Raptor Codes: From a Math Idea to LTE eMBMS

BIRS, October 2015

The plan is to ...

1. introduce LT codes and Raptor codes
2. provide insights into their design
3. address some common misconceptions

The Ideal Abstract Properties of Codes

1. The encoder should be able to generate, from k source symbols, as many encoded symbols as required for decoding. $\leftarrow$ rateless
2. ANY k encoded symbols should be sufficient for decoding.
3. Encoding/Decoding computation time should be linear in k.

Raptor Codes are Good for MBMS Because

They enable reliable communications over multiple, unknown channels:

1. They are rateless $\Rightarrow$ their redundancy can be flexibly adapted to changing channel/network conditions of e.g. as in mobile wireless.
2. In the case of multiple erasure channels as in Multimedia Broadcast/Multicast Service (MBMS), they can be made to universally achieve the channel capacity for all erasure rates.

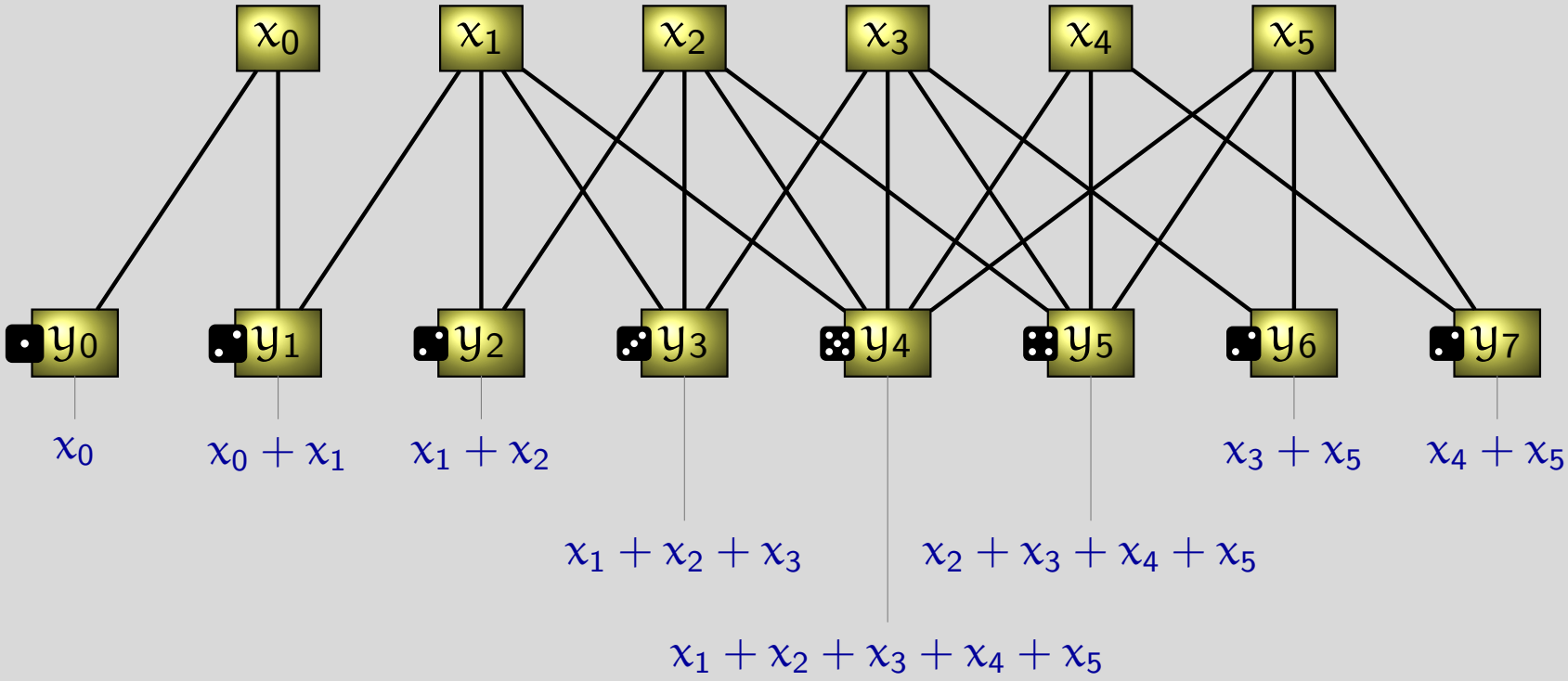
They are simple to encode and decode.

How do Raptor Codes achieve that, and can other codes perform as well?

Raptor Codes are Concatenated Codes

In the beginning, there were LT codes ...

LT Codes – Rateless Encoding



When can we hope to be able to recover the x values from y values?

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

Linear (Rateless) Codes

- Data symbols $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$.
- y_i are linear combinations of $\{x_1, \ldots, x_k\}$, $x_j, y_i \in \mathbb{F}_q$:

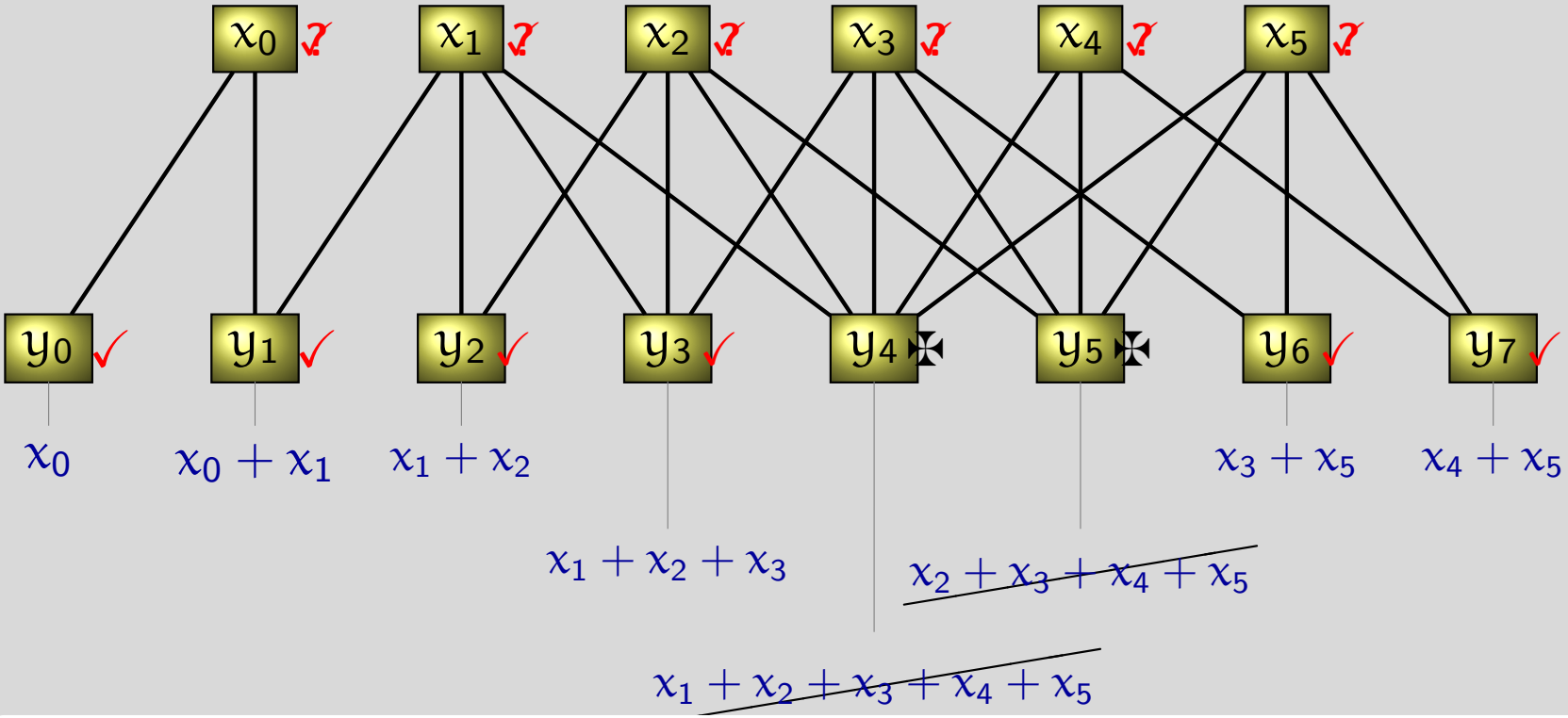
$$y_i = \alpha_1^i x_1 + \cdots + \alpha_k^i x_k, \quad \alpha_j^i \in \mathbb{F}_q$$

- For random codes α_i are chosen uniformly at random.
- For each y_i in LT codes,
  1. degree d is picked (non-uniformly) at random from $\{1, \ldots, k\}$, according to some degree distribution $\{p_1, \ldots, p_k\}$;
  2. only d of $\{\alpha_1^i, \ldots, \alpha_k^i\}$ are non-zero; picked uniformly at random.
  3. the values for the non-zero α_i are chosen uniformly at random.

  To design an LT code means to pick the degree distribution.

What are the unique advantages of LT Codes?

LT Codes – Simple (Belief Propagation) Decoding



Complexity is determined by the number of edges in the decoding graph.

face	⚀	⚁	⚂	⚃	⚄	⚅
Ω	.05	.5	.1	.05	.25	.05

LT Codes Design Objectives

- Data symbols $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$.
- y_i are linear combinations of $\{x_1, \ldots, x_k\}$, $x_j, y_i \in \mathbb{F}_q$:

$$y_i = \alpha_1^i x_1 + \cdots + \alpha_k^i x_k, \quad \alpha_j^i \in \mathbb{F}_q$$

- For each y_i in LT codes,
  1. degree d is picked (non-uniformly) at random from $\{1, \ldots, k\}$, according to some degree distribution $\{p_1, \ldots, p_k\}$;
  2. only d of $\{\alpha_1^i, \ldots, \alpha_k^i\}$ are non-zero; picked uniformly at random.
  3. the values for the non-zero α_i are chosen uniformly at random.

The distribution should enable simple linear time decoding of $\{x_1, \ldots, x_k\}$ as soon as any $k(1+\epsilon)$ of y-s are received.

Can such a degree distribution be found?

Can LT Codes Meet the Ideal Design Objectives?

The Ideal Abstract Properties of Codes:

1. The encoder should be able to generate, from k source symbols, as many encoded symbols as required for decoding. $\leftarrow$ rateless
2. ANY k encoded symbols should be sufficient for decoding.
3. Encoding/Decoding computation time should be linear in k.

vs.

LT Codes Design

1. Data symbols $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$ by an algorithm using a degree (probability) distribution $\{p_1, \ldots, p_k\}$.
2. The distribution should enable simple linear time decoding of $\{x_1, \ldots, x_k\}$ as soon as any $k(1+\epsilon)$ of y-s are received.

Can Complexity be Linear ...

if data symbols are recoverable from any $k(1+\epsilon)$ code symbols?

- Data symbols $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$ by an algorithm using a degree (probability) distribution $\{p_1, \ldots, p_k\}$.
- We may be able to recover $\{x_1, \ldots, x_k\}$ from some n y's only if
  1. $n \geqslant k \Leftrightarrow$ more equations than unknowns
  2. there are $\mathcal{O}(k \log k)$ edges incident to y's
     $\Leftrightarrow$ each x is in at least one equation whp?
- Complexity is determined by the number of edges incident to y's but only for BP decoding; otherwise a matrix inversion is required.

A Polya-Like Urn Model & Coding Overhead

An urn contains k balls that are numbered from 1 to k.

A multiple-ball draw from the urn is carried out in two stages:

1. Number i of balls to be drawn is selected with probability p_i.
2. i balls are drawn without replacements, their numbers are recorded, and the balls are then returned to the urn.

The number of draws necessary to see all balls is, on average,

$$\frac{1}{a_1} k H_k + \frac{a_1 - a_2}{2a_1^2}(H_k - 1),$$

where a_i is the i-th moment of $\{p_i\}_{i=1}^k$ and $H_k = \sum_{\ell=1}^{k} \frac{1}{\ell} = \mathcal{O}(\log k)$.

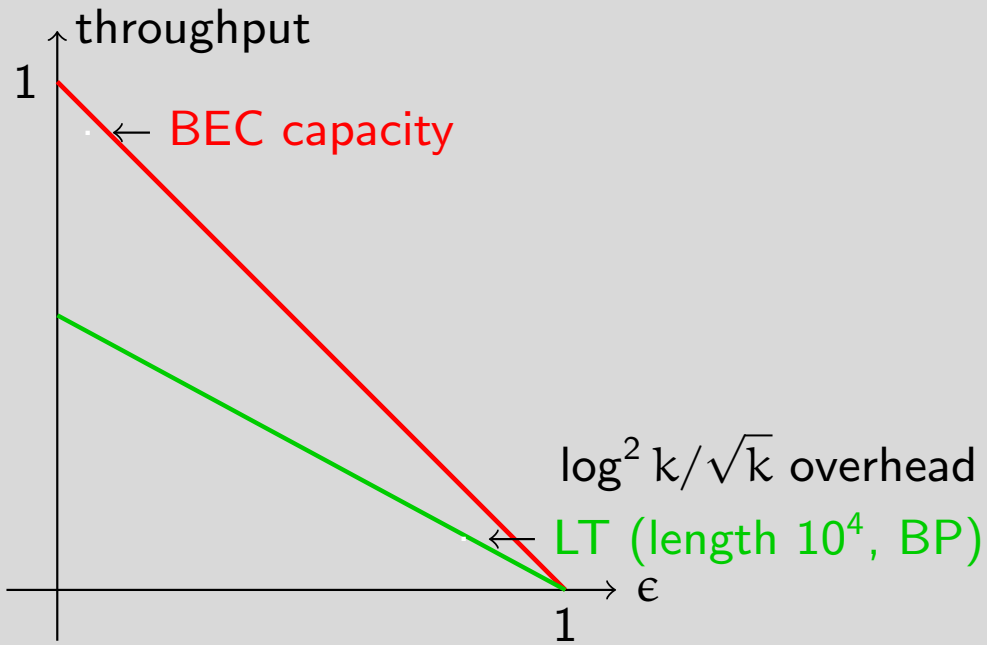
What About Coding Overhead?

When $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$ by using the following, Ideal Soliton, degree distribution:

$$p_d = \begin{cases} 1/k, & d = 1, \\ 1/[d(d-1)], & d = 2, 3, \ldots, k, \end{cases}$$

the variance in the decoding process will cause BP decoder to fail.

More robust distributions can guarantee a BP decoding failure rate of at most δ, when $k(1 + \log^2(k/\delta/\sqrt{k})$ or more y's are received.



The Raptor (Partial) Remedy

Soliton based LT code with BP decoding failure probability δ has

1. decoding complexity $\mathcal{O}(k \log(k/\delta))$
2. average overhead $\log^2(k/\delta)/\sqrt{k}$

Q: Can we recover all data in linear time from $\mathcal{O}(k)$ code symbols?
A: No, under our probabilistic model, but we can recover a large fraction.

Raptor codes are concatenated codes where

- data $\{x_1, \ldots, x_k\}$ are first pre-coded into $\{z_1, \ldots, z_m\}$, $m > k$, then
- precode symbols $\{z_1, \ldots, z_m\}$ are mapped into code symbols $\{y_i\}_1^\infty$

We don't have to recover all z's from $\mathcal{O}(k)$ received y's BUT just enough of them to allow us to decode all $\{x_1, \ldots, x_k\}$.

The Coupon Collection Problem

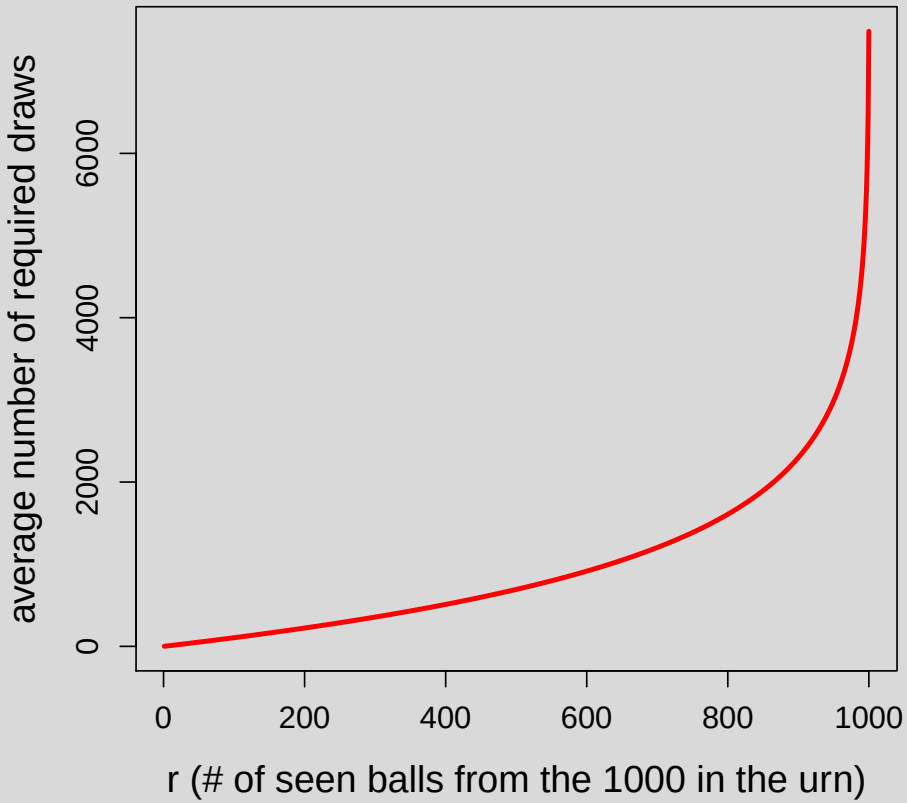
An urn contains k balls that are numbered from 1 to k.
Balls are drawn one at the time with replacement.

The average number of draws reacquired to see r different balls:

$$\sum_{\ell=0}^{r-1} \frac{k}{k-\ell} = k(H_k - H_{k-r})$$

$$\sim \boxed{k \log \frac{k}{k-r}}$$

for large k and $k-r$.



Further Raptor Remedy?

Seeking to recover only a constant fraction of z's from $\mathcal{O}(k)$ received y's

- enables BP decoding in $\mathcal{O}(k)$ steps $\leftarrow$ linear time
  by admitting degree distribution with small constant average degree
- still requires non-negligible overhead for BP decoding

How about a hybrid BP&GE (Gaussian Elimination) decoding?

An Example:

Suppose that after decoding, we are left with the following code symbols:

$$y_{\ell_1} = x_1 + x_2$$

$$y_{\ell_2} = x_1 + x_3$$

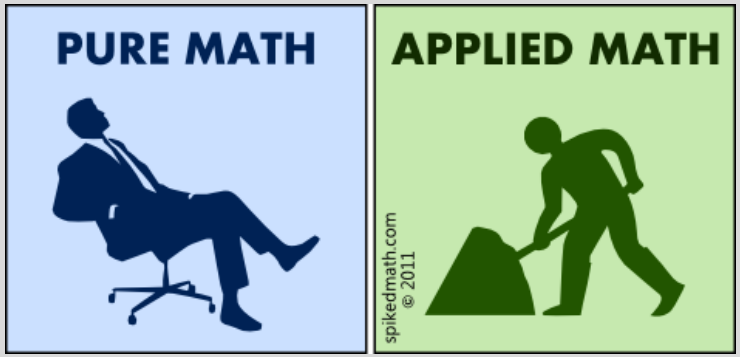
$$y_{\ell_3} = x_1 + x_3 + x_4$$

$$y_{\ell_4} = x_1 + x_2 + x_3 + x_4$$

The Inactivation Decoder

How about a hybrid BP&GE decoding?

Back to the drawing board:



$\Rightarrow$ inactivation decoder – integral part of R10 and beyond

Further Raptor Evolution – Systematic Codes

Systematic codes map data symbols $\{x_1, \ldots, x_k\}$ into code symbols $\{y_i\}_1^\infty$ s.t. $y_i = x_i$ for $i = 1, \ldots, k$, i.e., $\{y_i\}_1^\infty = \{\underbrace{x_1, \ldots, x_k}_{\text{systematic part}}, \underbrace{y_{k+1}, y_{k+2}, \ldots}_{\text{parity symbols}}\}$.

Systematic (data) part is useful

- since no decoding is required when the original data is received
- in broadcast, when some receivers have decoders and some do not
- for non-coding purposes, such as synchronization and monitoring

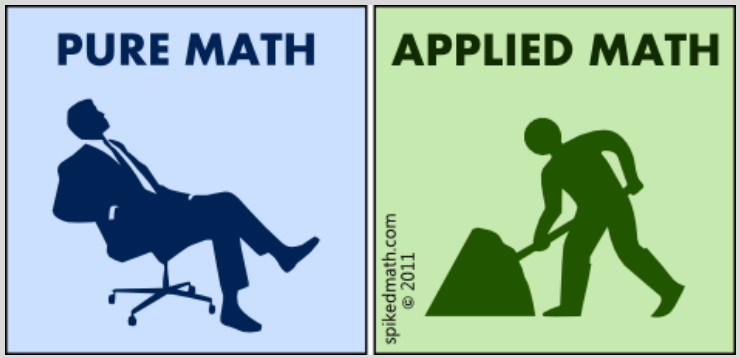
A fixed-rate, non-systematic, linear code can be easily transformed into a systematic code with essentially identical erasure recovery capabilities.

Systematic Raptor Codes

How can we construct a systematic rateless code?

What about a code where data is followed by Raptor-generated symbols?
That systematic code does not have Raptor's overhead/erasure-recovery.

Back to the drawing board:



$\Rightarrow$ systematic Raptor code

$$\{y_i\}_1^\infty = \{\underbrace{x_1, \ldots, x_k}_{\text{data}}, \underbrace{y_{k+1}, y_{k+2}, \ldots}_{\text{parity symbols}}\}$$

(cf. delivery vs. repair symbols).

Standardized Raptor Codes: R10 and RQ

From the *"Raptor Codes"* NOW tutorial by Shokrollahi and Luby:

> *... their provable properties are even more real than a theoretical proof: highly optimized software implementing the R10 and the RQ codes has been developed, tested, and deployed in mission critical applications.*

Raptor10 (R10)

- designed for applications with relatively modest requirements
  e.g., mobile broadcast applications
- works on source blocks of up to 8192 symbols, $\leftarrow$ medium size
  and supports up to 65536 encoded symbols.
- failure probability of 10^{-6} is achieved with an overhead of a few symbols, for all supported source block lengths. $\leftarrow$ reasonable

RaptorQ (RQ)

- designed for a much larger range of applications with more stringent requirements, e.g., high-end streaming, large mobile data delivery.
- fast encoding and decoding and exceptional overhead-failure curves are crucial, and support for large source blocks is mandatory.
- works on source blocks of up to 56403 symbols, $\leftarrow$ large size and supports up to 16777216 encoded symbols.
- overhead vs. failure tradeoff essentially mimics that of a random fountain code over the field $\mathbb{F}_{256}$

Random Linear Codes

- Data symbols $\{x_1, \ldots, x_k\}$ are mapped into code symbols $\{y_i\}_1^\infty$.
- y_i are linear combinations of $\{x_1, \ldots, x_k\}$, $x_j, y_i \in \mathbb{F}_q$:

$$y_i = \alpha_1^i x_1 + \cdots + \alpha_k^i x_k, \quad \alpha_j^i \in \mathbb{F}_q$$

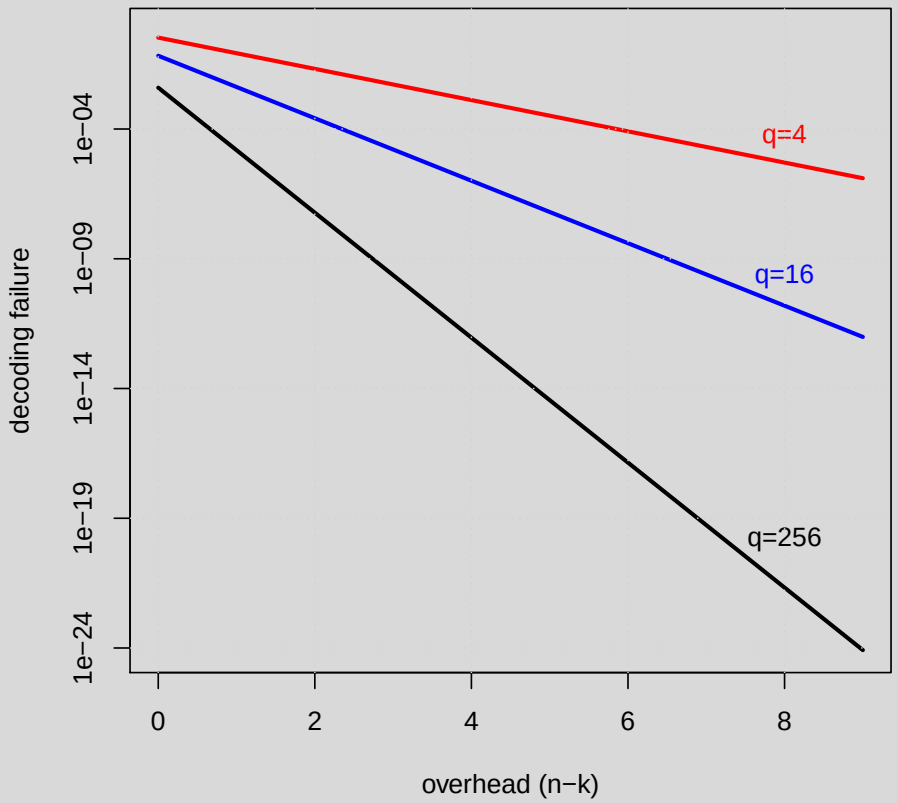
where α_i are chosen uniformly at random from $\mathbb{F}_q$.

Are these network codes or fountain codes or ... ?

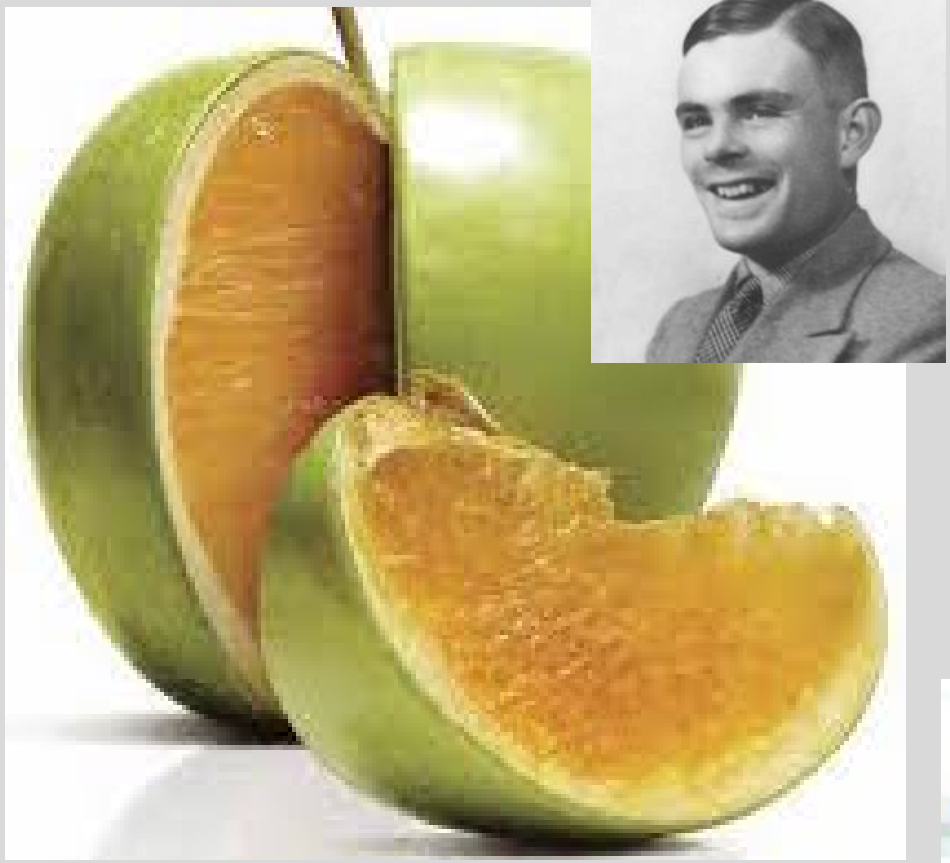
Overhead vs. Failure Tradeoff for Random Linear Codes

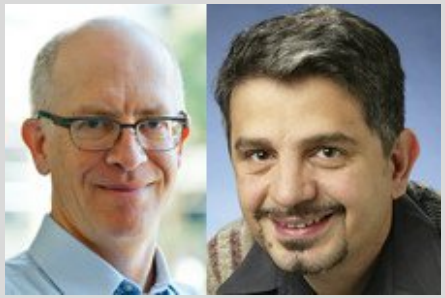
The probability of having k linearly independent y's among n received:

$$\prod_{s=0}^{k-1}(1-q^{s-n}) \geqslant \begin{cases} 0.288, & \text{if } q=2 \text{ and } k=n; \\ 1-\dfrac{1}{q^{n-k}(q-1)}, & \text{otherwise.} \end{cases}$$



What are Raptor Codes?





for the conception, development, and analysis of **practical** rateless codes

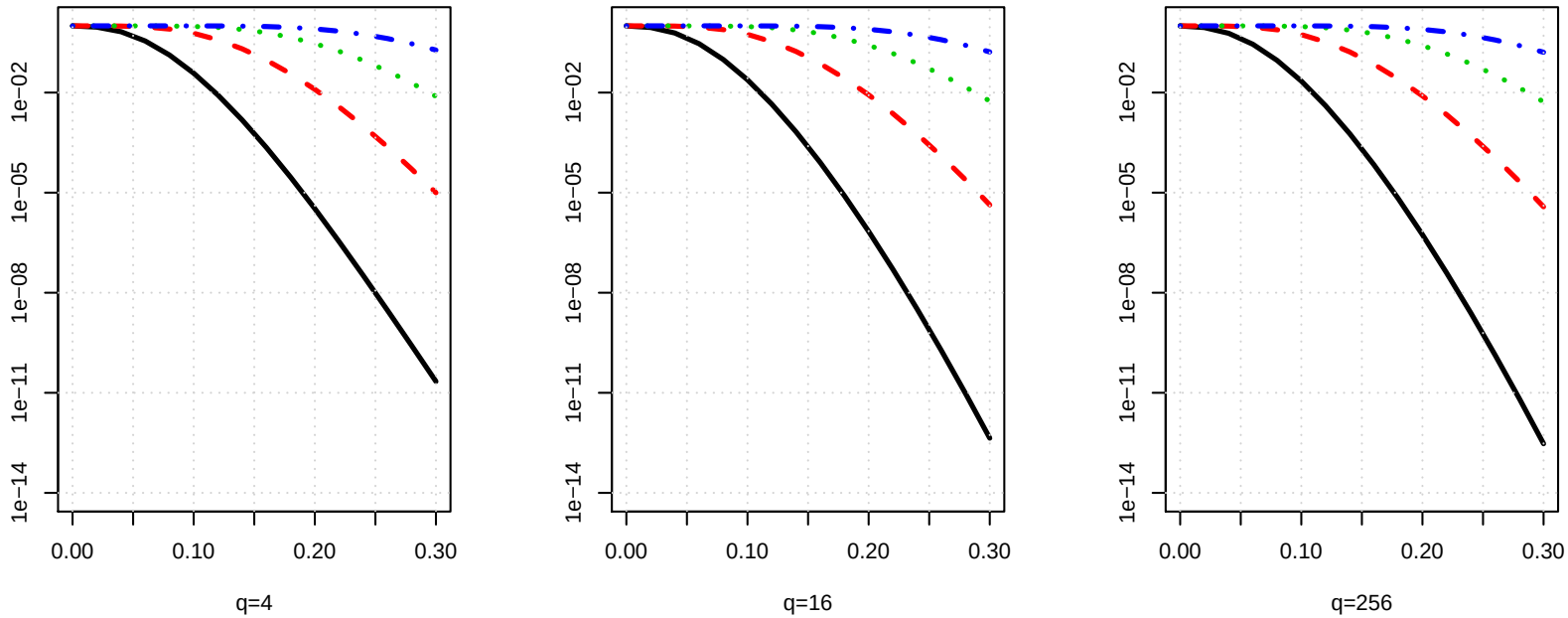
Transmission Failure Rate – Required Coding and Transmission Overhead

Suppose m code symbols are transmitted over a channel with loss rate ε. Then the probability of successful decoding is

$$\sum_{j=k}^{m} \binom{m}{j} (1-\varepsilon)^j \varepsilon^{m-j} \prod_{s=0}^{k-1} (1-q^{s-j}) \gtrsim \sum_{j=k}^{m} \binom{m}{j} (1-\varepsilon)^j \varepsilon^{m-j} \left(1 - \frac{1}{q^{j-k}(q-1)}\right).$$

Transmission Failure Rate vs. Transmission Redundancy

for $k = 100$ and $\varepsilon = 5, 10, 15, 20\%$



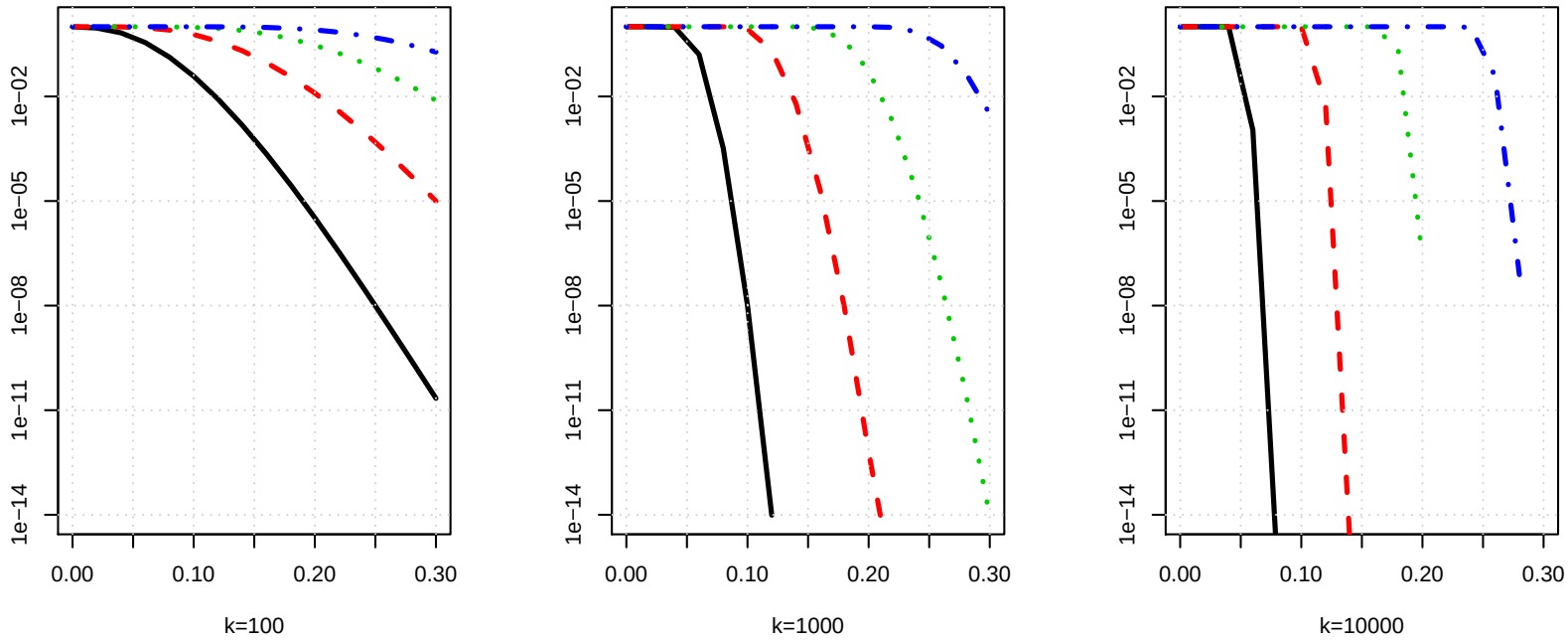
Transmission Failure Rate – Required Coding and Transmission Overhead

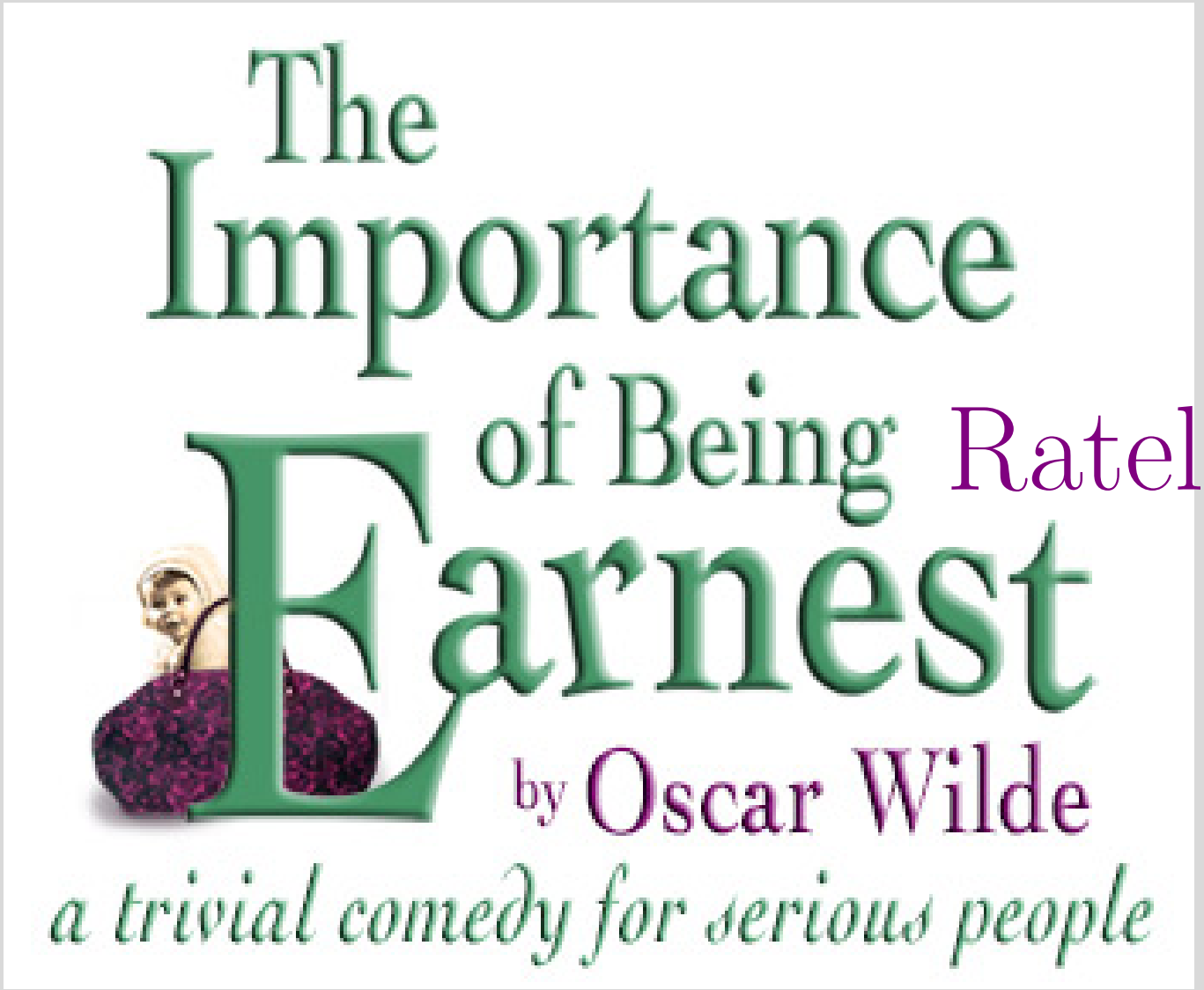
Suppose m code symbols are transmitted over a channel with loss rate ε. Then the probability of successful decoding is

$$\sum_{j=k}^{m} \binom{m}{j} (1-\varepsilon)^j \varepsilon^{m-j} \prod_{s=0}^{k-1} (1-q^{s-j}) \gtrsim \sum_{j=k}^{m} \binom{m}{j} (1-\varepsilon)^j \varepsilon^{m-j} \left(1 - \frac{1}{q^{j-k}(q-1)}\right).$$

Transmission Failure Rate vs. Transmission Redundancy

for $q = 4$ and $\varepsilon = 5, 10, 15, 20\%$




The
Importance
of Being
Earnest
by Oscar Wilde
a trivial comedy for serious people

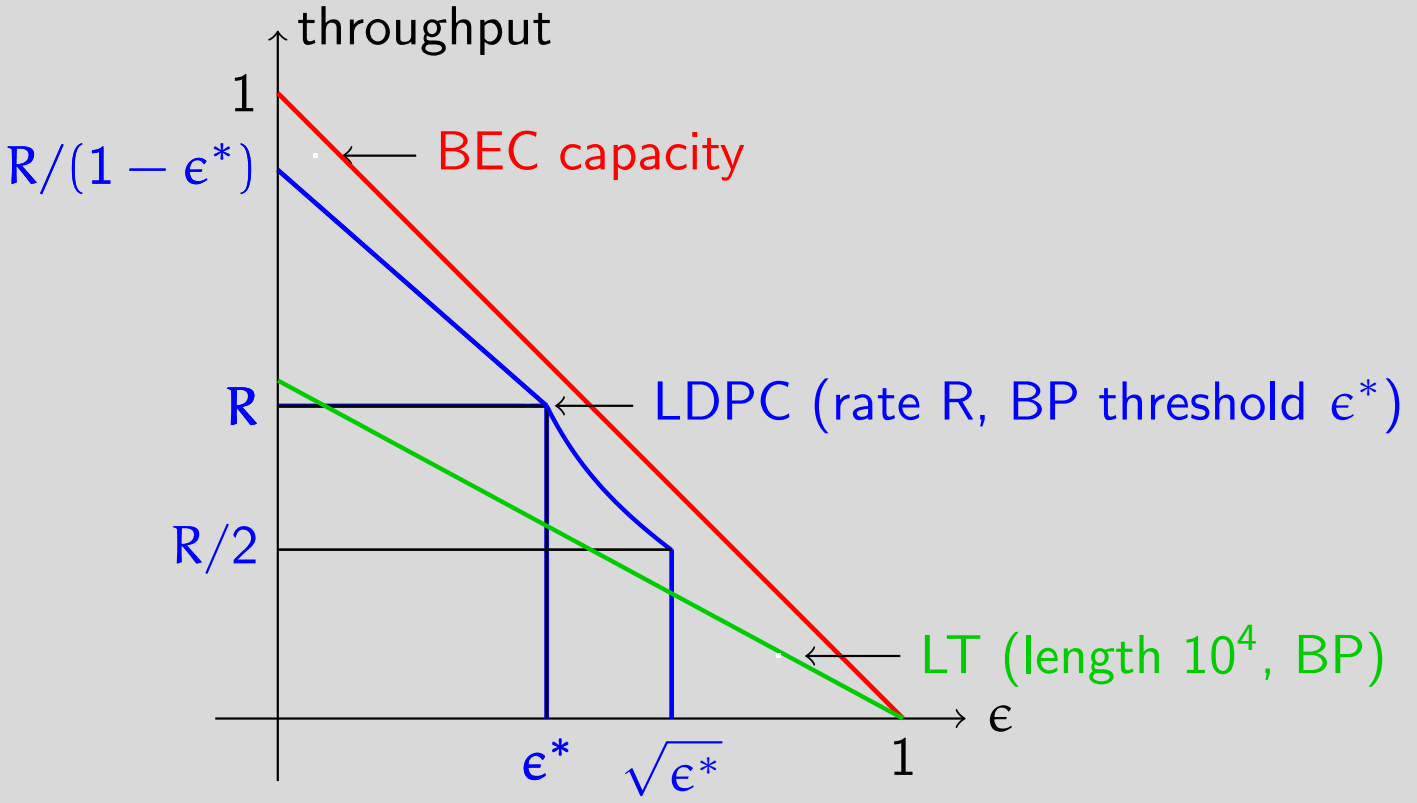
Rateless

What is Ratelessness and is it Overrated?

Ratelessness matters and means different things to different people:

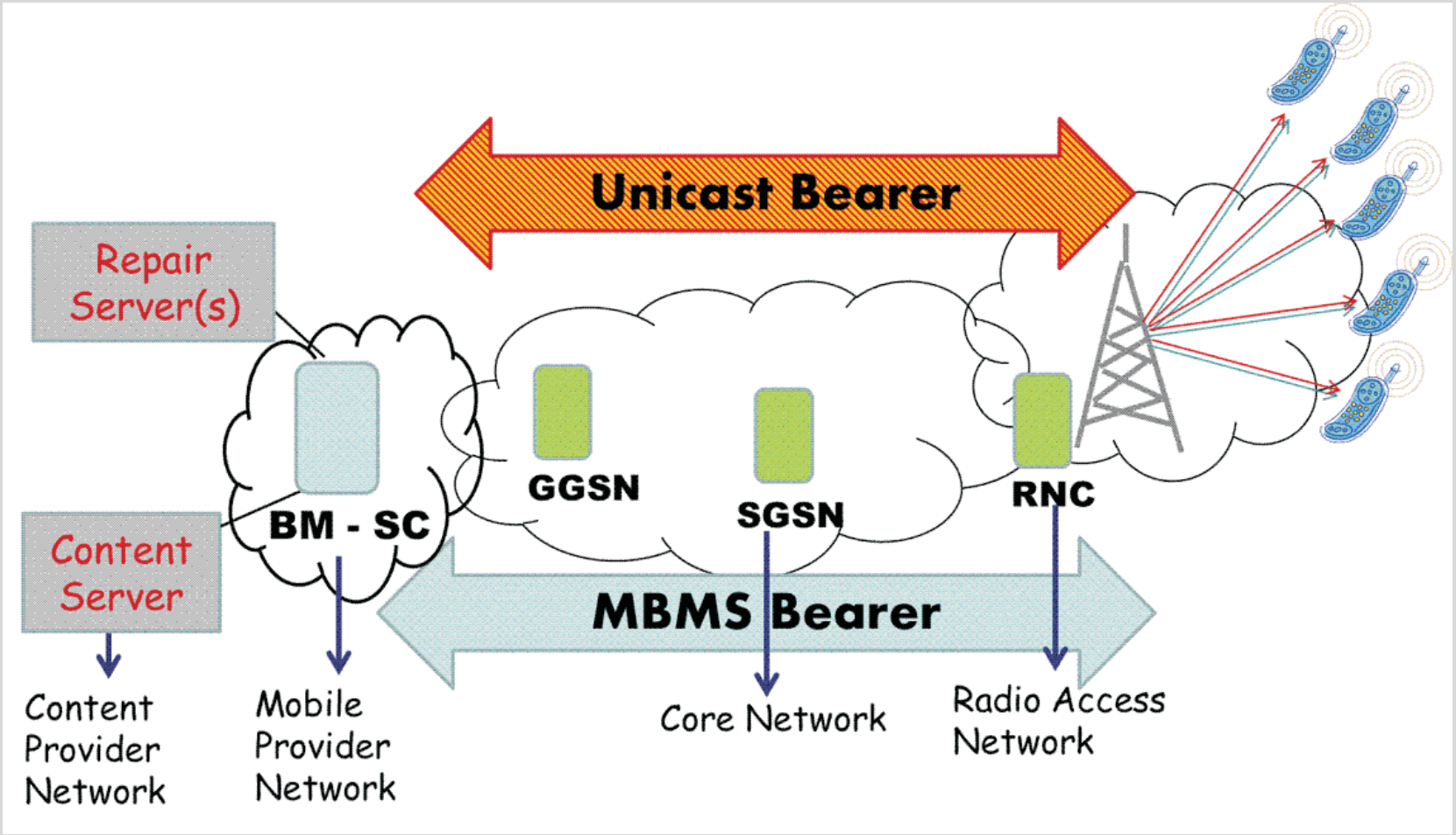
- encoder can produce potentially infinite stream of symbols
- each code symbol is statistically identical

Can fixed rate codes be made rateless?



The Unbearable Ratelessness of Coding

Two eMBMS Phases: Multicast Delivery & Unicast Repair:



... and other unBEARables

Have a great hike!