
 - **Extra delay** for reconstruction at B.
 - # transmissions for one packet to reach C is $\max(1/(1-e(AB)), (1-e(BC)))$

3-node example: why coding helps



- Node B does not bother decode or reconstruct, instead, send random linear combination of what is received
 - No delay for decoding in the middle.
 - Node A encode k data into $k/(1-e(AB))$ pieces.
 - Roughly B gets k data pieces, recover the original.
 - B again boost up to $k/(1-e(BC))$ pieces.
 - C is able to reconstruct.

Network Tomography

- Network diagnosis of loss rate of links.
- In a multi-cast tree, the receivers that miss the same packet can derive the failed link.
- With coding one can get more detailed information about the failure links from the [pattern of the received codes](#).
 - Active diagnosis
 - Passive network monitoring.

Security

- Information gets smashed.
- Protection from eavesdroppers.
 - It is difficult to interpret and short-term overhear does not work.
- Packet modification is harder too.
 - Need to fake data that makes sense
 - Challenge: no idea about the original data packets.
- Jamming
 - Less of a problem. Jamming a few packets does not affect a large set of data packets much.

Summary

- Network coding is good for
 - Scalability
 - Limited topological information
 - Highly dynamic network
- Key insights
 - Treat the packets equally
 - No need to read the content, just do counting
 - Anything helps.
 - Don't need to know what is where.

Discussions

- More application scenarios?
 - Vehicle network (taxi network)
 - Moving vehicle with access points in proximity.
- New ideas?
- When network coding is **NOT** appropriate?