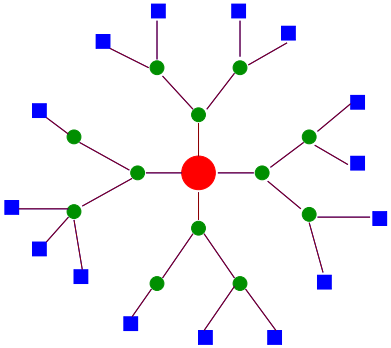
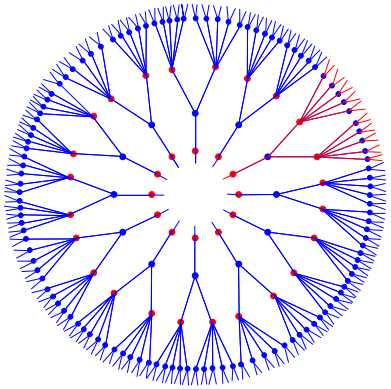
Efficient Content Delivery and Low Complexity Codes





Amin Shokrollahi

digitalfountain

Content

- Goals and Problems
- TCP/IP, Unicast, and Multicast
- Solutions based on codes
- Applications

Goal

Want to transport data from a transmitter to one or more receivers over IP reliably.

Current solutions have limitations when amount of data is large and

- Number of receivers is large (point-multipoint transmission)
  - Video on Demand, new versions of computer games
- Or, network suffers from unpredictable and transient losses
  - Satellite, wireless
- Or, connection from transmitter to receiver goes over many hops
  - Software company with development sites around the globe

Cost Measures and Scalability

Cost measures:

- Number of servers
- Outgoing server bandwidth
- Bandwidth utilization

A solution is called *scalable* if its cost does not increase with the number of recipients. (Server-scalable, bandwidth-scalable.)

Are interested in scalable and reliable solutions which maximize bandwidth utilization.

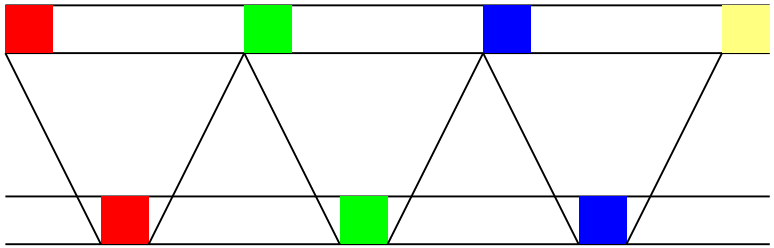
Implicit Assumptions

The solution has to be end-to-end, i.e., should run on any network with IP infrastructure without the need to change intermediate protocols.

The solution should be computationally efficient.

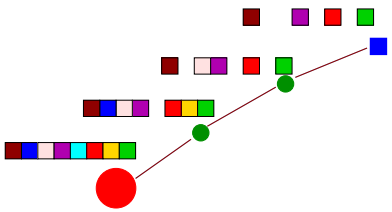
TCP over IP

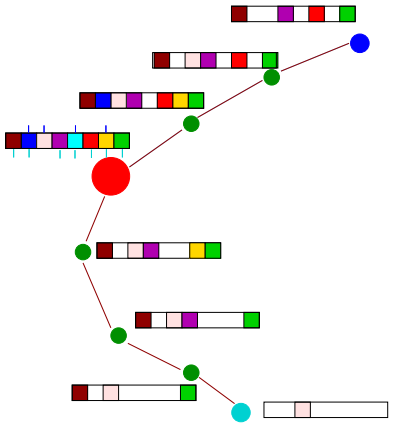
TCP ensures reliability using acknowledgements.

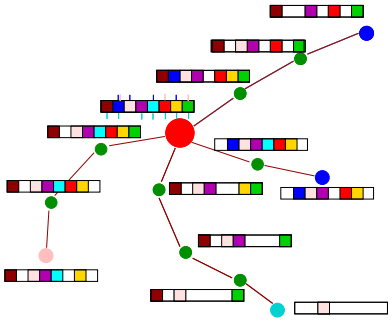


To avoid acknowledging all packets, a small *window* is used which constitutes packets in transit.

TCP/IP







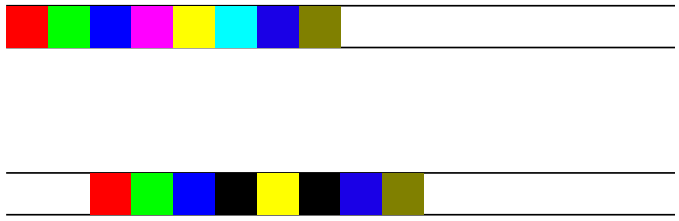
TCP/IP

TCP/IP is

- Reliable
- Not server scalable
  - Processing of ack's increases with number of receivers.
- Not bandwidth scalable
  - Each recipient requires a different connection.
- Not throughput maximizing
  - Loss rate of 2% reduces bandwidth utilization by factor of 5, instead of 0.98.

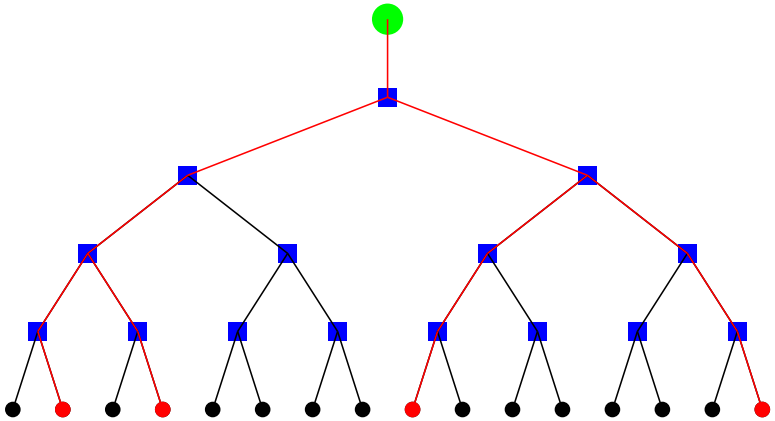
UDP Unicast

No back-channel to server.



- Not reliable
- Not server scalable
- Not bandwidth scalable
- Maximizes throughput

UDP Multicast



- Is not reliable,
- Is server scalable,
- Is bandwidth scalable,
- Maximizes throughput.

Scalability

The only known end-to-end bandwidth scalable protocol is Multicast.

It is not widely deployed because of fear of network breakdown (little, or no flow control).

In the following, we assume that the network is multicast enabled.

For more realisitic scenarios, one should replace Multicast with Unicast, and lose the bandwidth scalability.

Channel Model

On a computer network data is sent as packets.

Each packet has an identifier which identifies the entity it is coming from and its position within that entity.

Each packet has a CRC checksum to check its integrity.

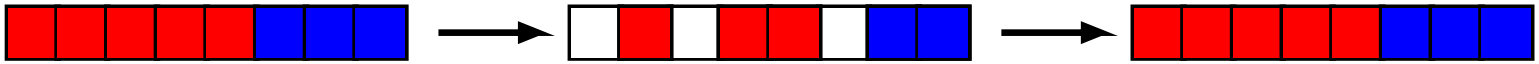
Corrupted packets can be regarded as lost.

Can concentrate on the erasure channel as a model for transmitting packets.

Solution: Codes

Want to have the advantages of Multicast and TCP/IP, but not their disadvantages.

Encode the original data and send encoded version across the network.



Reconstruction is possible if not too many packets were lost.

Reliability $\rightarrow$ Coding.

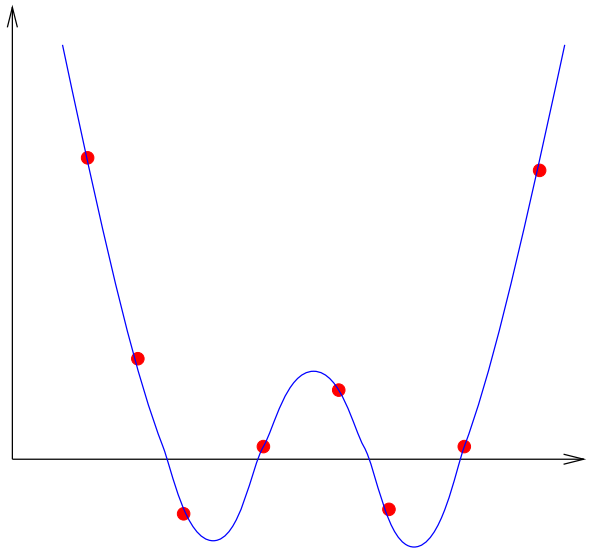
Scalability $\rightarrow$ Multicast (or unicast).

Reed-Solomon Codes

- Choose pairwise different values $x_1, x_2, \ldots, x_n$ in $\mathbb{F}_q$.

- RS-Code of dimension k is the image of the map

$$\mathbb{F}_q[x]_{<k} \to \mathbb{F}_q^n, \qquad f \mapsto (f(x_1), f(x_2), \ldots, f(x_n)).$$



Reed-Solomon Codes: Efficiency Issues

Encoding: multiplication of vector of length k with $k \times (n-k)$-matrix; needs $k \cdot (n-k)$ additions/multiplications in $\mathbb{F}_q$.

If $q = 2^\ell$, then operations in $\mathbb{F}_q$ need ℓ^2 bit-operations.

Number of operations per bit is $n \cdot \ell \cdot R \cdot (1-R)$, where R is rate of the code.

Decoding: More than $k \cdot e$ operations over $\mathbb{F}_q$ where e is number of erasures within the information positions.

Number of operations per bit is $\ell \cdot e$.

Depending on ℓ, number of operations is moderate if $n-k$ is small, i.e., only few erasures can be corrected.

Effect of Packet Size and Interleaving

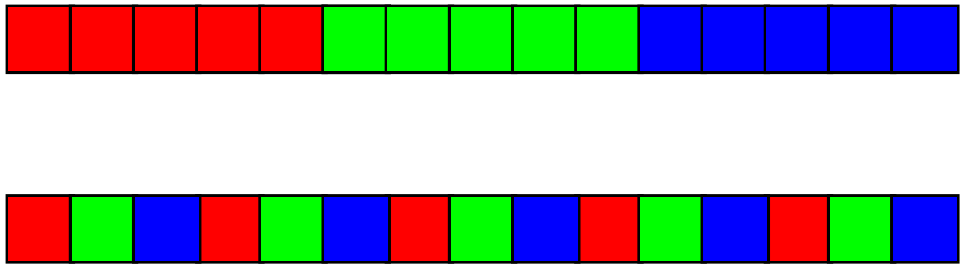
Internet packets are typically of size 1024 bytes, but alphabet of 1024 bytes is difficult to handle in software.

Good alphabet size in practice: 1 byte.

Use interleaving:

Can deal with files of size 256 KB at most. What about larger files?

Packet interleaving



Good if burst losses, extremely bad if random losses: Will receive lots of duplicate packets. (Occupancy problem.)

Reed-Solomon Codes: Summary

Encoding and decoding complexity per bit is $\ell \cdot k \cdot (1 - k/n)$.

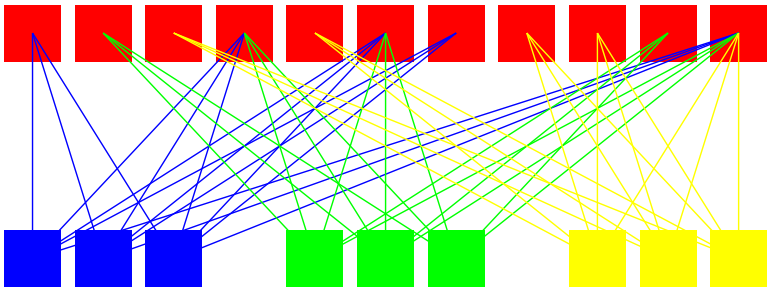
Reasonable encoding, decoding, and reception speeds if file is small, few redundant symbols, and worst possible loss rate of network known in advance.

But: Need lots of redundant symbols in many applications.

Need codes for which encoding and decoding complexity per bit is constant, and which do not need packet interleaving.

How to Make Reed-Solomon Codes more Efficient?

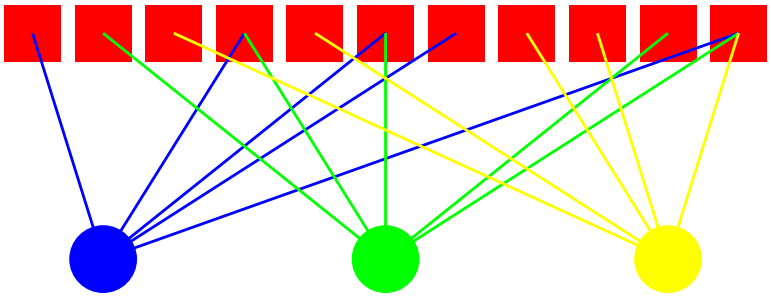
Use small blocks to compute RS redundant symbols from.



Choose the blocks randomly.

How to Make Reed-Solomon Codes more Efficient?

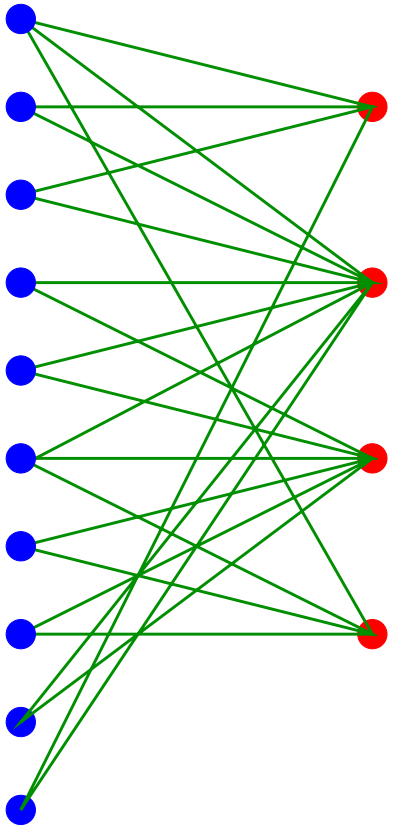
Use small blocks to compute RS redundant symbols from.



Choose the blocks randomly.

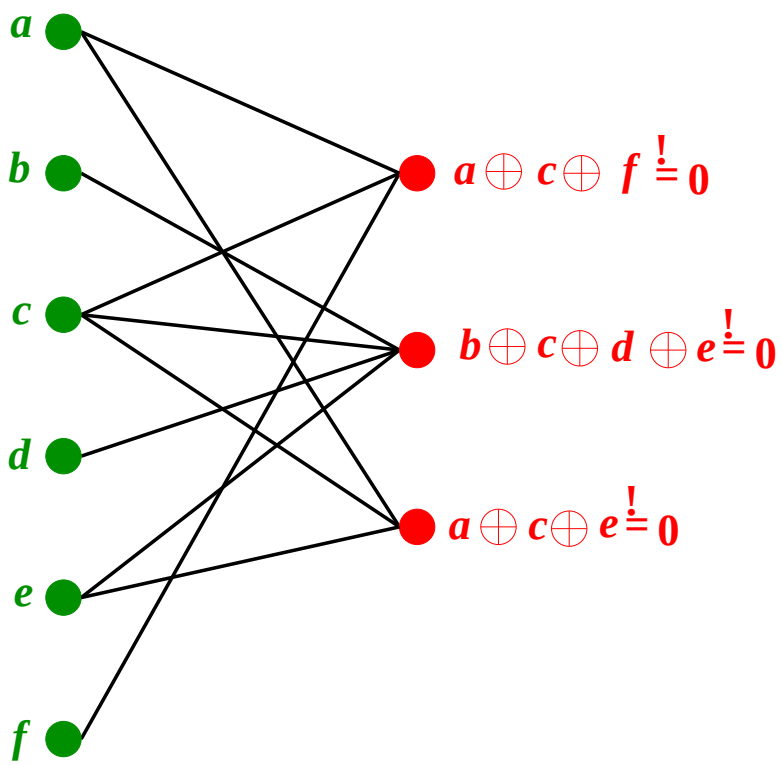
LDPC Codes

Constructed from sparse bipartite graphs.



Left nodes are called message nodes, right nodes are called check nodes.

Construction



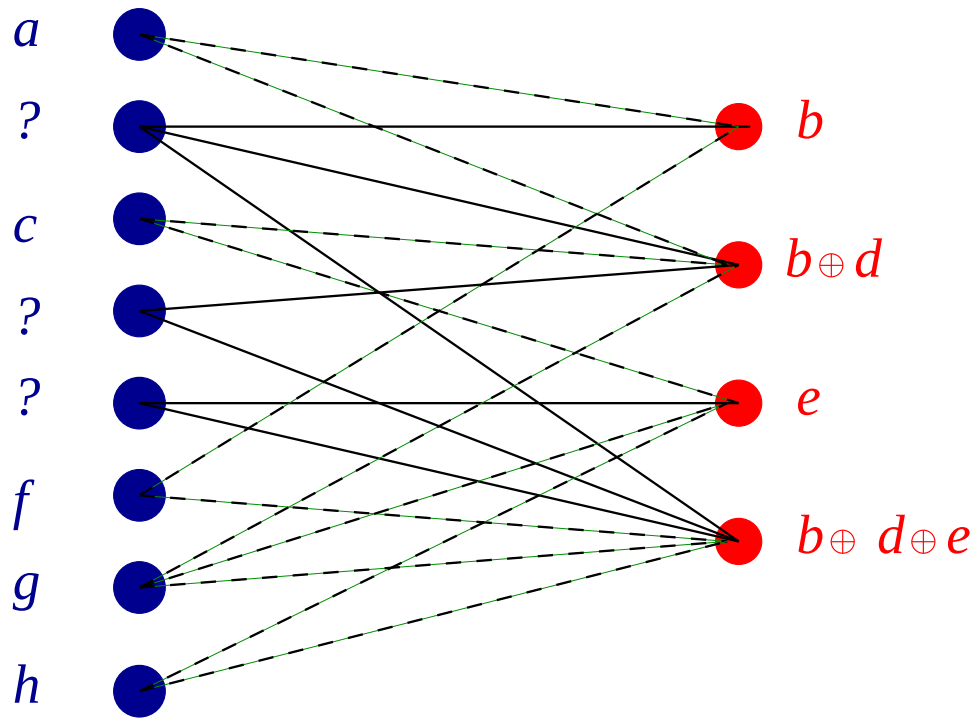
Every binary linear code has such a representation, but not every code can be represented by a sparse graph.

Encoding/decoding times?

Decoding

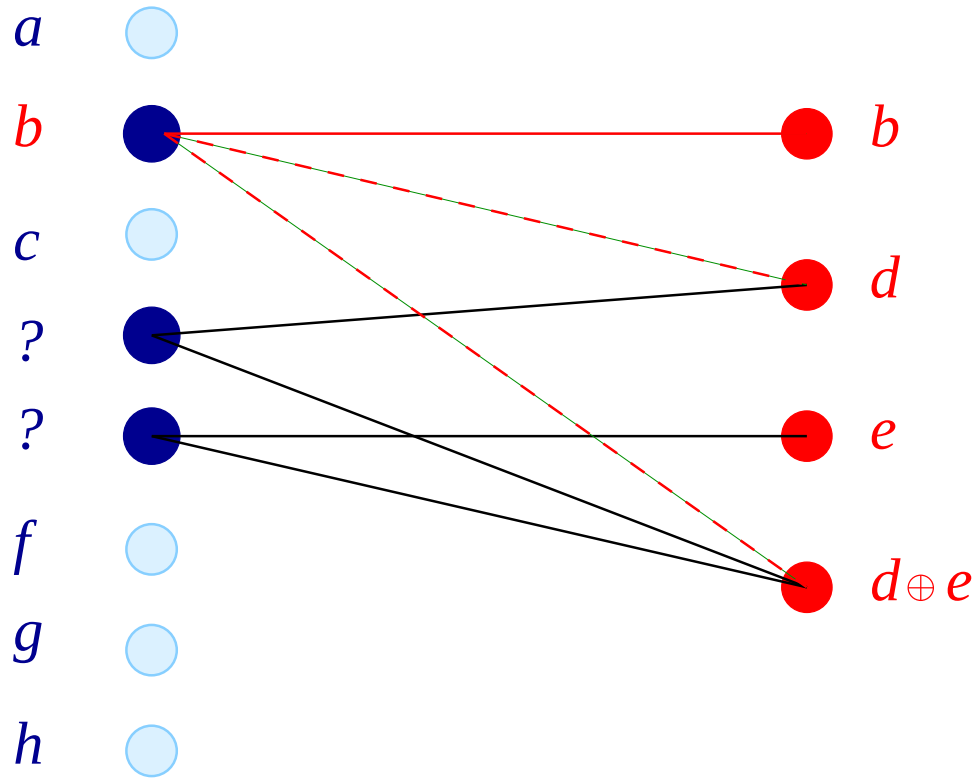
Luby-Mitzenmacher-Shokrollahi-Spielman-Stemann, 1997:

Phase 1: Direct recovery



Decoding

Phase 2: Substitution



Decoding Time

If each symbol is interpreted as a packet with b bits, then we need $\#E \cdot b$ operations to produce $n \cdot b$ bits, where $\#E$ is the number of edges in the graph, and n is the block-length of the code.

Number of operations per bit is

$$\#E/n$$

which is constant for sparse graphs.

The Inverse Problem

We have a fast decoding algorithm.

Want the design the graph in such a way that many erasures are correctable with this algorithm.

Experiments

Choose random bi-regular graph.

p_0 := Maximal fraction of correctable erasures.

d	k	d/k	p_0
3	6	0.5	0.429
4	8	0.5	0.383
5	10	0.5	0.341
3	9	0.33	0.282
4	12	0.33	0.2572

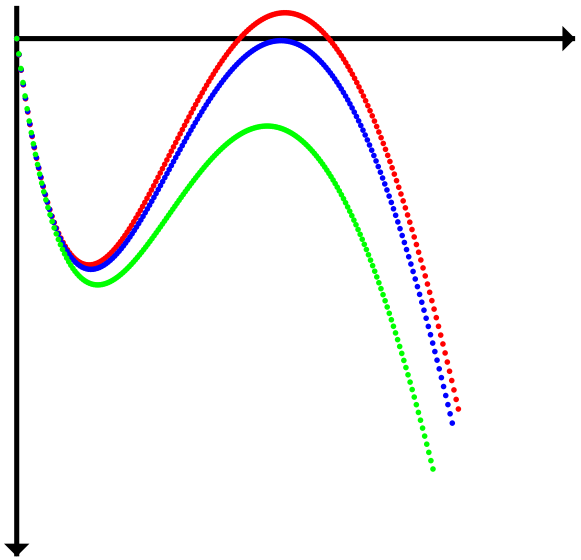
Where do these numbers come from?

Theorem

(Luby, Mitzenmacher, Shokrollahi, Spielman, Stemann 1997) A random (d, k)-graph allows correction of a p_0-fraction of erasures (with high probability) if and only if

$$p_0 \cdot (1 - (1 - x)^{k-1})^{d-1} < x \qquad \text{für } x \in (0, p_0).$$

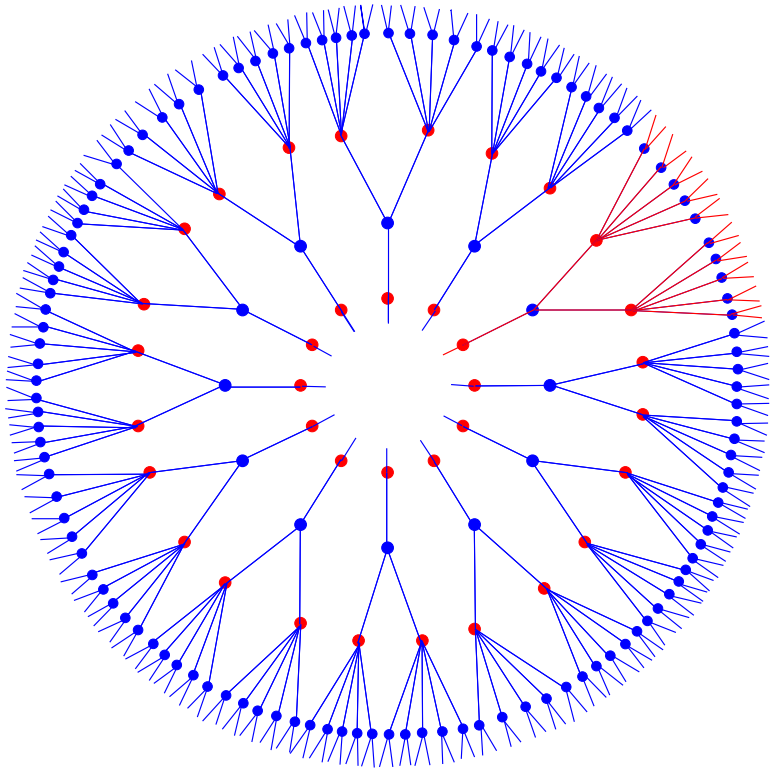
$d = 3, k = 6$: $\quad f(x) = p_0(1 - (1 - x)^5)^2 - x$



$p_0 = 0.435$
$p_0 = 0.429$
$p_0 = 0.41$

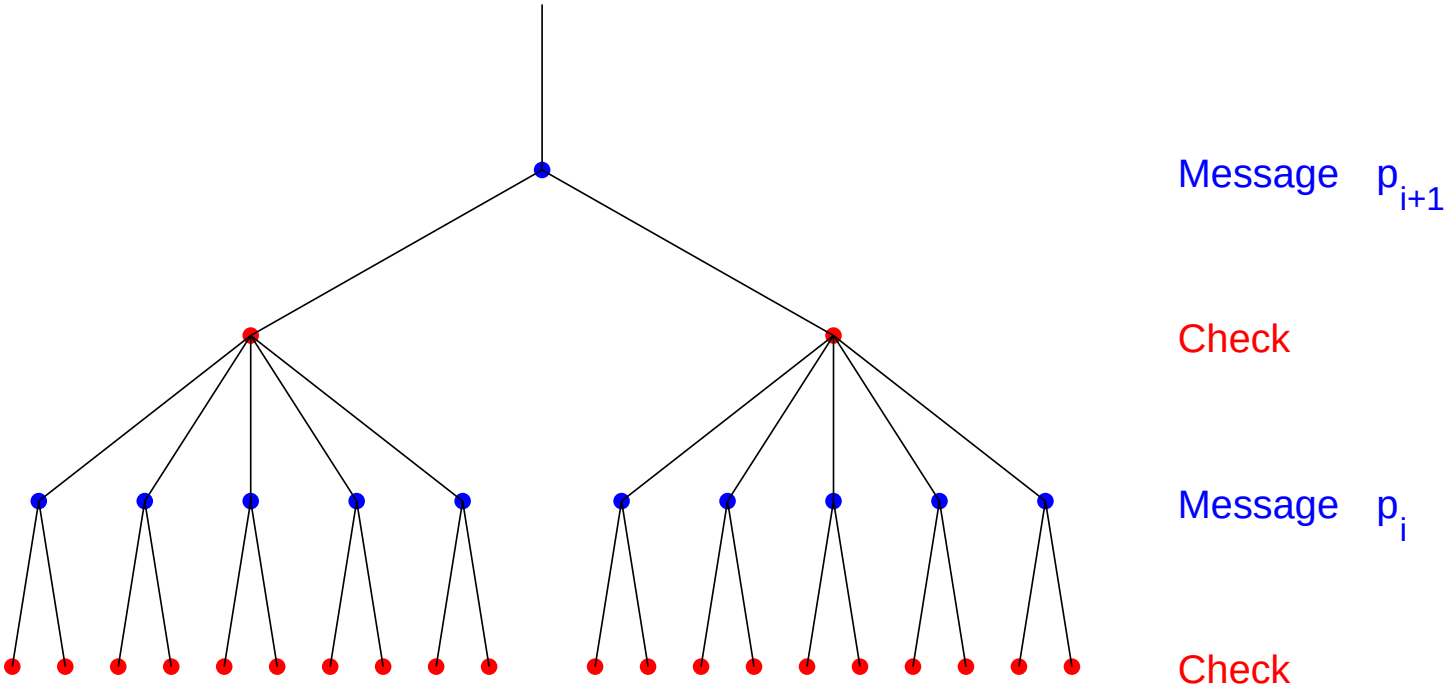
Analysis: $(3,6)$-**Graph**

Expand neighborhood of message node



Analysis: $(3,6)$-Graph

p_i = Probability that a message node is not corrected after the i-th iteration.



$$p_{i+1} = p_0(1-(1-p_i)^5)^2 < p_i.$$

Analysis: $(3, 6)$-Graph

Rigorous argument:

- Neighborhood is a tree with high probability.

- Above argument works fine for the expected fraction of erasures after the i-th iteration.

  Actual value is concentrated around the expectation p_ℓ: Edge exposure martingale, Azuma's Inequality.

The General Case

λ_i and ρ_i fraction of edges of degree i on the left and the right hand side of the graph.

$\lambda(x) := \sum_i \lambda_i x^{i-1}$, $\rho(x) := \sum_i \rho_i x^{i-1}$.

Condition for successful decoding given a loss fraction p_0:

$$p_0 \lambda\left(1 - \rho(1-x)\right) < x$$

for all $x \in (0, p_0)$.

Encoding

Trick: Use erasure correction

Richardson-Urbanke, 1999: linear time encoding possible under mild conditions.

- Enough message nodes of degree 2: $\lambda_2 \rho'(1) > 1$.

- $\rho(1 - \lambda(1 - x)) < x$ for $x \in (0, 1)$.

Tornado Codes

Want to design codes which can asymptotically correct an optimal fraction of erasures.

Design λ and ρ such that

$$p_0 \lambda(1 - \rho(1 - x)) < x$$

for all $x \in (0, p_0)$, and p_0 arbitrarily close to

$$1 - R = \frac{\int_0^1 \rho(x) \mathrm{d}x}{\int_0^1 \lambda(x) \mathrm{d}x}.$$

Tornado Codes

Choose design parameter D:

$$\lambda(x) := \frac{1}{H(D)}\left(x + \frac{x^2}{2} + \cdots + \frac{x^D}{D}\right)$$

$$\rho(x) := \exp\left(\mu(x-1)\right),$$

$H(D) = 1 + 1/2 + \cdots + 1/D$, $\mu = H(D)(1-R)/\left(1 - 1/(D+1)\right)$.

$$p_0\lambda(1-\rho(1-x)) = p_0\lambda(1-\exp(-\mu x)) < -\frac{p_0}{H(D)}\ln(\exp(-\mu x))$$

$$= \mu\frac{p_0}{H(D)}x < x.$$

This is true for $p_0 < H(D)/\mu = (1-R)(1-1/(D+1))$.

Tornado Codes: Efficiency

Need $k \cdot (1 + \varepsilon)$ entries of a codeword to recover the codeword.

Per-bit running time of encoding is $O(\log(1/\epsilon))$.

Per-bit running time of the decoder is $O(\log(1/\epsilon)/R)$.

Tornado codes scale to arbitary lengths and do not requrie packet interleaving.

However, need a good guess of the loss rate of the channel.

Theoretical Applications

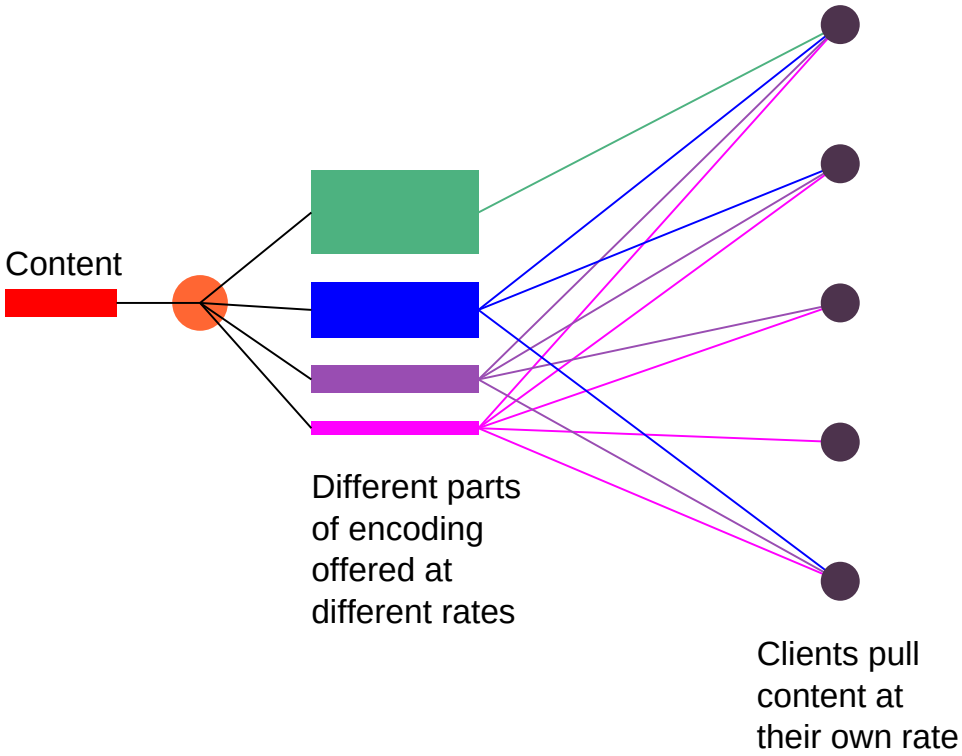
- Capacity achieving sequences on the erasure channel
- Relationship to random graphs
- Codes on other channels
- Algebraic constructions
- Analysis of finite length codes
- Cryptography
- ...

Practical Applications

- Receiver driven rate control
- Multipath delivery
- Download from multiple servers
- Peer-to-peer networks
- Video-on-Demand
- Fault tolerant storage
- .....

Receiver Driven Rate Control

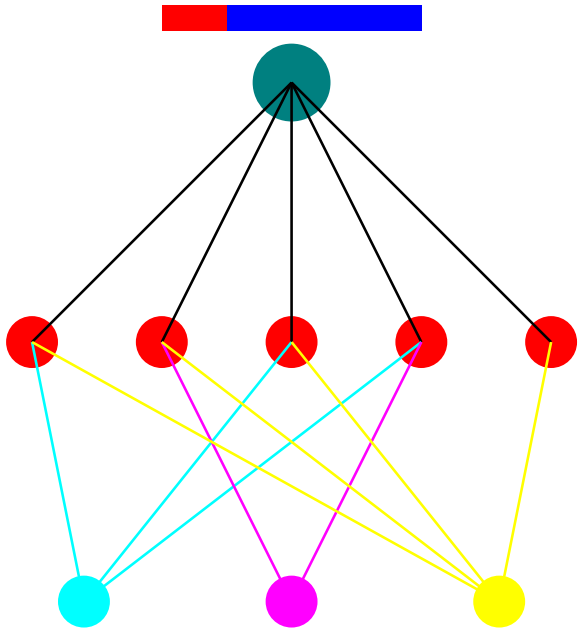
Network traffic generated by UDP should be fair to other network flows like TCP.



A lot of loss, by design. Avoiding duplicate packets is difficult.

Path Diversity

Send data across different paths to receiver to bypass links that are down.



Avoiding duplicate packets is difficult.

Shortcomings of Traditional Codes

Protocols involving erasure correcting codes are excellent candidates for replacing TCP/IP in certain applications. However, traditional codes have many disadvantages:

- One has to have a good guess on the loss rate of the clients; this is very difficult in scenarios like mobile wireless.

- Many applications require coordination between senders and receivers to avoid reception of duplicate packets.

- Many applications require codes of very small rate. But, the running time of fast codes like Tornado codes is proportional to the block-length, rather than the length of the original content.

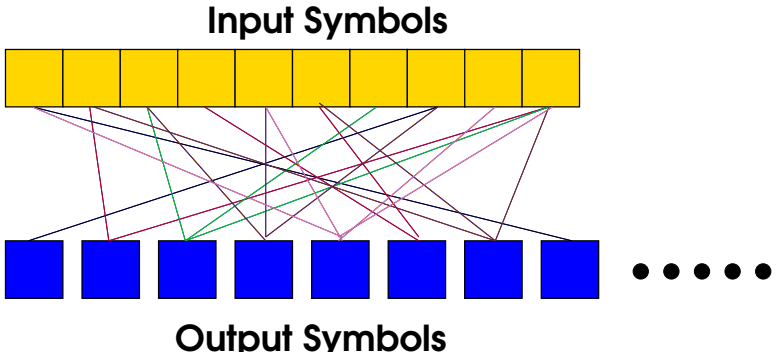
What we Really Want

To design a completely receiver driven and scalable system one needs codes

- That adapt themselves to the individual loss rates of the clients; clients with more loss need longer to recover the content;
- For which the decoding time depends only on the length of the content;
- That achieve capacity of the erasure channel between server and any of the clients;
- That have fast encoding and decoding algorithms.

Beyond Tornado: LT Codes

Michael Luby has invented a class of codes that achieves all these goals.

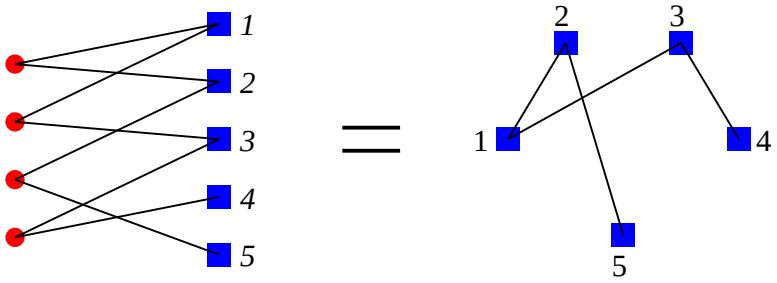


Their per-bit complexity of encoding/decoding is $\log(k)$. (This is optimal.)

If a receiver has loss rate p, then the code corresponding to that receiver has rate $1 - p - c/\sqrt{k}$.

Relationship to Random Graphs

Consider codes in which all message nodes have degree 2. These are graphs on the set of check nodes.



Choose neighbors of message nodes randomly (Poisson distribution). This yields a random graph on the set of check nodes.

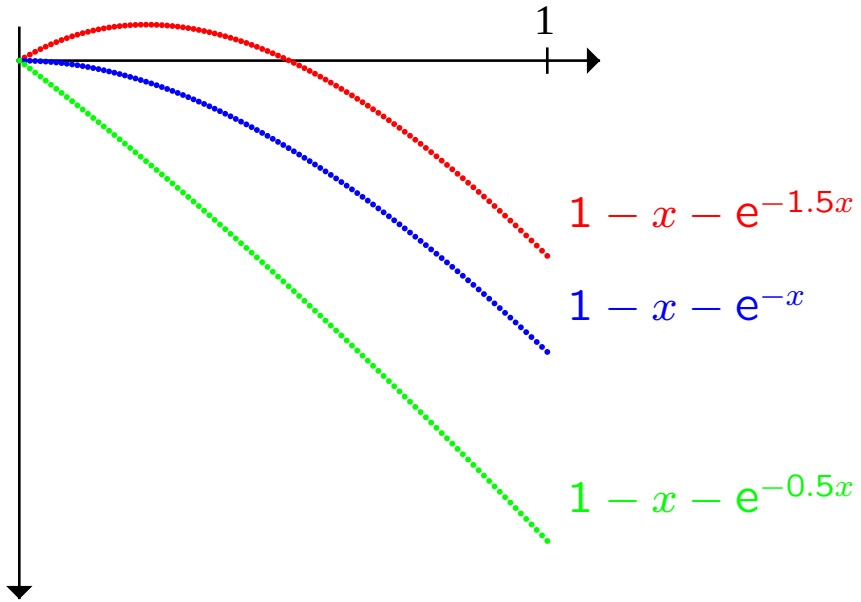
$\lambda(x) = x$, $\rho(x) = \mathrm{e}^{a(x-1)}$, a is average degree.

$\lambda(1 - \rho(1 - x)) < x$, i.e., $1 - x - \mathrm{e}^{-ax} < 0$.

Relationship to Random Graphs

Largest solution of $1 - x - e^{-ax} = 0$ in the interval $(0, 1)$ gives fraction of the largest component of the graph.

There is a giant component if $a > 1$. The decoder will not be able to correct all erasures iff $a > 1$, i.e., iff there is a giant component.



Relationship to Random Graphs

Above relationship can be put into a general framework, which works even when not all the message nodes are of degree 2.

The Tornado distribution can be obtained from this observation using a "self-similarity" assumption.

This connection also yields linear time encoders.