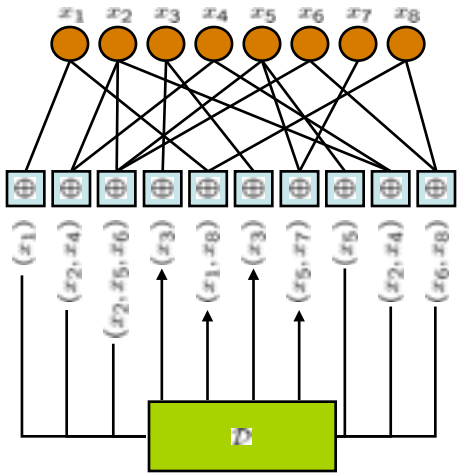
Fountain Codes



Amin Shokrollahi

EPFL

Overview

PART I: The Erasure Channel

- Communication Problem
- Fountain Codes
- Asymptotic and Finite Length Design
- Designs for very small lengths (3GPP)
- Applications

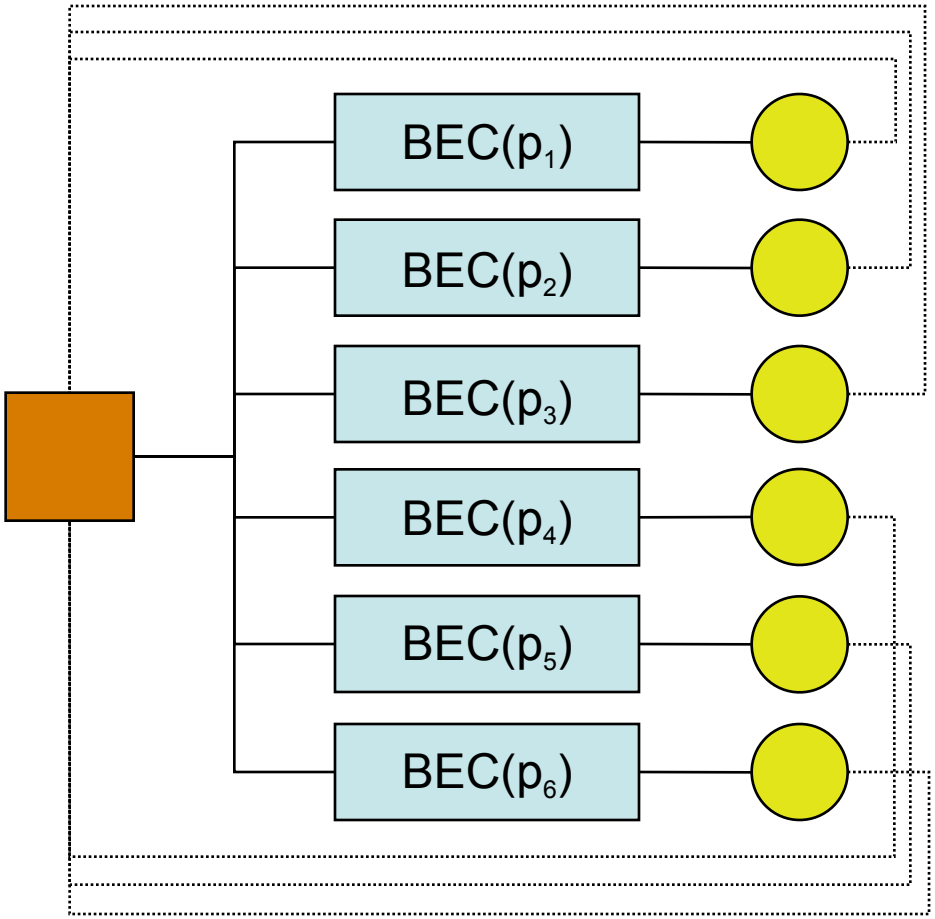
PART II: General Symmetric Channels

- Communication Problem
- Fountain Codes for Symmetric Channels
- Asymptotic Design
- Applications

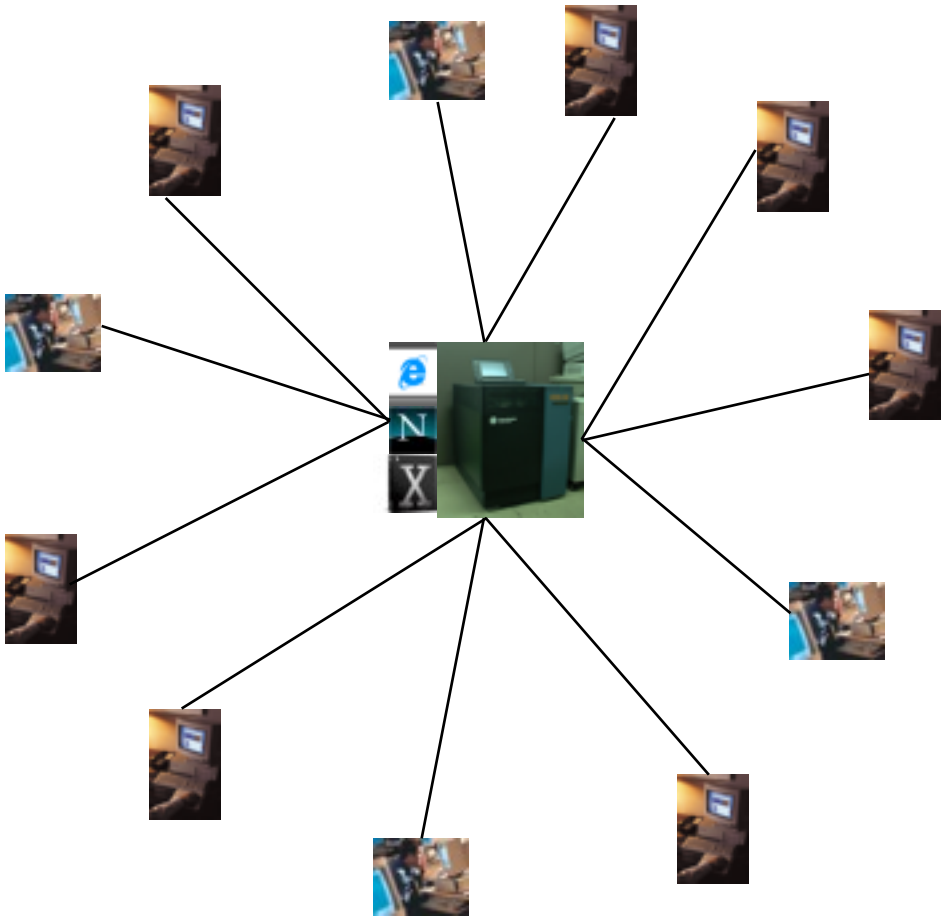
algolma
laboratoire d'algorithmique
laboratoire de mathematiques algorithmique

BEC

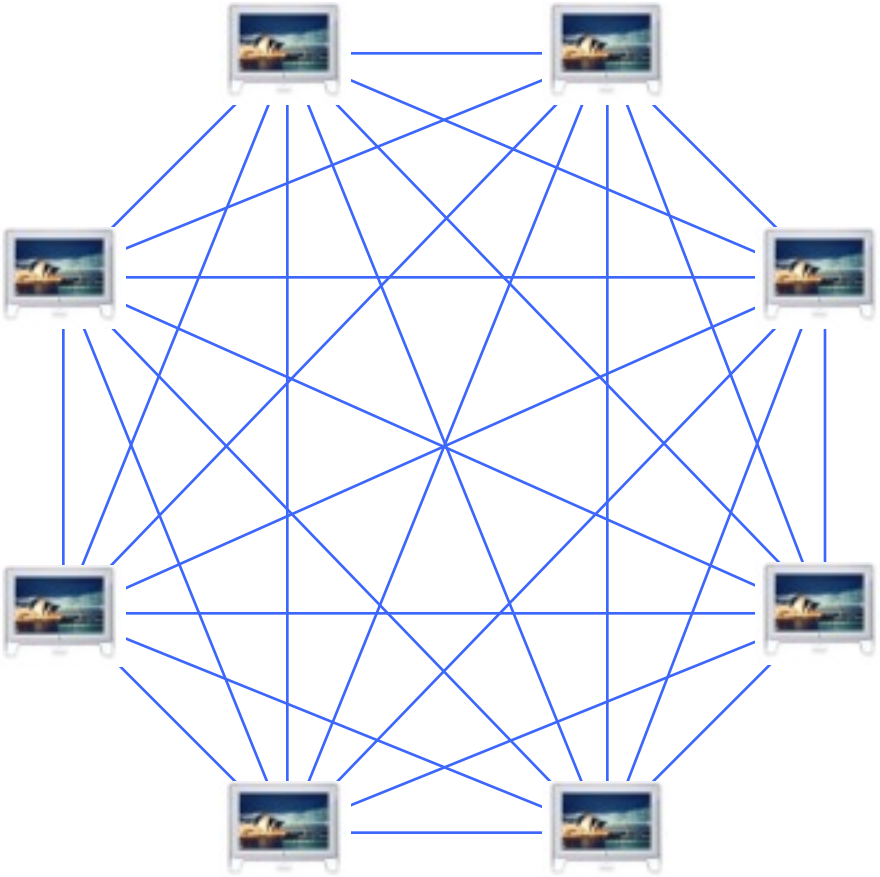
Communication on Multiple Unknown Channels



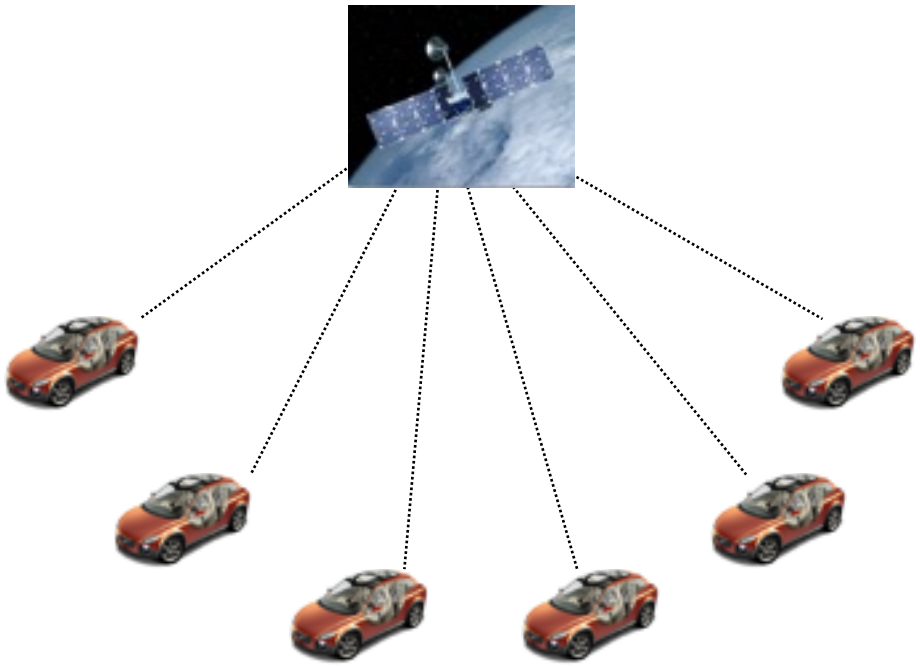
Example: Popular Download



Example: Peer-to-Peer



Example: Satellite



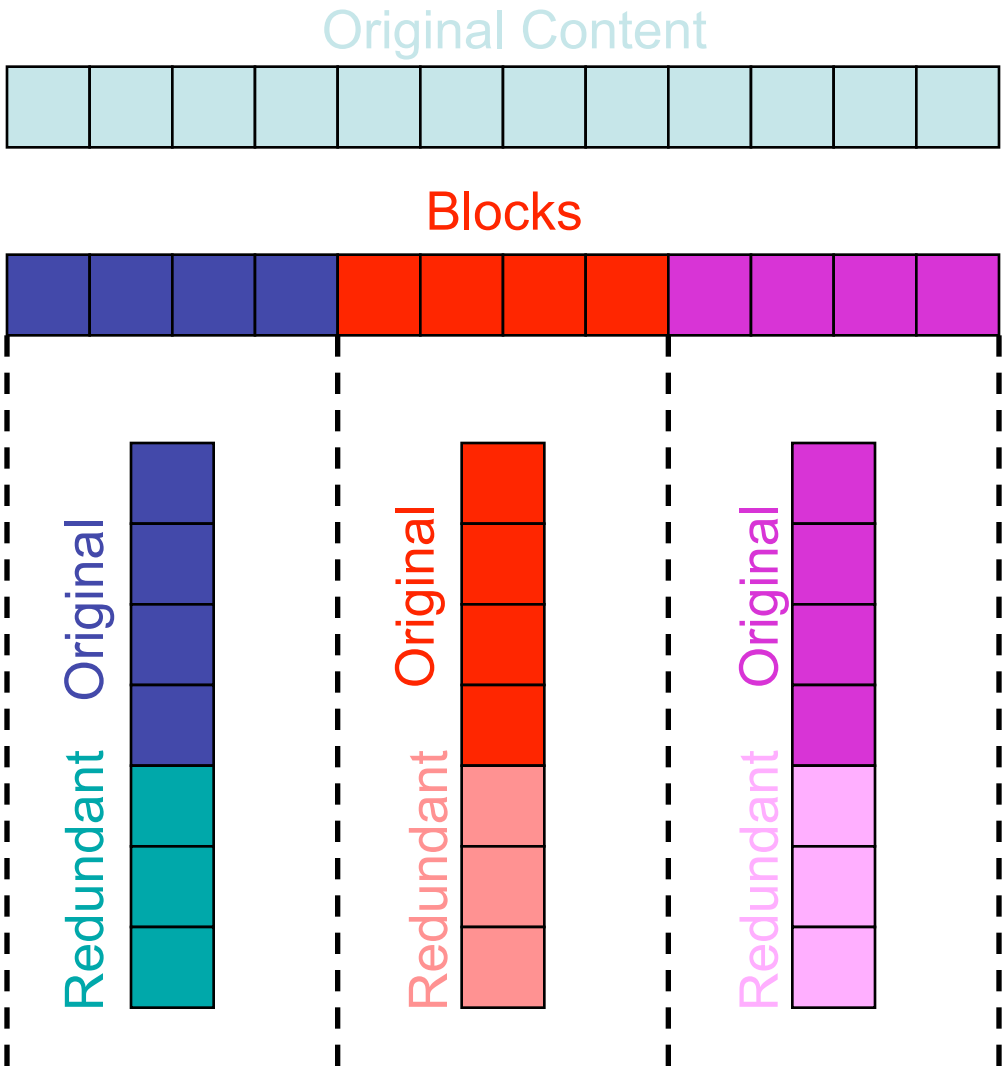
The erasure probabilities are unknown.

Want to come arbitrarily close to capacity on each of the erasure channels, with minimum amount of feedback.

Traditional codes don't work in this setting since their rate is fixed.

Need codes that can adapt automatically to the erasure rate of the channel.

Traditional FEC



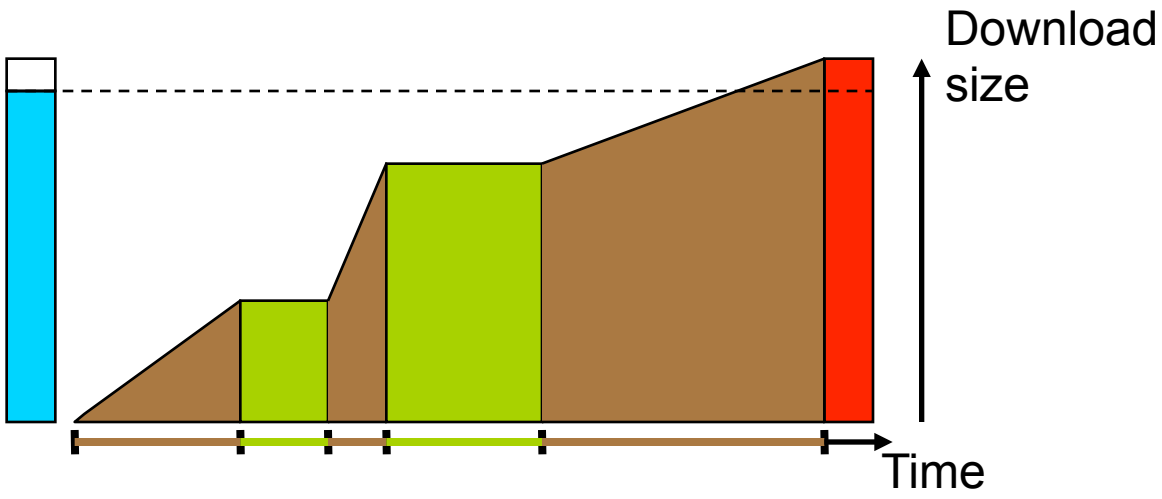
algoIma
laboratoire d'algorithmique
laboratoire de mathématiques algorithmique

Problems with FEC

- Fraction of losses *must* be less than $\frac{r}{k+r}$
- Worst user dictates amount of redundancy
- Loss provisioning is complicated and leads to overhead

FEC + Carousel

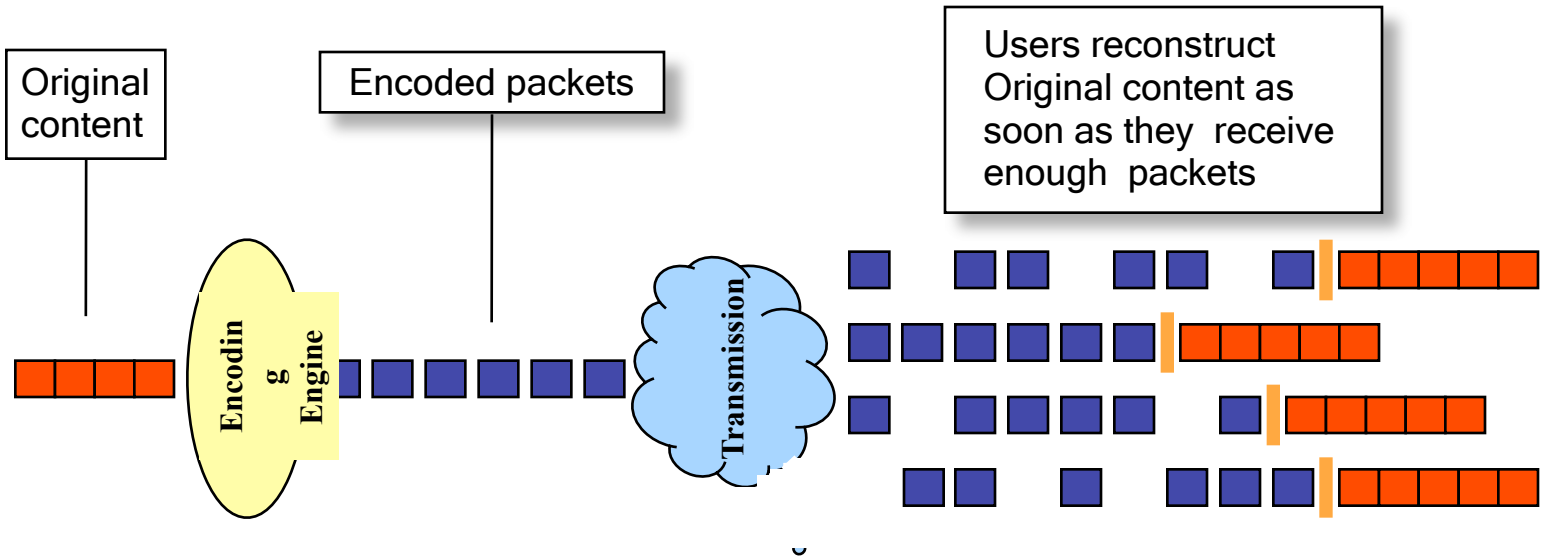
Users adjust to their individual reception rate.



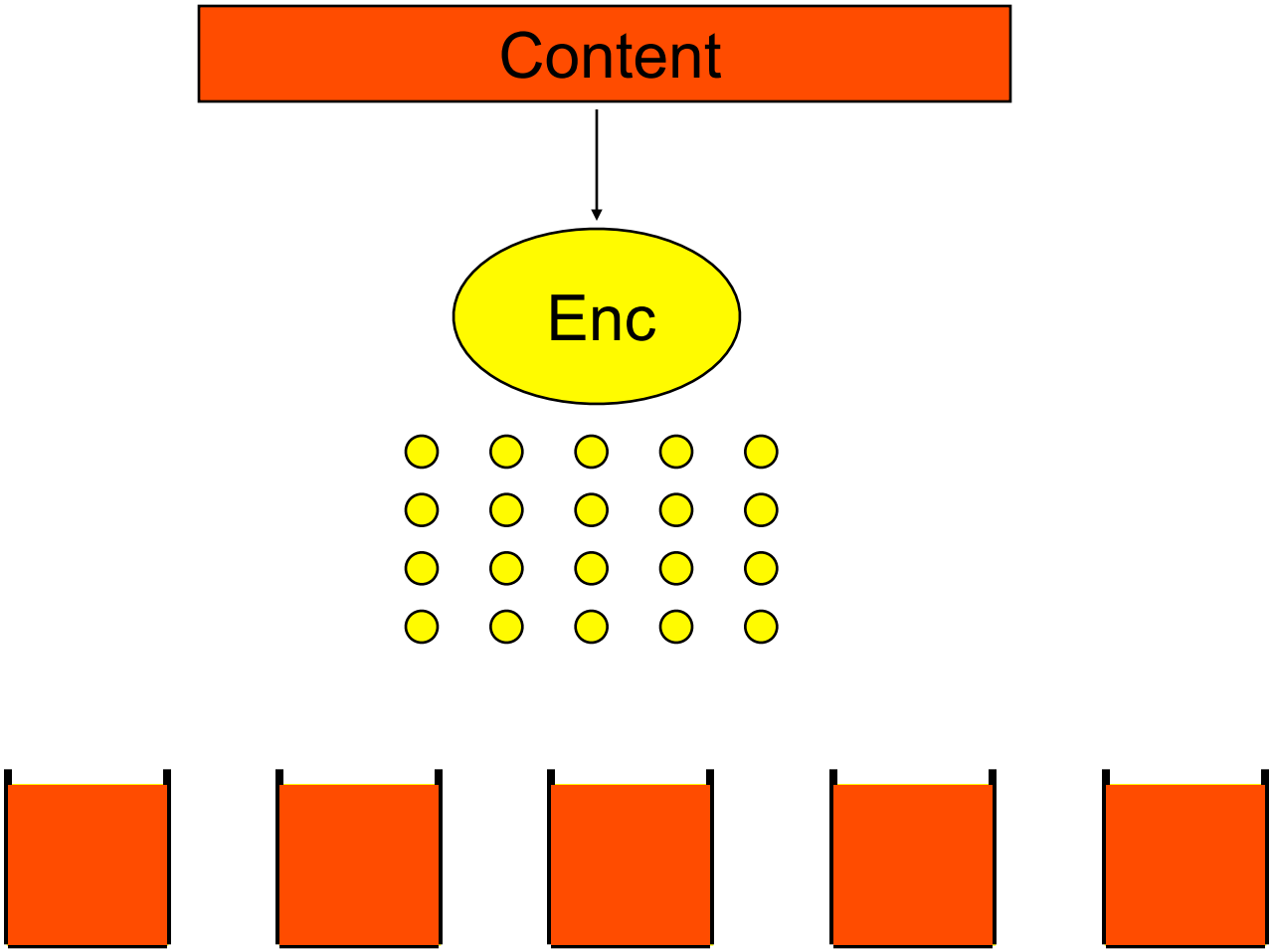
FEC + Carousel

- Need fast codes for large content (e.g., Tornado)
- Need high redundancy to avoid coupon collector phenomenon
- Tornado codes run in time *O(n)* not *O(k)*

What we Really Want



Reconstruction time should depend only on size of content



Digital buckets

Fountain Codes

Sender sends a potentially limitless stream of encoded bits.

Receivers collect bits until they are reasonably sure that they can recover the content from the received bits, and send STOP feedback to sender.

Automatic adaptation: Receivers with larger loss rate need longer to receive the required information.

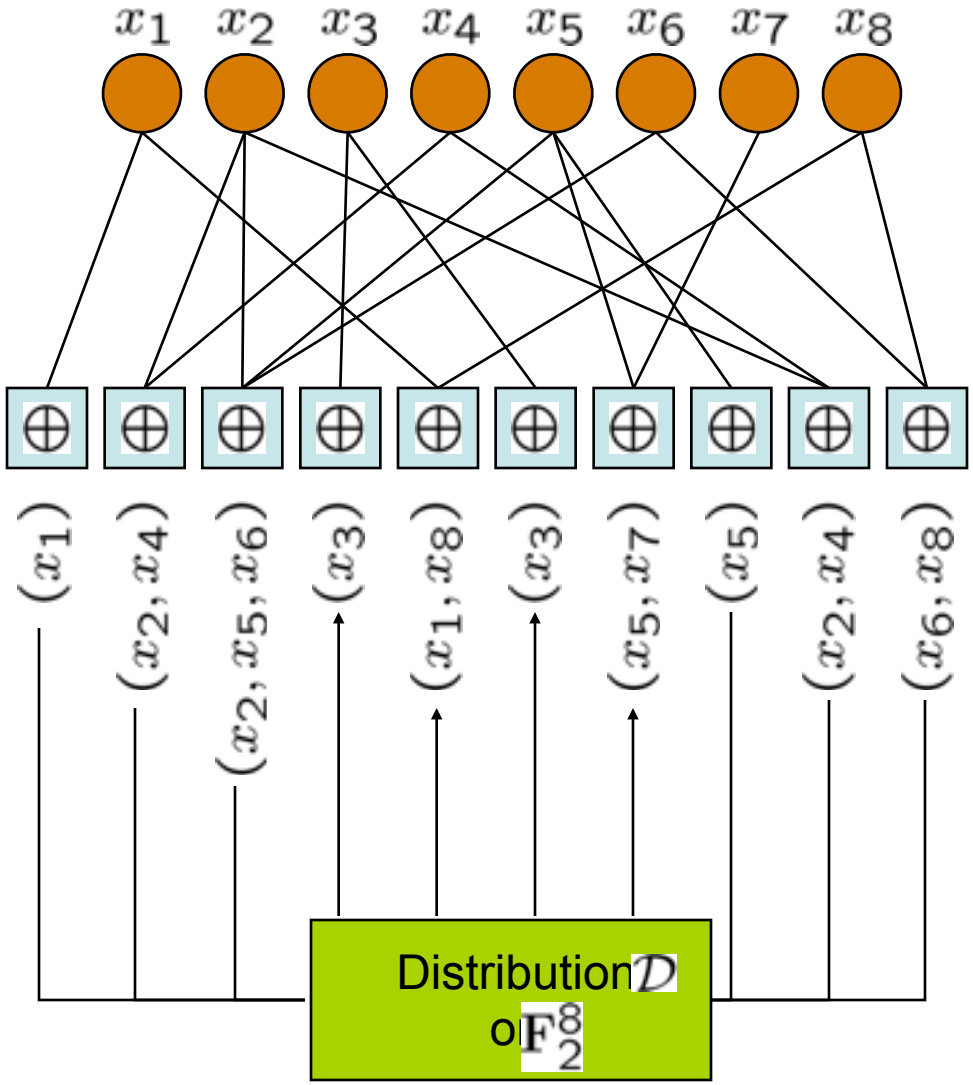
Want that each receiver is able to recover from the minimum possible amount of received data, and do this efficiently.

Fountain Codes

Fix distribution $\mathcal{D}$ on $\mathbf{F}_2^k$, where k is number of input symbols.

For every output symbol sample independently from $\mathcal{D}$ and add input symbols corresponding to sampled subset.

Fountain Codes



Universality and Efficiency

[Universality]
Want sequences of Fountain Codes for which the overhead is arbitrarily small

[Efficiency]
Want per-symbol-encoding to run in close to constant time, and decoding to run in time linear in number of output symbols.

LT-Codes

- Invented by Michael Luby in 1998.
- First class of universal and almost efficient Fountain Codes
- Output distribution has a very simple form
- Encoding and decoding are very simple

LT-Codes

LT-codes use a restricted distribution on $\mathbf{F}_2^k$:

Fix distribution $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$ on $\{1, 2, \ldots, k\}$

Distribution $\mathcal{D}$ is given by

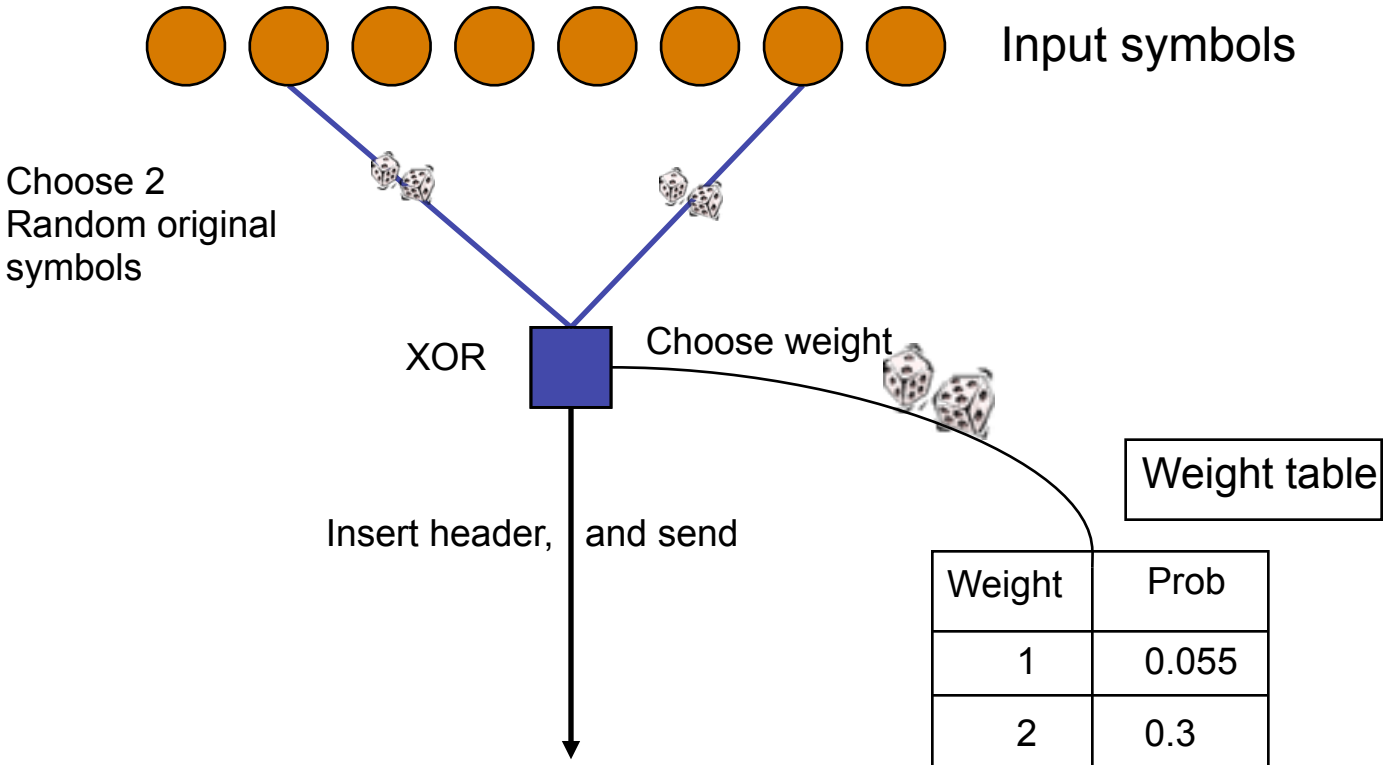
$$\mathrm{Prob}_{\mathcal{D}}(x) = \frac{\Omega_w}{\binom{k}{w}}$$

where w is the Hamming weight of x

Parameters of the code are $(k, \Omega(x))$

$$\Omega(x) = \Omega_1 x + \Omega_2 x^2 + \cdots + \Omega_k x^k$$

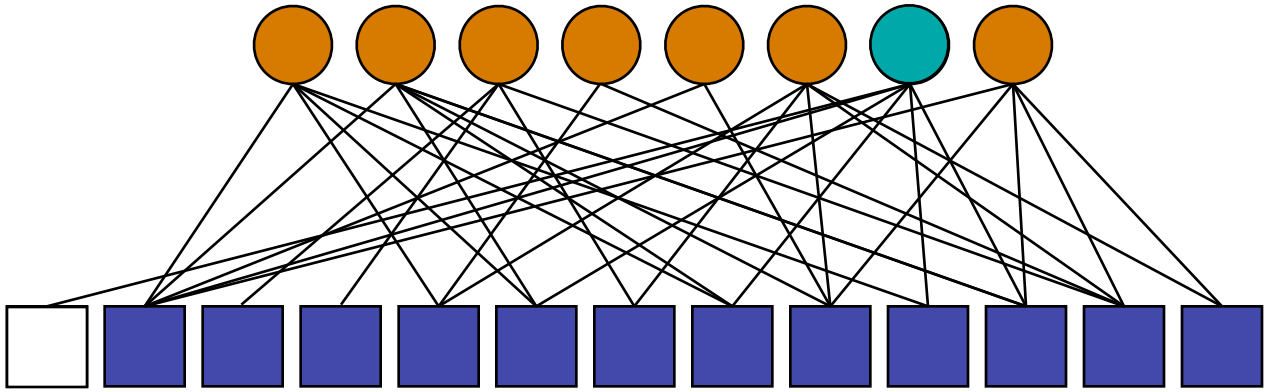
The LT Coding Process



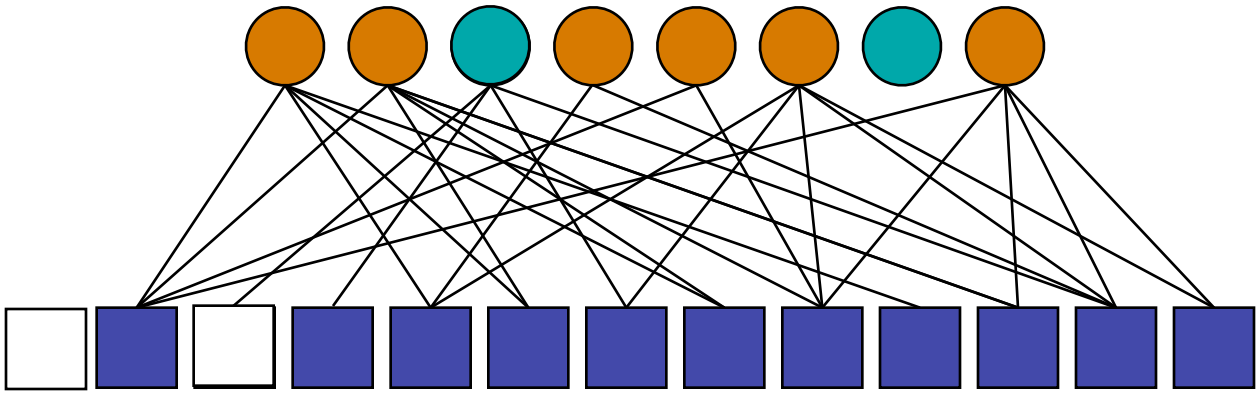
Weight table

Weight	Prob
1	0.055
2	0.3
3	0.1
4	0.08
⋮	⋮
100000	0.0004

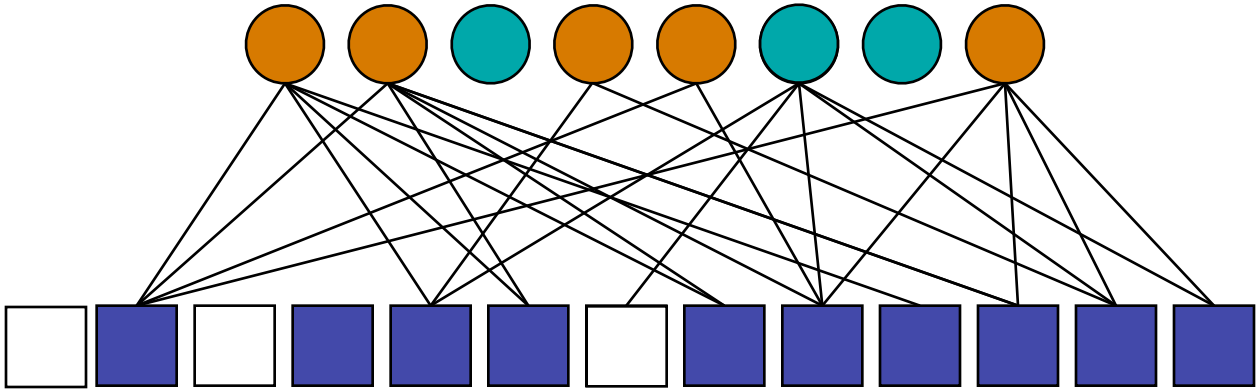
Decoding



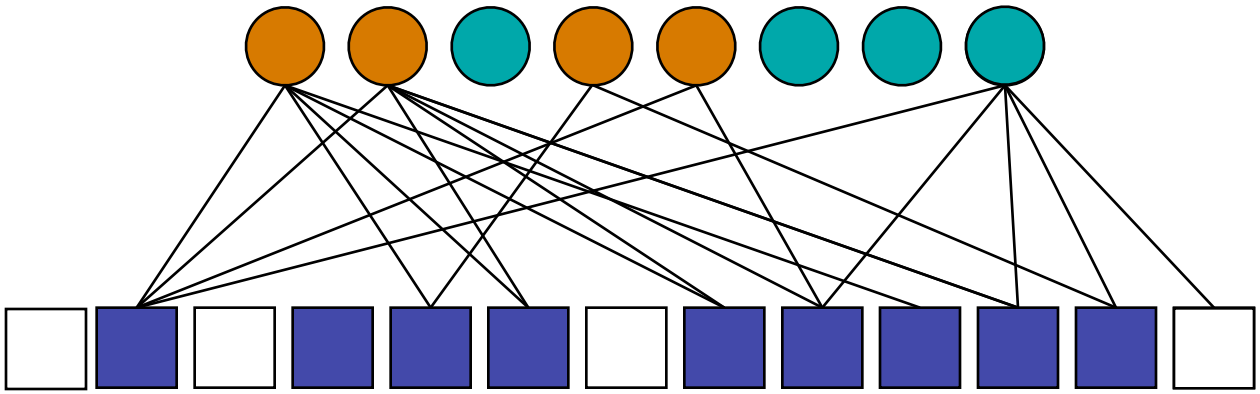
Decoding



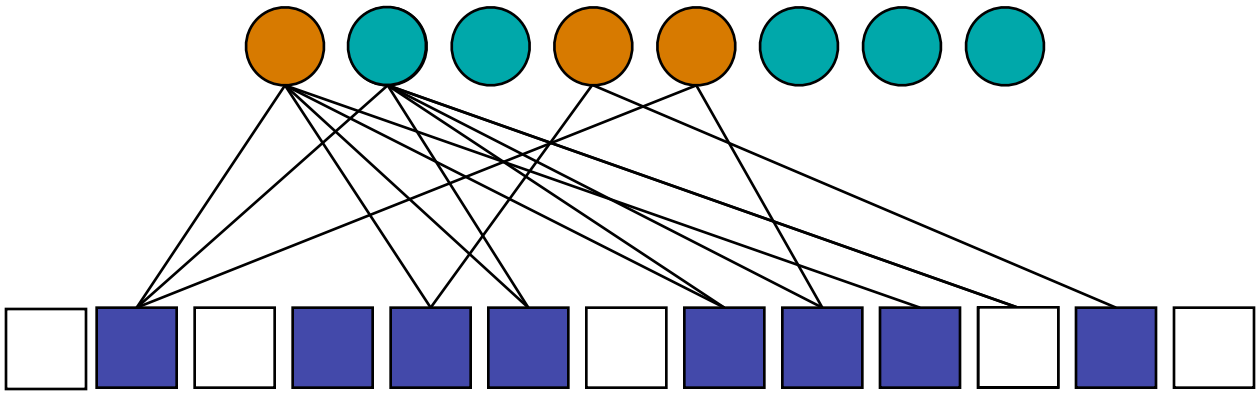
Decoding



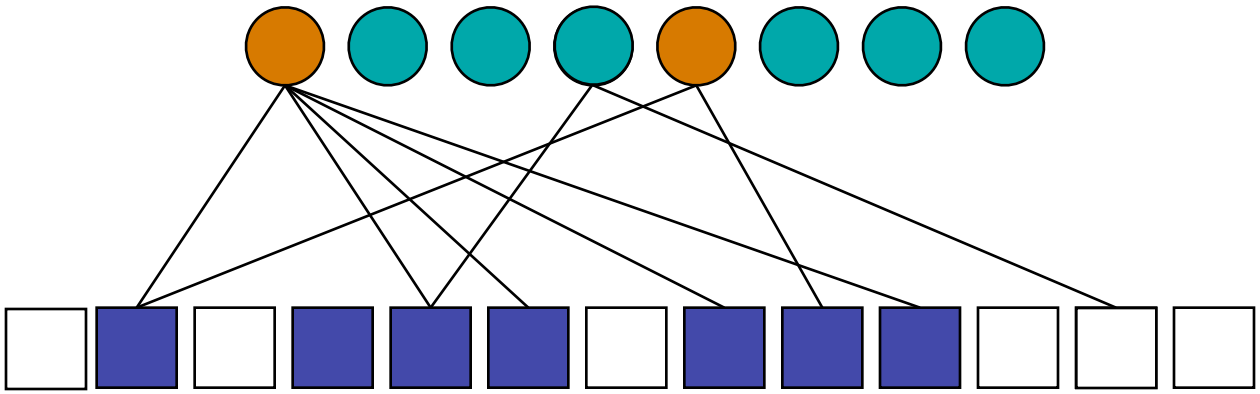
Decoding



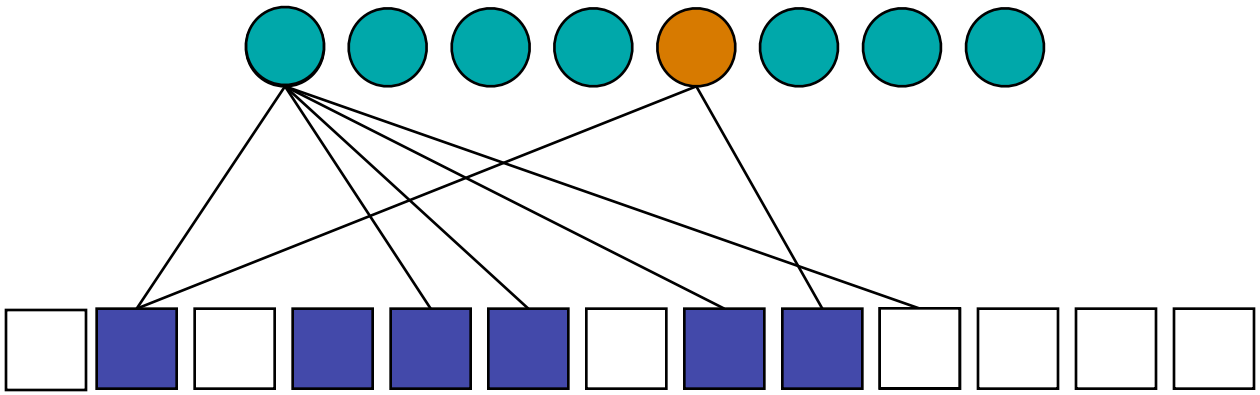
Decoding



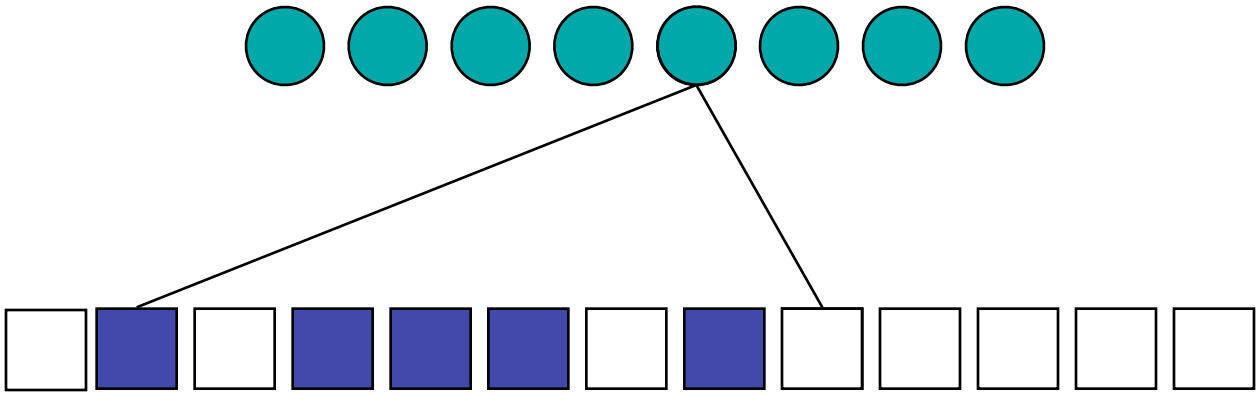
Decoding



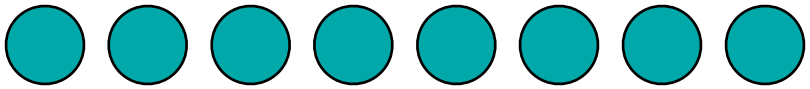
Decoding

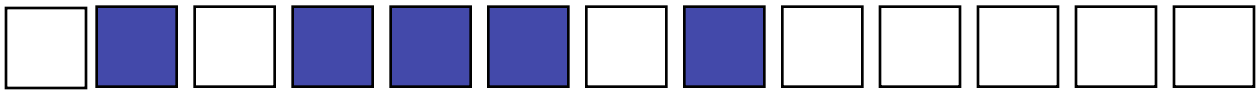


Decoding



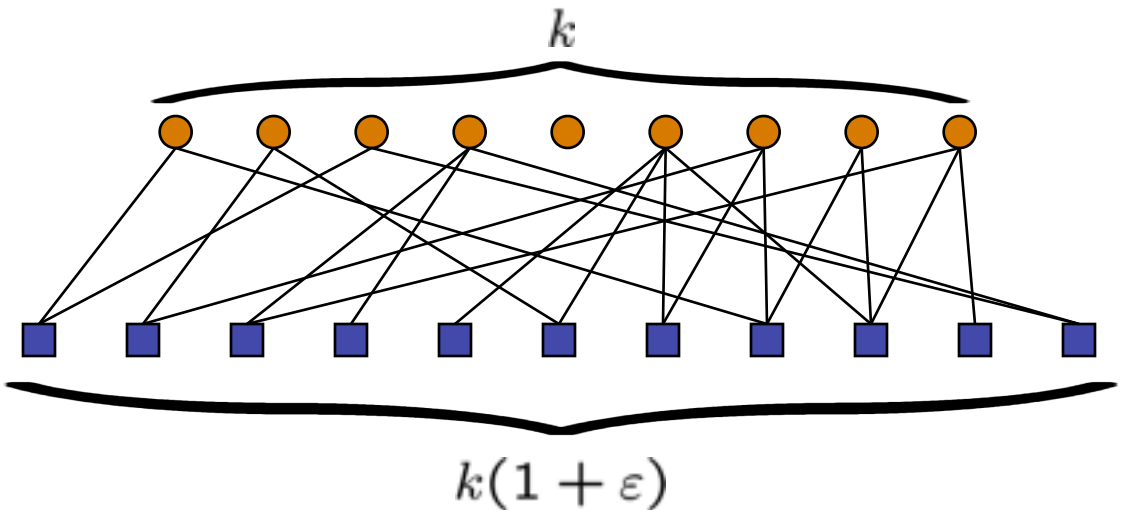
Decoding





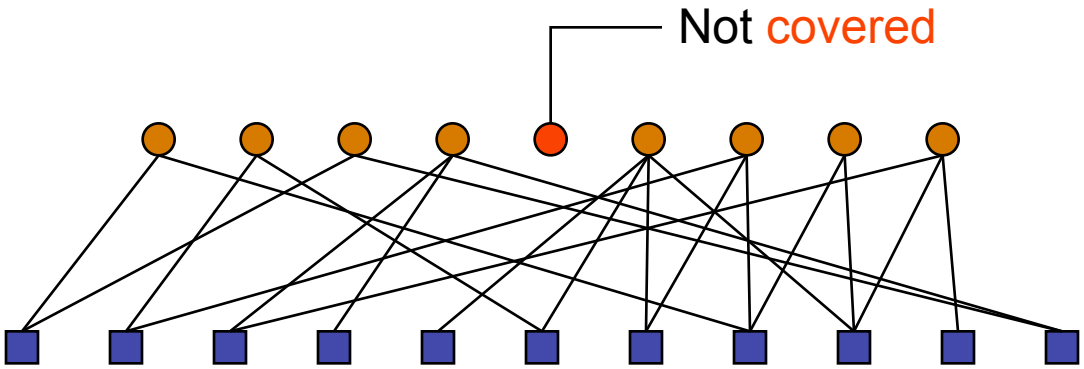
Average Degree of Distribution should be

$$O(\log(k))$$



Average Degree of Distribution should be

$$O(\log(k))$$



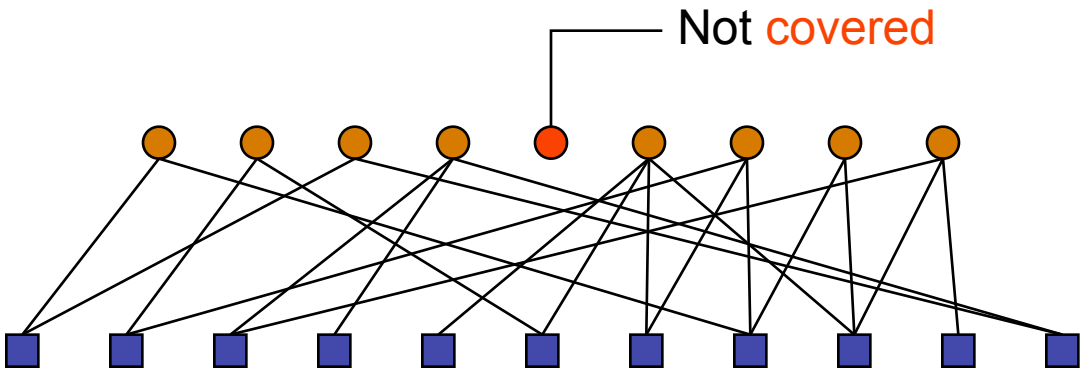
Prob. Decoding error $\geq$ Prob. Non-coverage

$$\geq \left(1 - \frac{1}{k}\right)^{k(1+\varepsilon)\Omega'(1)}$$

$$\simeq e^{-(1+\varepsilon)\Omega'(1)}$$

Average Degree of Distribution should be

$$O(\log(k))$$



$$e^{-(1+\varepsilon)\Omega'(1)} \leq \frac{1}{k} \Leftrightarrow \Omega'(1) \geq \frac{\ln(k)}{1+\varepsilon}$$

Luby has designed universal LT-codes with average degree around $O(\log(k))$ and overhead $O(\log^2(k)/\sqrt{k})$

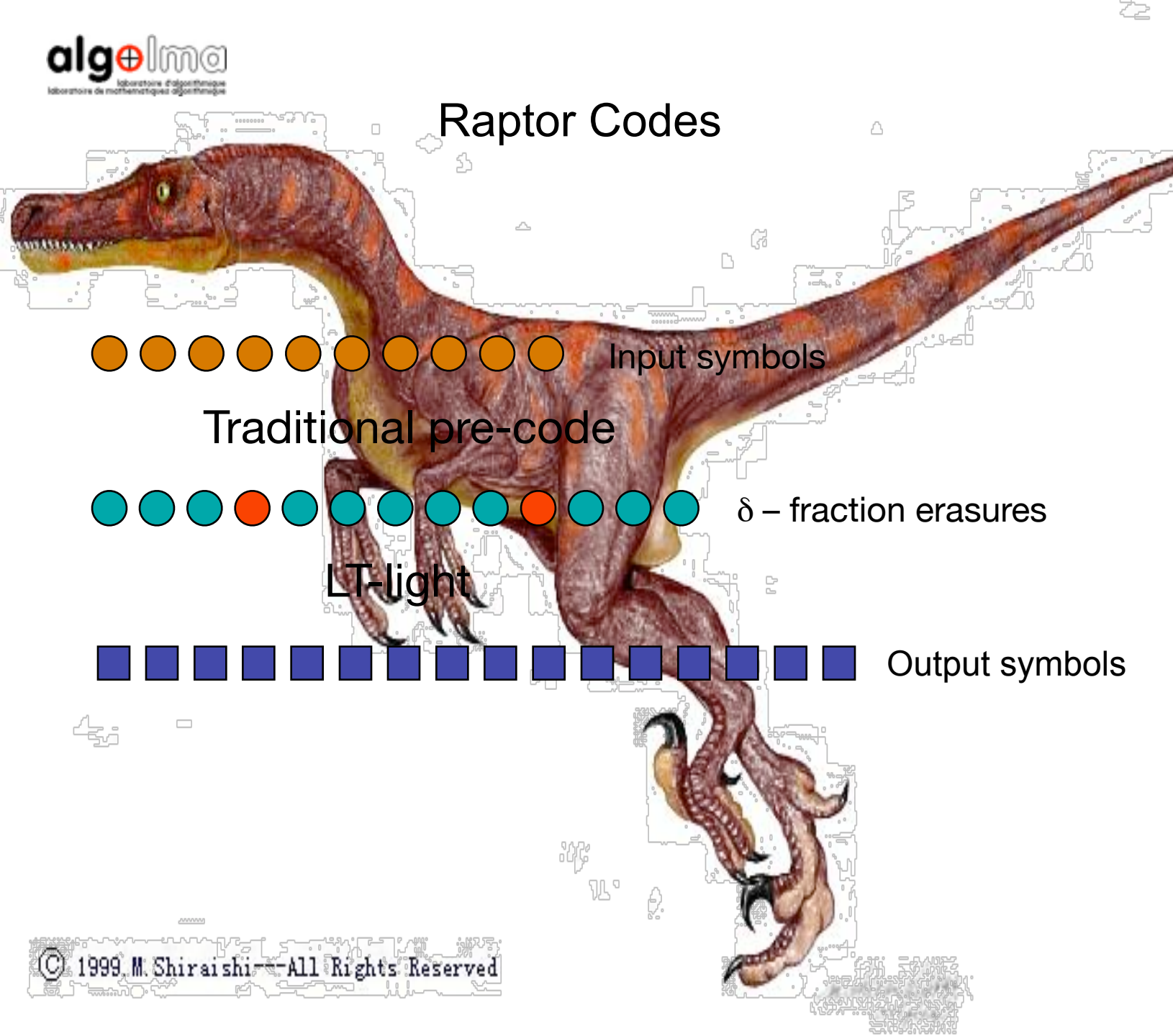
So:

Average degree constant) error probability constant

How can we achieve constant workload per output symbol, and still guarantee vanishing error probability?

Raptor codes achieve this!

Raptor Codes



Input symbols

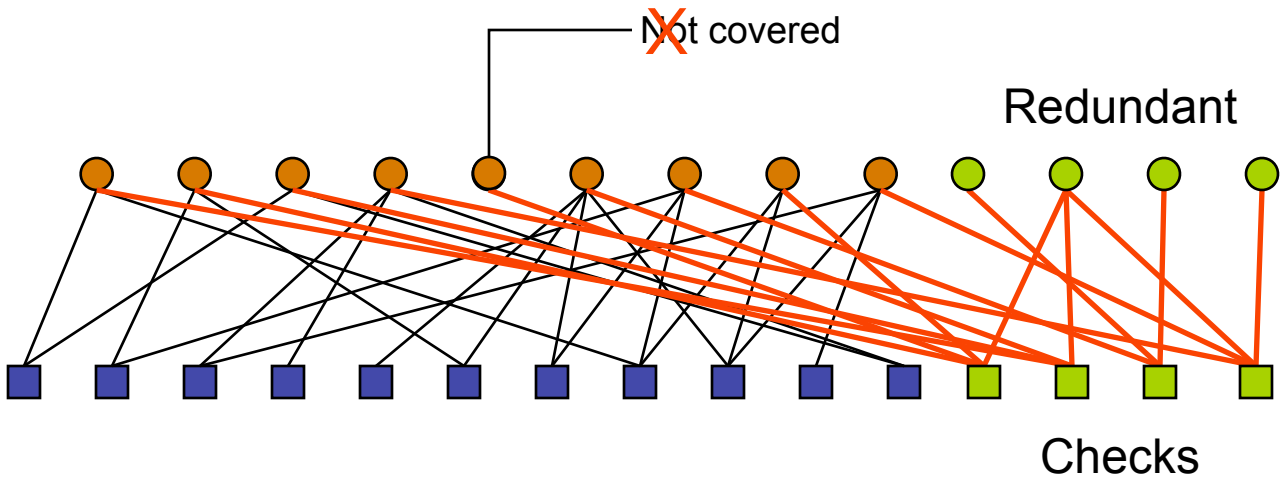
Traditional pre-code

δ – fraction erasures

LT-light

Output symbols

Raptor Codes



If pre-code is chosen properly, then the LT-distribution can have constant average degree, leading to linear time encoding.

Raptor Code is specified by the input length k, precode $\mathcal{C}$ and output distribution $\Omega(x)$.

How do we choose $\Omega(x)$ and $\mathcal{C}$?

Raptor Codes: History

Raptor Codes were invented in late 2000, and patented in 2001.

They were originally designed to solve a speed bottleneck problem for LT-Codes in Digital Fountain's products.

The first preprint of Raptor Codes appeared in late 2002.

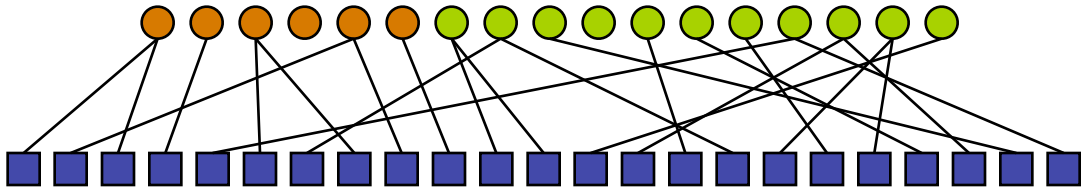
Special Raptor Codes: LT-Codes

LT-Codes are Raptor Codes with trivial pre-code: Need average degree $O(\log(k))$

LT-Codes compensate for the lack of the pre-code with a rather intricate output distribution.

Special Raptor Codes: PCO-Codes

Pre-code-only (PCO) codes:



$$\Omega(x) = x$$

Trivial output distribution, state of the art, very low rate pre-code.

Large computational and storage overhead.

Universal Raptor Codes: Asymptotic Design

Given any δ, want to construct output distribution so that after decoding the residual erasure probability is at most δ.

The pre-code has to be chosen so that it can successfully decode if erasure probability is δ.

Use modified Soliton distribution: Choose $D \sim 1/\delta, \mu \sim 2\delta$

$$\Omega(x) = \frac{1}{\mu+1}\left(\mu x + \frac{1}{2}x^2 + \cdots + \frac{x^D}{D(D-1)} + \frac{x^{D+1}}{D}\right)$$

Pre-code can be chosen to be of rate very close to $1-\delta$

Gives code that can come arbitrarily close to capacity with constant encoding time per symbol, and linear decoding time!

Finite Length Design

Expected number of output symbols of reduced degree 1, when x-fraction of input symbols decoded:

$$k\left(1-x-\mathrm{e}^{-\Omega'(x)(1+\varepsilon)}\right)$$

Solve

$$k\left(1-x-\mathrm{e}^{-\Omega'(x)(1+\varepsilon)}\right) \geq c\sqrt{(1-x)k}$$

Heuristic: Process is like random walk on set of output symbols of reduced degree one, with a binomial distribution.

Output distribution can be optimized using linear programming.

Finite Length Design: Error Probabilities

After a candidate output distribution is obtained, and a precode is determined the error probability of the Raptor Code can be calculated using a combination of

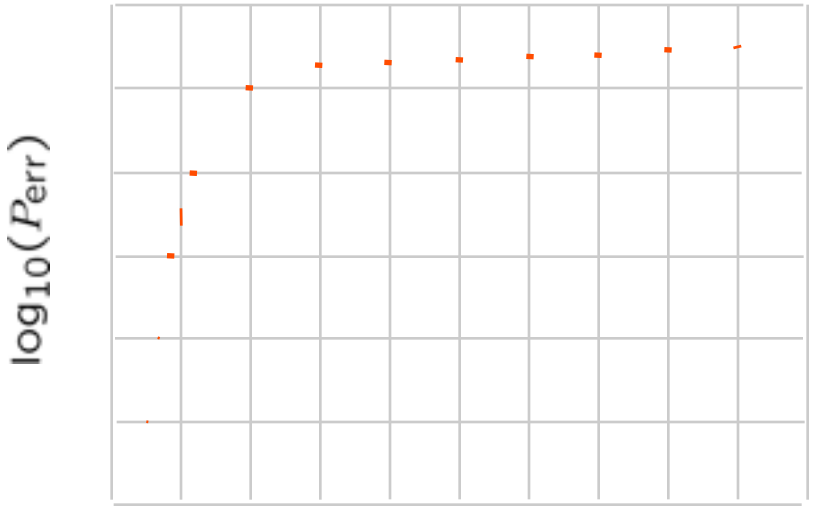
- Finite Length Analysis of LT-Codes [Karp-Luby-S]

- Finite Length Analysis of the pre-code [S-Urbanke for certain pre-codes]

Example:

$$\begin{aligned}\Omega(x) = {} & 0.008x + 0.049x^2 + 0.166x^3 + 0.072x^4 + \\ & 0.083x^5 + 0.056x^8 + 0.037x^9 + 0.056x^{19} + \\ & 0.025x^{66} + 0.003x^{67}\end{aligned}$$

Error probability of LT-decoder



Combined error probability is less than 10^{-14} for overhead less than 2%.

3GPP-MBMS

Digital Fountain is involved in the 3GPP-MBMS standard for Multimedia Broadcast Multicast Service to handsets.

The proposal of Digital Fountain is a Raptor Code suited for a wide range of applications (very small file sizes to large file sizes).

The pre-coder is a multi-stage code consisting of a suitable LDPC and a Hamming code.

The LT-part has been particularly designed for robustness and efficient encoding/decoding (joint work with Michael Luby, and, in part, Andrew Brown).

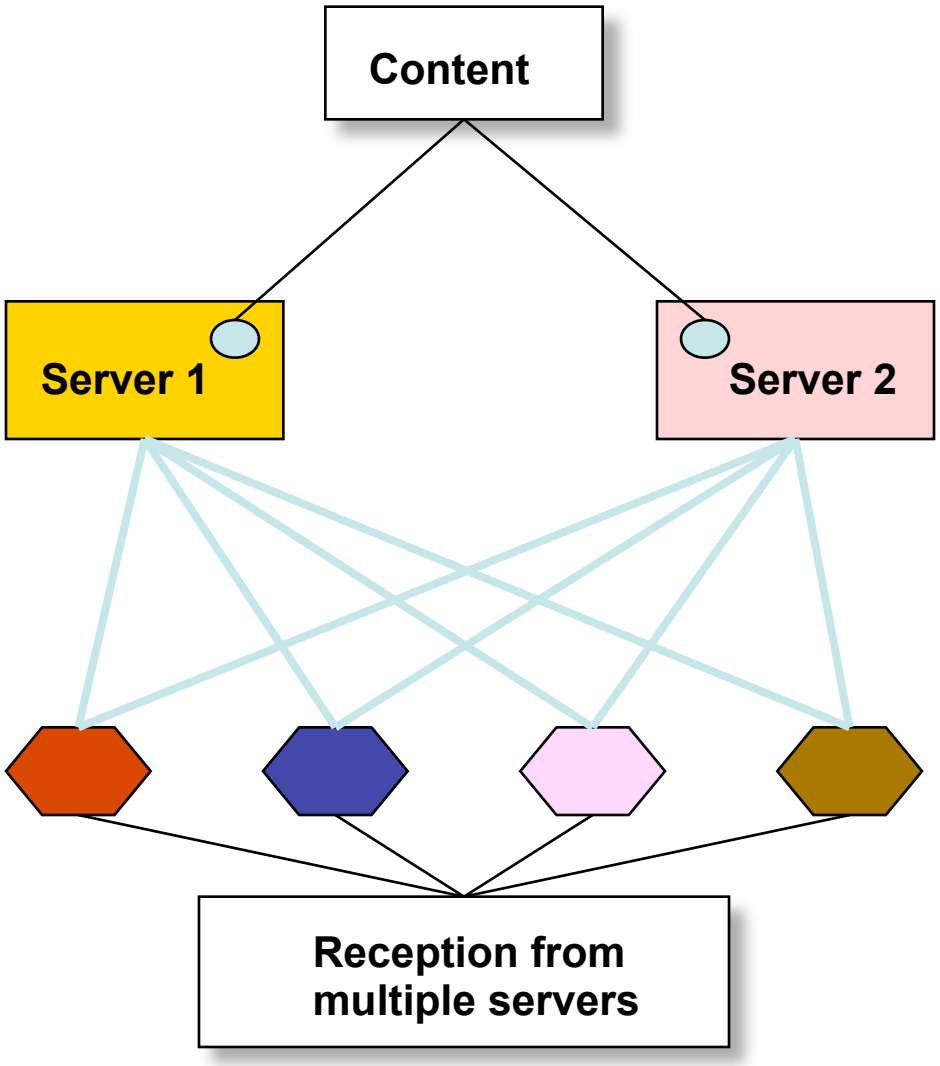
Systematic Codes

Systematic versions of Raptor Codes are sometimes desirable.

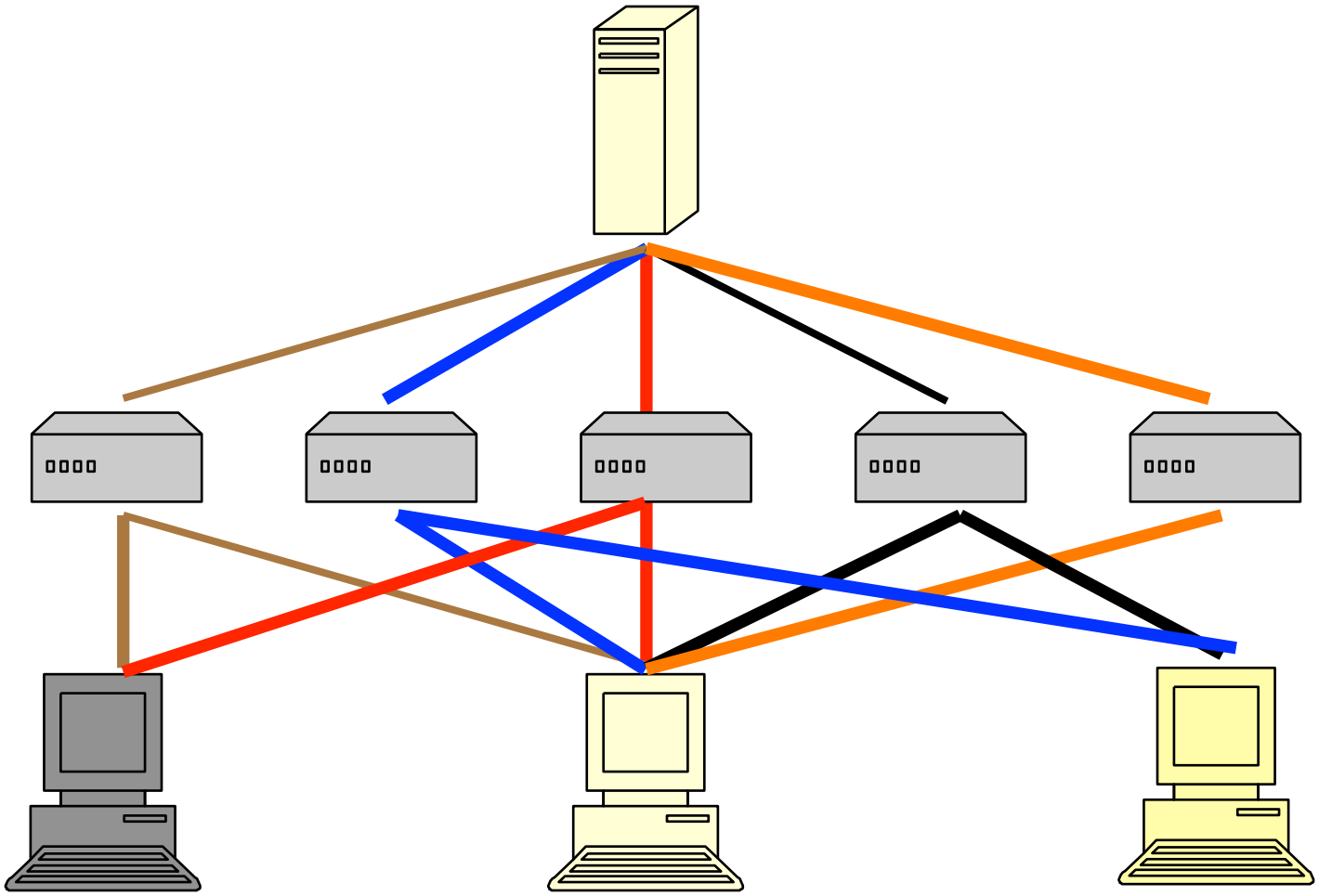
The trivial way of making these codes systematic does not work.

However, there is a method to make these codes systematic.

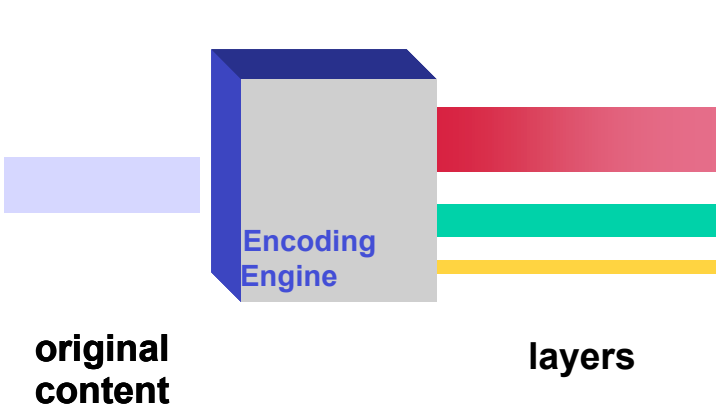
Applications: Multi-site downloads

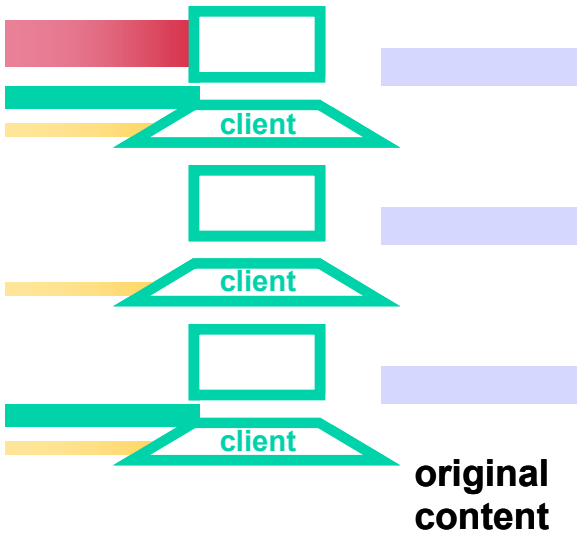


Applications: Path Diversity

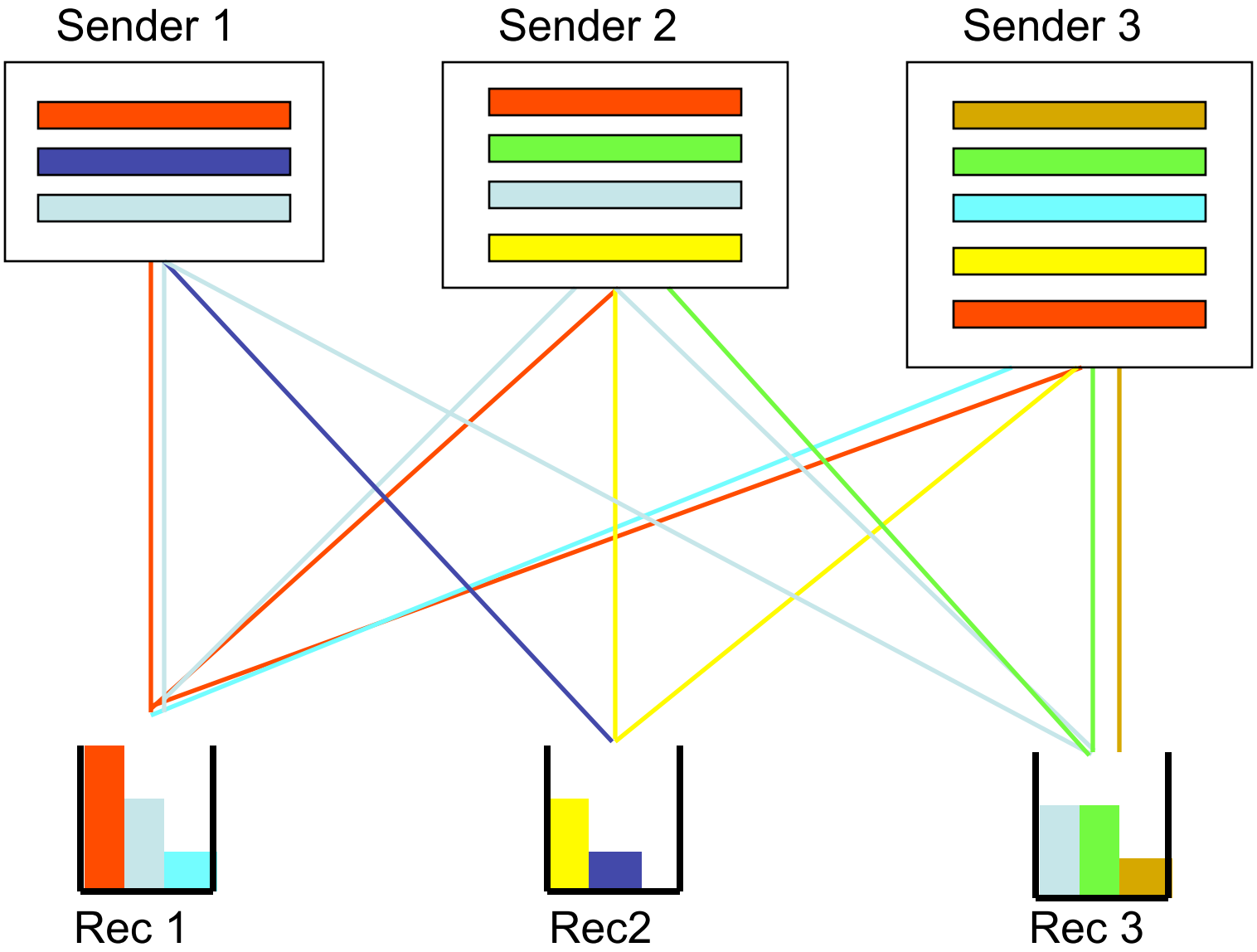


Applications: Congestion Control





Applications: Peer-2-Peer



algolma
laboratoire d'algorithmique
laboratoire de mathematiques algorithmique

BIMSC

Content

- How do Raptor Codes designed for the BEC perform on other symmetric channels (with BP decoding)?
- Information theoretic bounds, and fraction of nodes of degrees one and two in capacity-achieving Raptor Codes.
- Some examples
- Applications

Parameters

Raptor Code with parameters $(k, \mathtt{C}, \Omega(x))$.

Channel $\mathcal{C}$.

Overhead ε, if decoding is possible from $\frac{k(1+\varepsilon)}{\mathsf{Cap}(\mathcal{C})}$ many output symbols.

Measure residual error probability as a function of the overhead for a given channel.

Incremental Redundancy Codes

Raptor codes are true incremental redundancy codes.

A sender can generate as many output bits as necessary for successful decoding.

Suitably designed Raptor codes are close to the Shannon capacity for a variety of channel conditions (from very good to rather bad).

Raptor codes are competitive with the best LDPC codes under different channel conditions.

Sequences Designed for the BEC

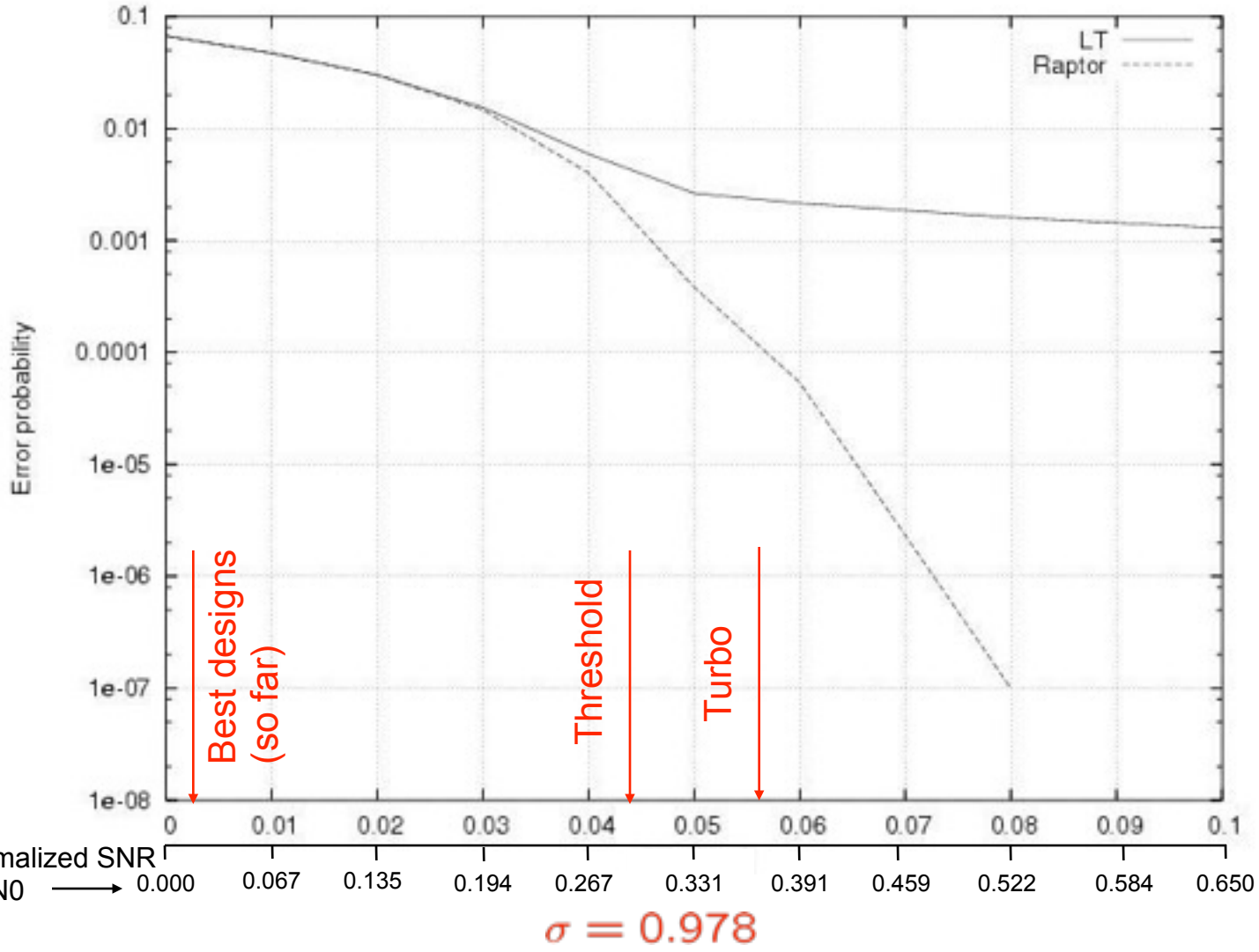
Type: $(65536, \mathtt{C}, \Omega(x))$

$\mathtt{C}$: Left-regular of degree 4, right Poisson, rate 0.98

$$\begin{aligned}\Omega(x) \;=\; & 0.008x + 0.049x^2 + 0.166x^3 + 0.072x^4 + \\ & 0.083x^5 + 0.056x^8 + 0.037x^9 + 0.056x^{19} + \\ & 0.025x^{66} + 0.003x^{67}\end{aligned}$$

Simulations done on AWGN(σ) for various σ

Sequences Designed for the BEC



Sequences Designed for the BEC

Not too bad, but quality decreases when the amount of noise on the channel increases.

Need to design better distributions.

Idea: adapt the Gaussian approximation technique of Chung, Richardson, and Urbanke.

Gaussian Approximation

Assume that the messages sent from input to output nodes are Gaussian.

Track the mean of these Gaussians from one round to another.

$$\varphi(\mu_{\ell+1}) \leq 1 - s\omega(1 - \varphi(\alpha\mu_\ell)).$$

Degree distributions can be designed using this approximation.

However, they don't perform that well.

Anything else we can do with this?

Nodes of Degree 2

Use equality, differentiate, and compare values at 0

$$\Omega_2 \geq \frac{\mathrm{Cap}(\mathrm{BIAWGN}(\sigma))}{2\mathrm{E}(\mathrm{BIAWGN}(\sigma))} = \frac{\Pi(\mathcal{C})}{2}.$$

where

$$\mathrm{E}(\mathrm{BIAWGN}(\sigma)) = \frac{1}{\sqrt{4\pi m}} \int_{-\infty}^{\infty} \tanh\left(\frac{x}{2}\right) \mathrm{e}^{-\frac{(x-m)^2}{4m}} \mathrm{d}x$$

and $m = \dfrac{2}{\sigma^2}$.

It can be rigorously proved that above condition is necessary for error probability of BP to converge to zero.

Nodes of Degree 2

What is the fraction of nodes of degree 2 for capacity-achieving Raptor Codes?

Turns out, that in the limit we need to have equality:

$$\Omega_2(\mathcal{C}) := \frac{1}{2}\Pi(\mathcal{C})$$

where, in general

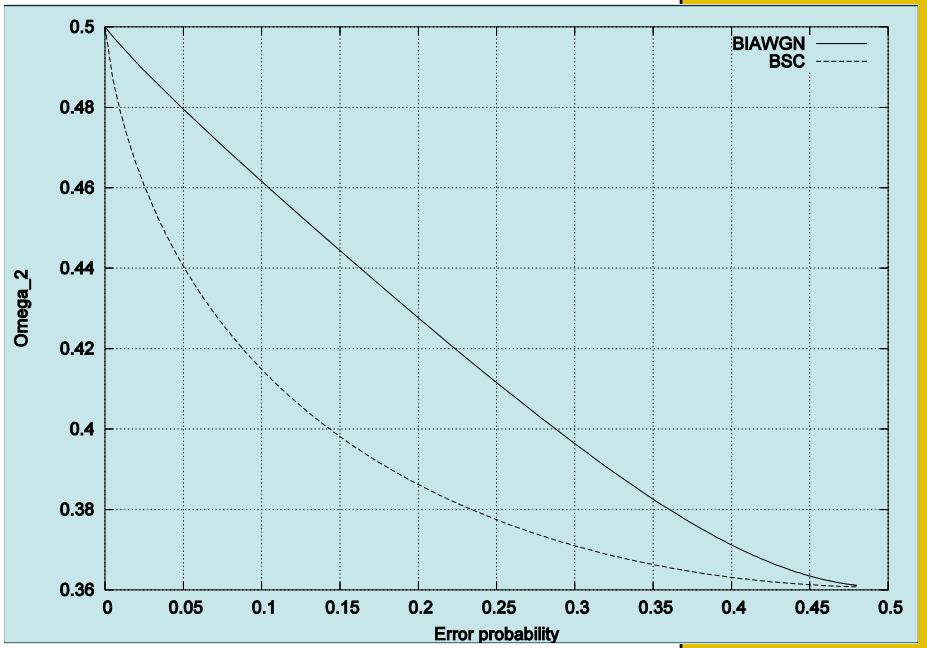
$$\Pi(\mathcal{C}) := \frac{\mathrm{Cap}(\mathcal{C})}{\mathrm{E}[\tanh(Z/2)]}$$

and Z is the LLR of the channel.

$\Omega_2(\mathcal{C})$

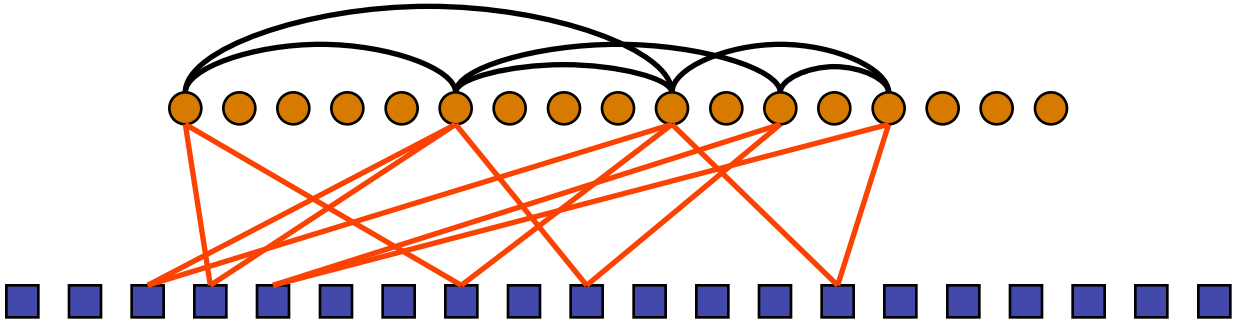
$$\Omega_2(\mathrm{BEC}(p)) = \frac{1}{2}$$

$$\Omega_2(\mathrm{BSC}(p)) = \frac{1 - h(p)}{2(1-2p)^2}$$



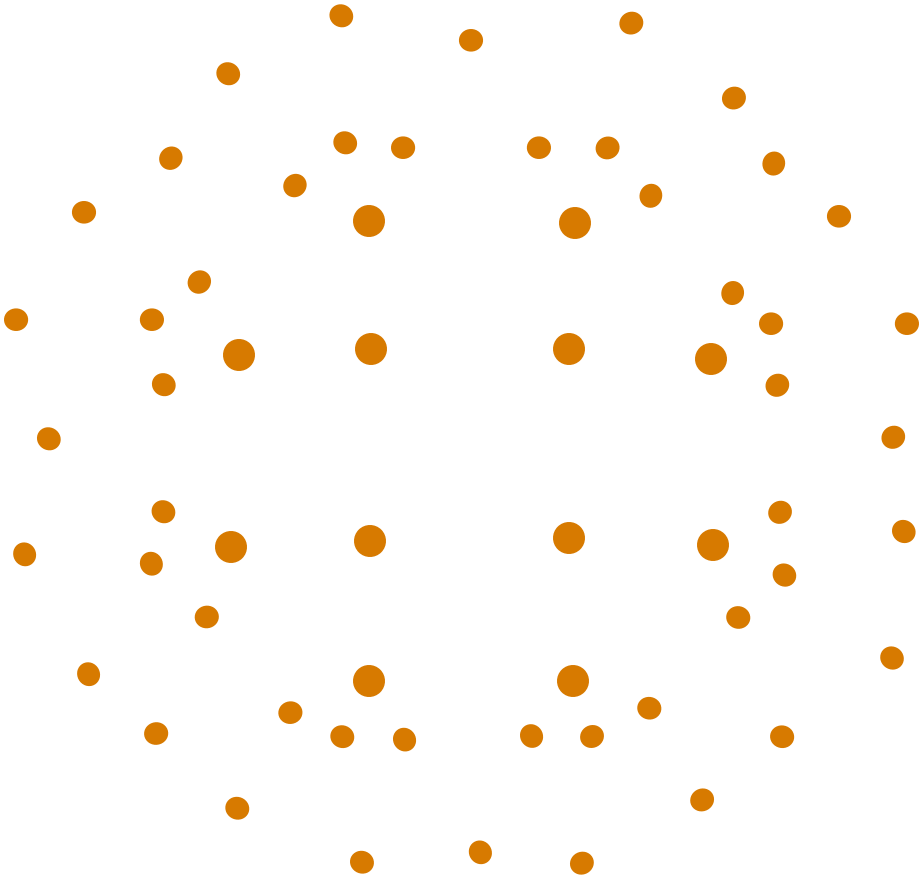
$$\Omega_2(\mathrm{BIAWGN}(\sigma)) = \frac{1}{4\ln(2)}(1 + O(m)).$$

Nodes of Degree 2

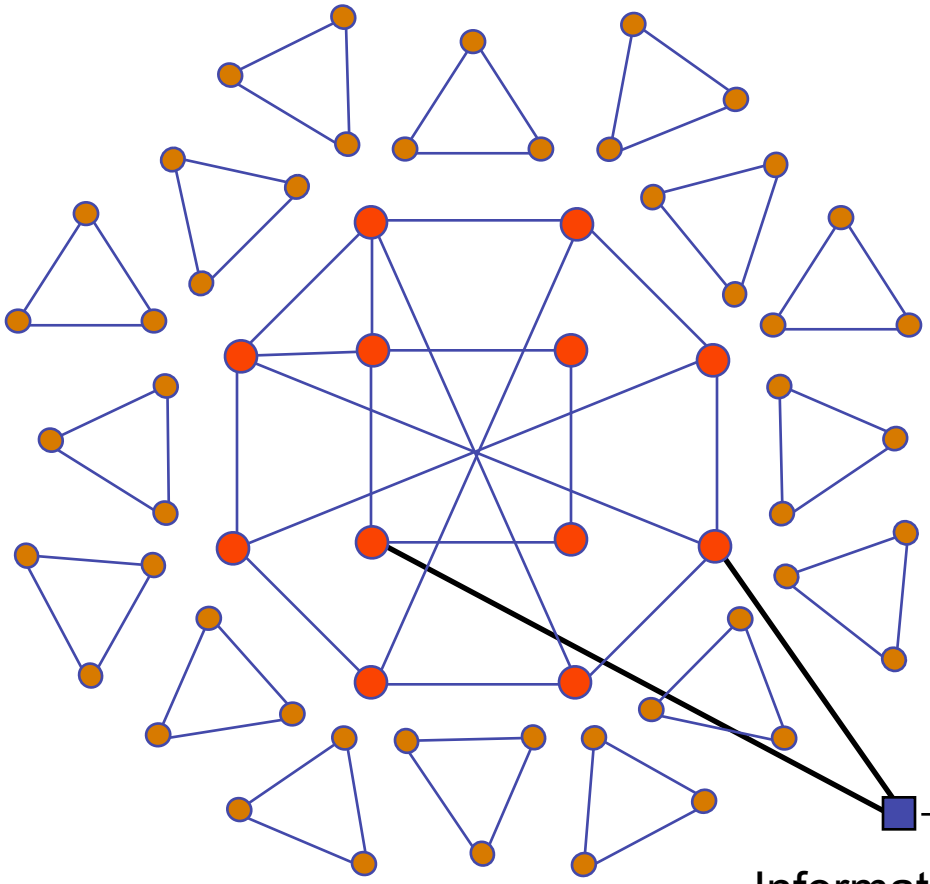


Use graph induced on input symbols by output symbols of degree 2.

Nodes of Degree 2: BEC



Nodes of Degree 2: BEC



New output node of degree 2

Information Loss!

Fraction of Nodes of Degree 2

If there exists component of linear size (i.e., a giant component), then next output node of degree 2 has constant probability of being useless.

Therefore, graph should not have giant component.

This means that for capacity achieving degree distributions we must have: $\Omega_2 \leq \frac{1}{2}$.

On the other hand, if $\Omega_2 < \frac{1}{2}$ then algorithm cannot start successfully.

So, $\Omega_2 = \frac{1}{2}$ for capacity-achieving codes:

General Symmetric Channels: Mimic Proof

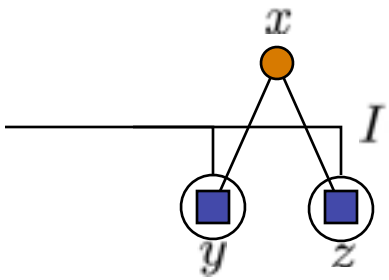
Proof is information theoretic: if fraction of nodes of degree 2 is larger by a constant, then :

- Expectation of the hyperbolic tangent of messages passed from input to output symbols at given round of BP is larger than a constant.
- This shows that $I(x; z_2) < n\Omega_2(\mathsf{Cap}(\mathcal{C}) - \tau)$
- So code cannot achieve capacity.

General Symmetric Channels: Mimic Proof

Fraction of nodes of degree one for capacity-achieving Raptor Codes:

Noisy observations of x



$$I(x; y, z) < I(x; y) + I(x; z)$$

Therefore, if $\Omega_1 > 0$, and if $z_1, \ldots, z_m$ denote output nodes of degree one, then

$$I(x; z_1, \ldots, z_m) < \sum_{i=1}^{m} \mathsf{Cap}(\mathcal{C}) - \eta$$

So $\Omega_1 = 0$

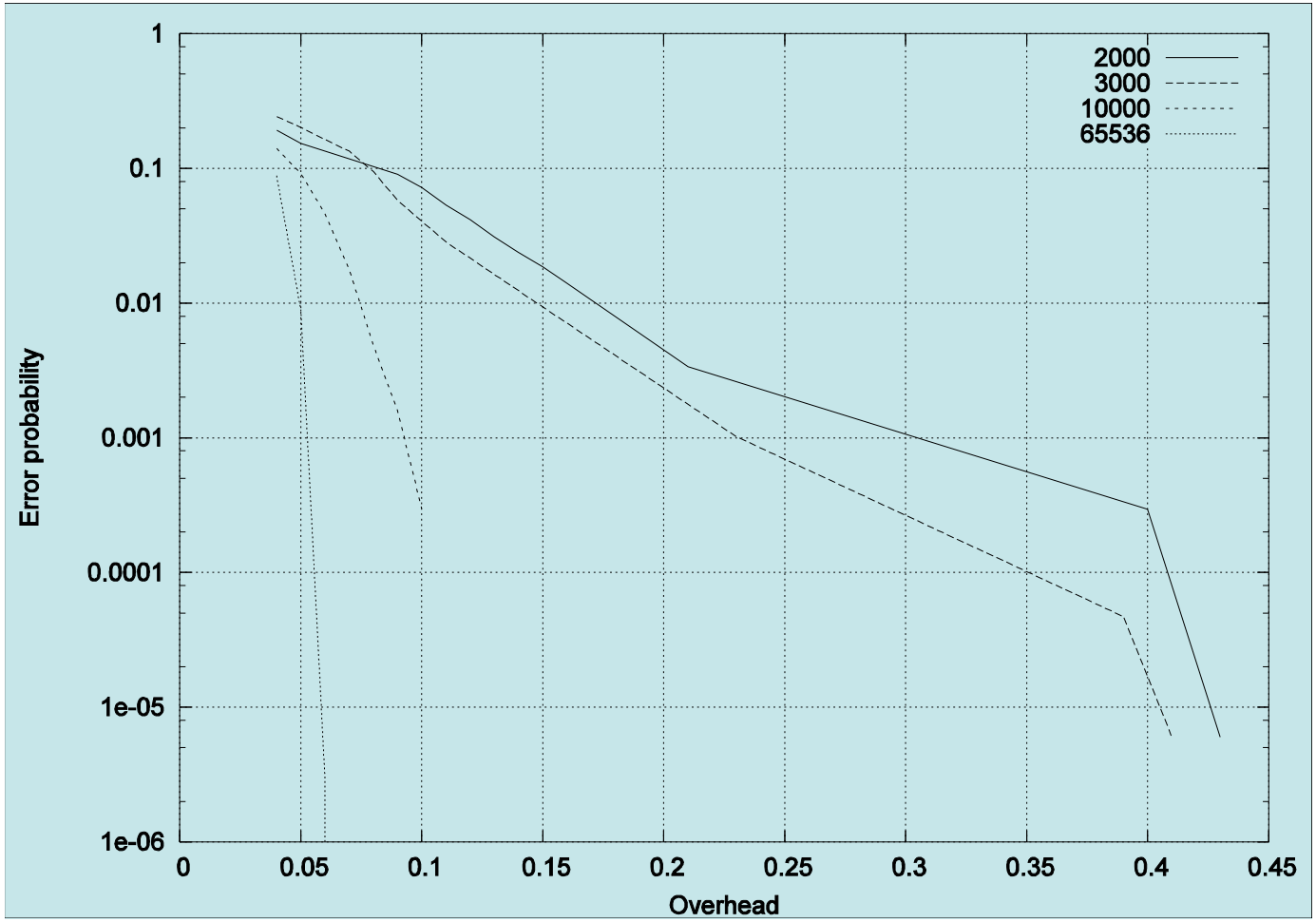
A better Gaussian Approximation

Uses adaptation of a method of Ardakani and Kschischang.

Heuristic: Messages passed from input to output symbols are Gaussian, but not vice-versa.

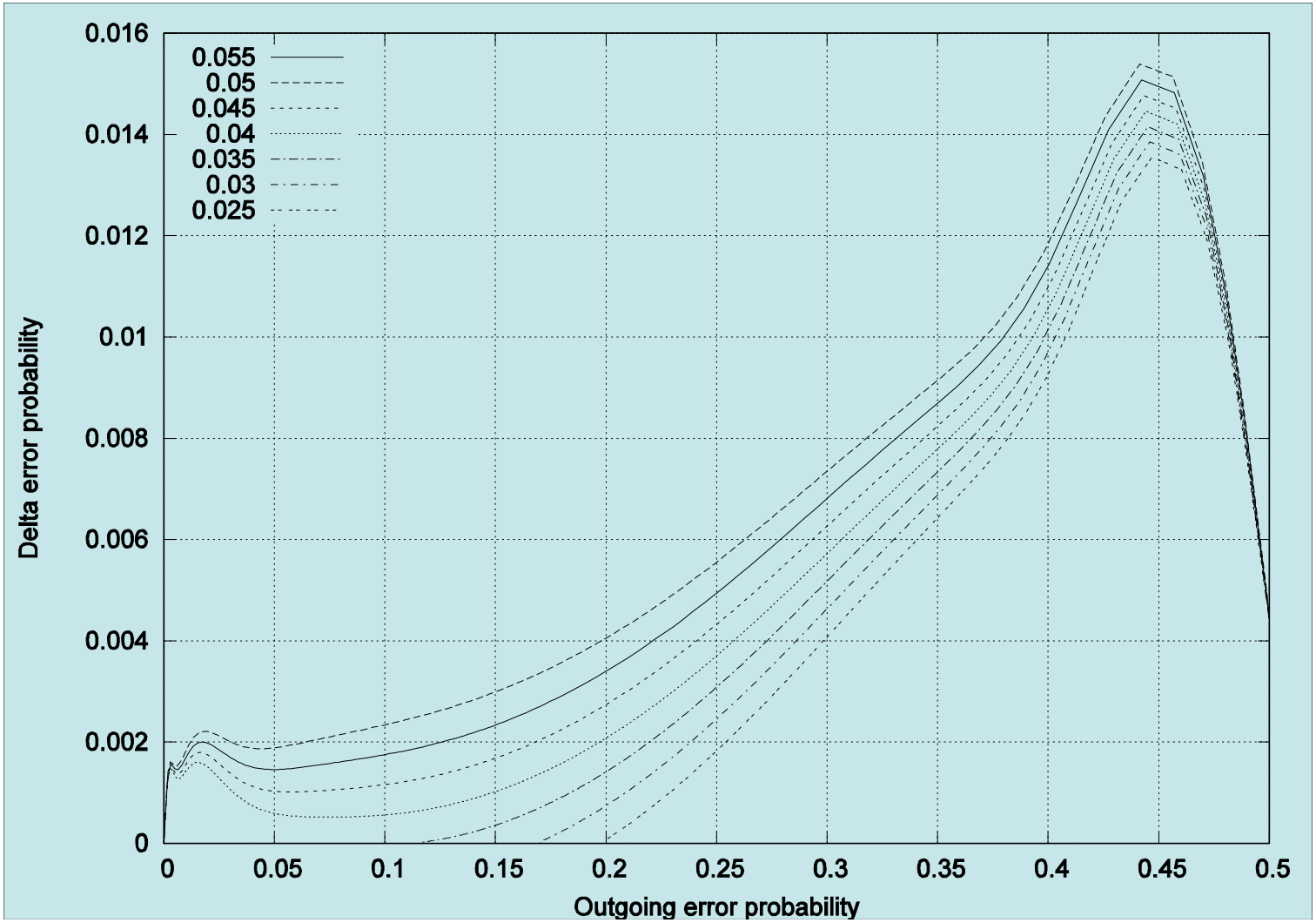
Find recursion for the means of these Gaussians, and apply linear programming.

$$\begin{aligned}\Omega(x) &= 0.006x + 0.492x^2 + 0.0339x^3 + 0.2403x^4 + \\ &\quad 0.2403x^5 + 0.095x^8 + 0.049x^{14} + 0.018x^{30} + \\ &\quad 0.036x^{33} + 0.033x^{200}\end{aligned}$$


1
0.1
0.01
0.001
0.0001
1e-05
1e-06
Error probability
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
Overhead
2000
3000
10000
65536

$\sigma = 0.5$

Density Evolution



Other Applications

Raptor codes have a variety of applications, some outside the normal communications scenario.

For example, they can be used to perform lossless compression (joint work with Caire, Shamai, and Verdu).

The algorithm can be used to perform joint source channel coding.

Conclusions

- Raptor Codes can be adapted to general symmetric channels using the Belief Propagation algorithm.
- Raptor Codes designed for the BEC are not bad on other channels, but their performance can be improved.
- There are no universal codes on channels other than the BEC, as the fraction of degree 2 nodes in capacity-achieving codes depends on the channel noise.
- General design techniques can be adapted to design good Raptor Codes that perform very well under a variety of channel conditions.

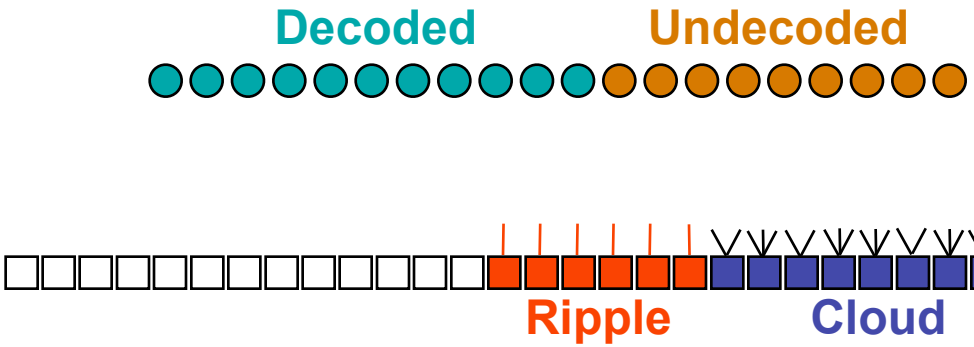
Finite Length Analysis

Given LT-code with parameters $(k, \Omega(x))$, calculate the error probability of the belief propagation decoder given a random subset of the output symbols.

Stopping sets: seems to be difficult in this case.

Alternative: detailed analysis of the decoder.

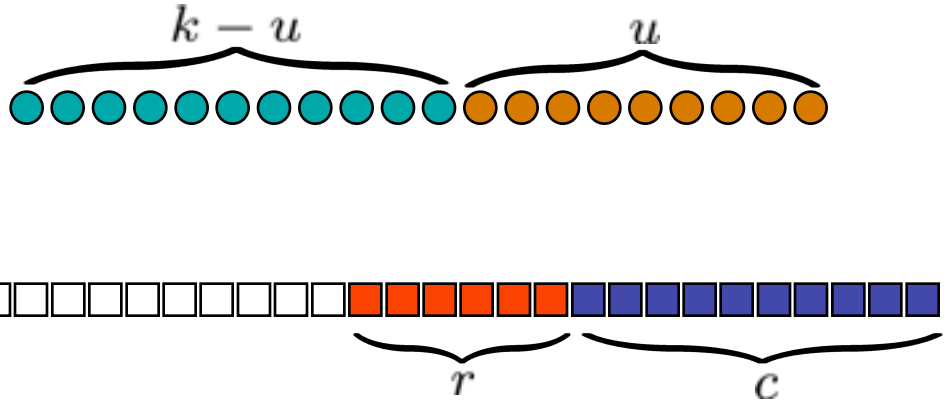
- Define the *state* of the decoder,
- Derive a recursion for the generating function of the states,
- Explain how to calculate these efficiently.



Ripple = Set of output symbols of reduced degree 1

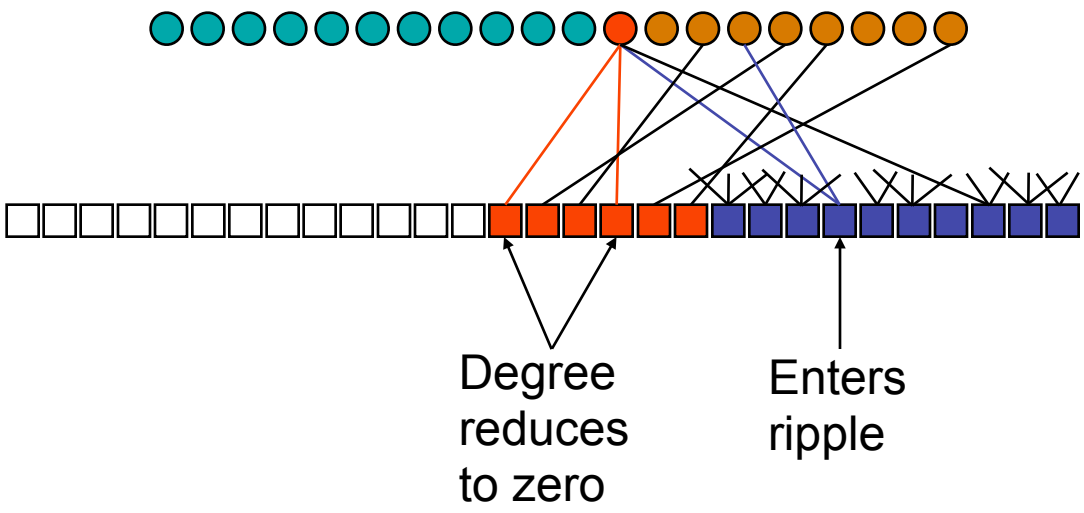
Cloud = Set of output symbols of reduced degree > 1

Decoder is successful iff ripple is not empty until the end



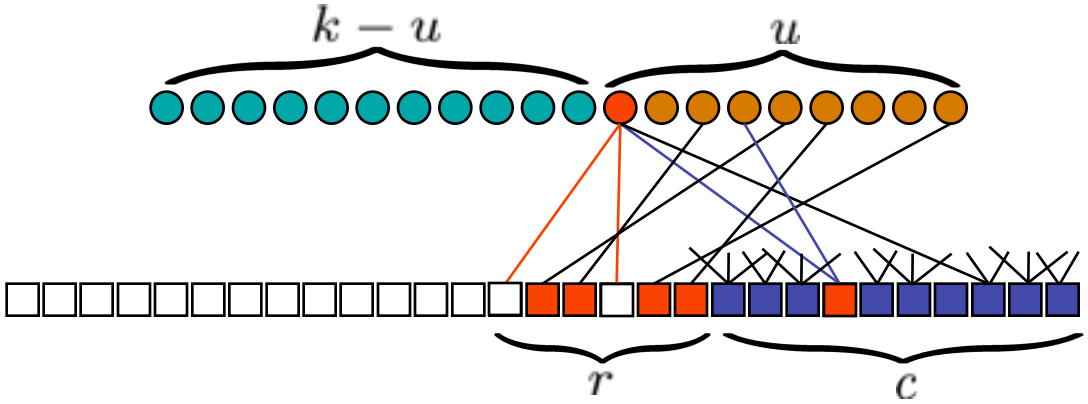
Decoder is in state (u, r, c) if there are r elements in the ripple and c elements in the cloud, when there are u undecoded input symbols left.

$\Pi_u(r, c) =$ Probability that decoder is in state (u, r, c)



At each point, at least one ripple element disappears, and 0 or more cloud elements enter the ripple.

What is the distribution of these numbers?



$$\Pi_{u-1}(r',c') = \sum_{r,c} \binom{c}{c'} p_u^{c-c'} (1-p_u)^{c'} \binom{r-1}{r'-c+c'} \left(\frac{1}{u}\right)^{r-1-r'+c-c'} \left(1-\frac{1}{u}\right)^{r'-c+c'}$$

What is p_u?

Probability that a randomly chosen input symbol reduces its degree to one exactly when the input symbol is decoded.

$\mathcal{A}$ = Event that a random input symbol is in the cloud when u input symbols undecoded.

$\mathcal{B}$ = Event that a random input symbol is of reduced degree one after the decoding step.

$$p_u = \Pr[\mathcal{B}|\mathcal{A}] = \frac{\Pr[\mathcal{B}\&\mathcal{A}]}{\Pr[\mathcal{A}]}.$$

Allowing Multiple Edges

If we allow multiple edges, the expression for p_u becomes simpler:

$$
\begin{aligned}
p_u &= \frac{\frac{u}{k}\left(\Omega'\left(1-\frac{u}{k}\right)-\Omega'\left(\frac{u-1}{k}\right)\right)}{1-\frac{u}{k}\Omega'\left(1-\frac{u}{k}\right)-\Omega\left(1-\frac{u}{k}\right)} \\
&\sim \frac{1}{k}\cdot\frac{u}{k}\cdot\frac{\Omega''\left(1-\frac{u}{k}\right)}{1-\frac{u}{k}\Omega'\left(1-\frac{u}{k}\right)-\Omega\left(1-\frac{u}{k}\right)}.
\end{aligned}
$$

The Recursion

State generating function:

$$P_u(x, y) := \sum_{c,r,r\geq 1} \Pi_u(r, c) x^c y^{r-1}$$

Formula gives recursion

$$P_{u-1}(x, y) = \frac{P_u\left(x(1-p_u) + yp_u, \frac{1}{u} + y\left(1 - \frac{1}{u}\right)\right) - P_u\left(x(1-p_u), \frac{1}{u}\right)}{y}.$$

Error probability is equal to $1 - P_u(1, 1)$

Error probability at every step can be calculated precisely.
In practice, it is sufficient to find a good approximation.

Computational Cost

Normal calculation:

Upate the $O(n^2)$ coefficients of $P_u(x, y)$ in time $O(n^2)$

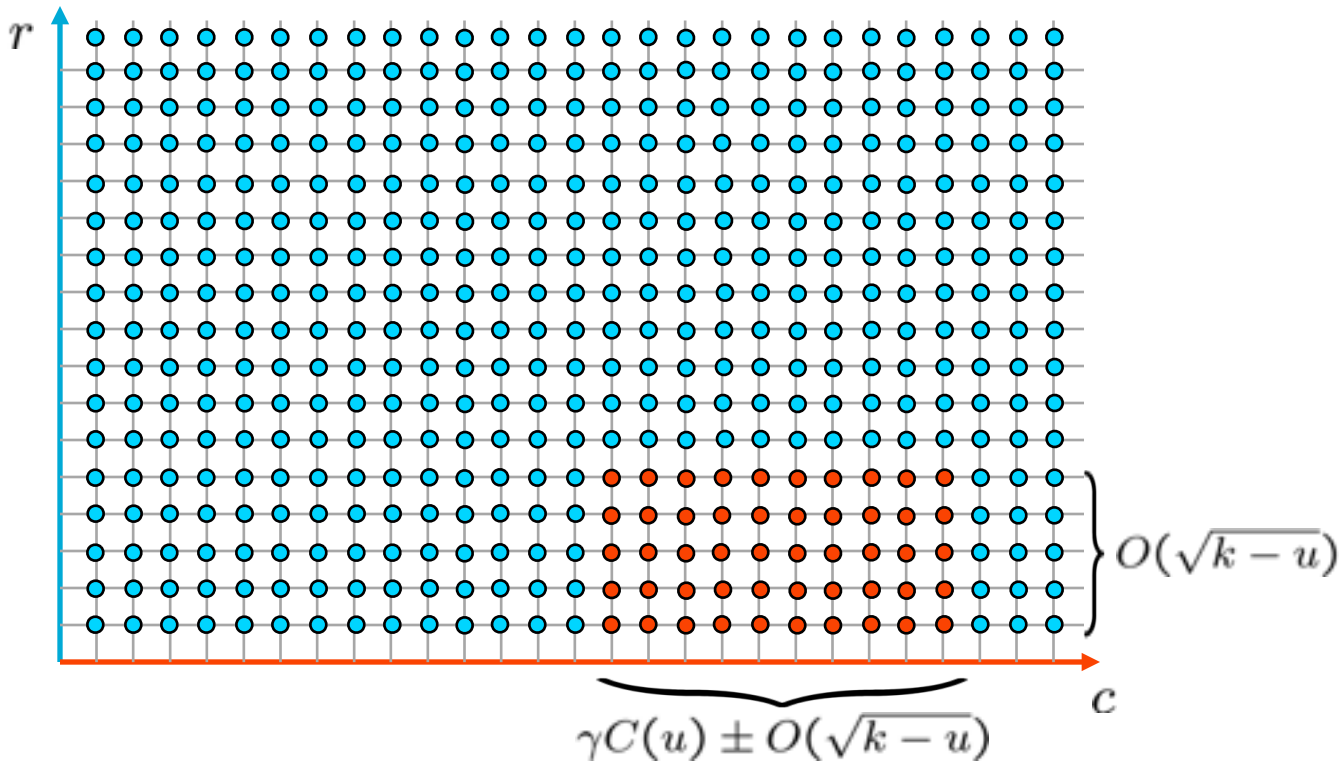
$O(n)$ update steps, so in total $O(n^5)$ operations.

Fast interpolation:

$O(n^3 \log^2(n) \log\log(n))$ Operations.

Approximate techniques?

Approximation Algorithms



Concentrate only on **hot** zones) $O(n^2 \text{polylog}(n))$

Expected Cloud and Ripple Size

$$C(u) = \frac{\partial}{\partial x} P_u(x,y) \Big|_{(x,y)=(1,1)}$$

$$R(u) = \frac{\partial}{\partial y} P_u(x,y) \Big|_{(x,y)=(1,1)}$$

Recursions:

$$C(u-1) = C(u)(1-p_u) - \quad \text{» 0}$$

$$R(u-1) = p_u C(u) + \left(1 - \frac{1}{u}\right) R(u) - \quad \text{» 1}$$

Approximate Solutions

Translate into difference equation, and then differential equation:

$$x = \frac{u}{k} \begin{cases} C(x) & \sim & 1 - x\Omega'(1-x) - \Omega(1-x), \\ \\ R(x) & \sim & x\Omega'(1-x) + x\ln(x) \end{cases}$$

The approximations are due to finite length (with an O(1/k) error), and due to the approximation of the drift terms.

These are the same solutions as obtained by the tree analysis (density evolution).

The drift terms explain why the tree analysis is not exact.

Second Moment Recursions

$$F(u) := \frac{\partial^2 P_u}{\partial x^2}(1,1)$$

$$G(u) := \frac{\partial^2 P_u}{\partial x \partial y}(1,1)$$

$$H(u) := \frac{\partial^2 P_u}{\partial y^2}(1,1)$$

Second Moment Recursions

$$
\begin{aligned}
F(u-1) &\sim F(u)(1-p_u)^2 \\
G(u-1) &\sim \left(F(u)p_u + G(u)\left(1-\frac{1}{u}\right)\right)(1-p_u) - \\
&\quad C(u)(1-p_u) \\
H(u-1) &\sim \left(F(u)p_u + G(u)\left(1-\frac{1}{u}\right)\right)p_u + \\
&\quad \left(G(u)p_u + H(u)\left(1-\frac{1}{u}\right)\right)\left(1-\frac{1}{u}\right) - \\
&\quad 2C(u)p_u - 2R(u)\left(1-\frac{1}{u}\right) + 2
\end{aligned}
$$

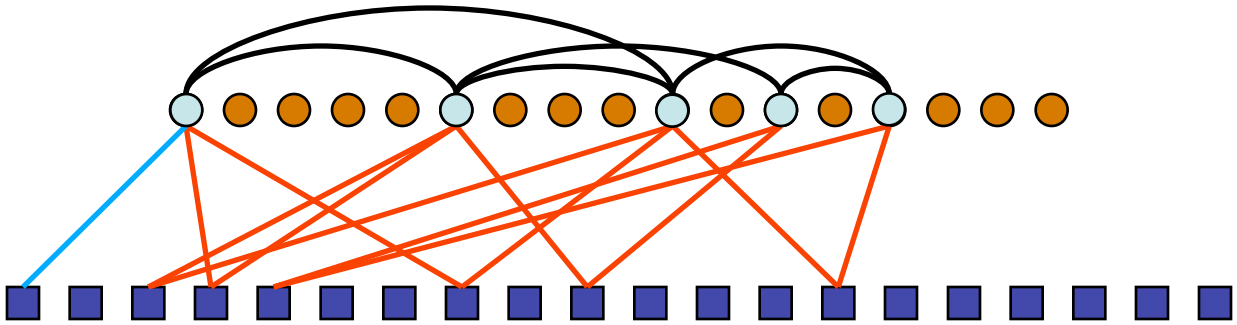
Second Moment Recursions

Using the second moment recursions an approximate evolution of the variance can be calculated.

The real values of the cloud and the ripple size are within a tube of the expected values, and the size of the tube can be calculated from the second (and higher moment) recursions.

Progressive Giant Component Analysis

A different method for the analysis of the decoder:



Want enough nodes of degree 2 so there exists a giant component in the induced (random) graph on input symbols.

Analysis yields Soliton distribution in the limit where average degree of induced graph is 1.

Progressive Giant Component Analysis

Analysis can be used to obtain error bounds for the decoding algorithm.

It can also be used to obtain capacity-achieving distributions on the erasure channel.

A modified version can be used to obtain Ω_2 for capacity-achieving distributions for other symmetric channels.