The Evolution of Erasure Codes for Large Scale Data Storage and Multimedia Broadcast

Suayb S. Arslan
Quantum Corporation,
University of California, San Diego

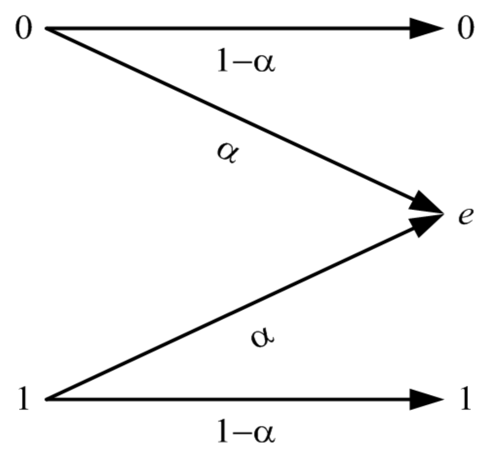
Faculty of Computer and Informatics
Istanbul Technical University
09/09/2013

Contents

- Fundamentals
- Efficient Erasure Codes
  - Rateless Codes, Raptor codes, Online codes
- Use of erasure codes for Disk array storage
- Reliability of large scale storage using optimal/sub-optimal erasure codes
- Optimization of rateless codes for progressive source transmission
- Conclusions

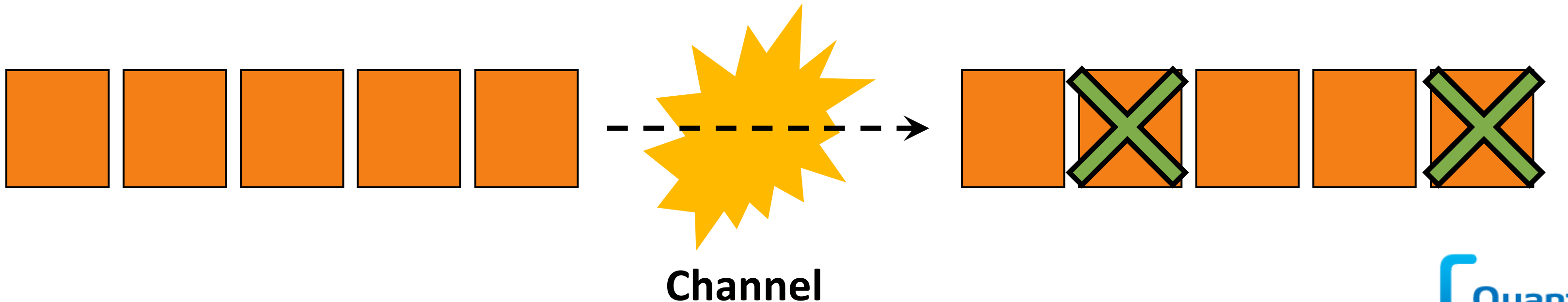
Channel Model

- Binary Erasure Channel



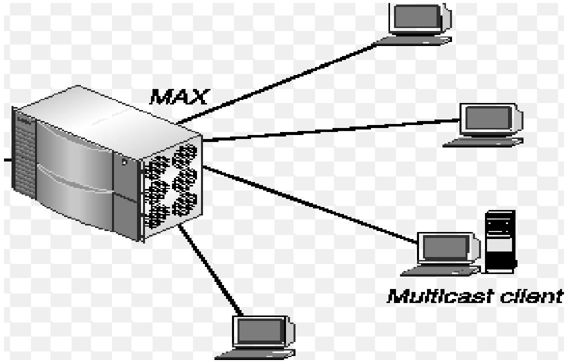
- *e : Erasure*
- *α : erasure probability*
- *Capacity: 1- α*
- *An average of α fraction of bits are lost in the channel, we can at most recover a proportion (1- α) of the bits.*

- Message bits are typically packetized.
- Packets may be corrupted or lost during transmission.



Possibilities for lost data recovery

- If there is a feedback channel, request the lost packets to be retransmitted.
- This may result in round-trip delays and lots of feedback channel use.
- For broadcast, number of request might be overwhelming!



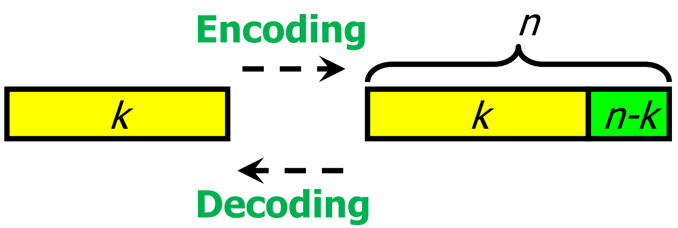
- Forward Error Correction (Erasure coding to restore data).

Coding theory basics

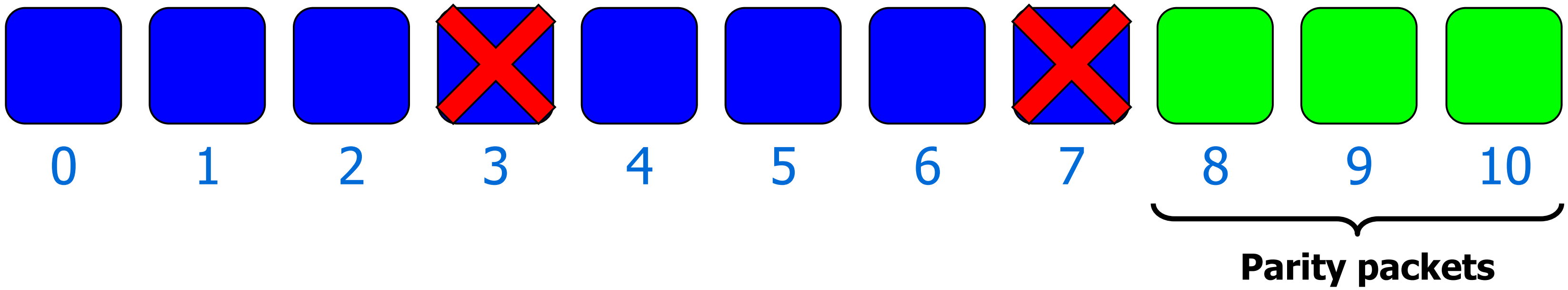
- A code C over a finite alphabet $\sum$ of length n is a subset of $\sum^n$
  - The elements of C are called the codewords in C.
  - If $|\sum| = q$, C is called q-ary code.
- A binary code $(q = 2)$ is a code over the alphabet $\{0, 1\}$.
  - $C_1 = \{000, 010, 101, 100\}$
  - $C_2 = \{00000, 01101, 10111, 11011\}$
- The maping between codewords and message sequences is called “encoding”. The reverse of this operation is called “decoding”.
- Hamming distance:
  - $h(x, y) =$ the number of symbols *x* and *y* differ. $h(10101, 01100) = 3$,
- Minimal distance of a code C:
  - $d_{min}(C) = \min\{h(x, y) | x, y \in C, x \neq y\}$,
- **Theorem 1:** A code with minimal distance d_{min} can correct $d_{min} - 1$ erasures.

Classical Erasure codes

- Encode an original data of k packets to n code packets. Such a code is referred as (n, k) block code.
- **Theorem 2:** A (n, k) code with minimal distance d_{min} satisfies $d_{min} \leq n - k + 1$. (Singleton bound)
- The decoder needs $\tilde{n} \geq k$ code packets to reconstruct the original data.
- If $\tilde{n} = k$, the code is called maximum distance separable (MDS)
- Redundant packets: $n - k$.
- Rate of the code: $r = k/n$
- Overhead: $c = n/k$



Example: MDS block code (11,8)



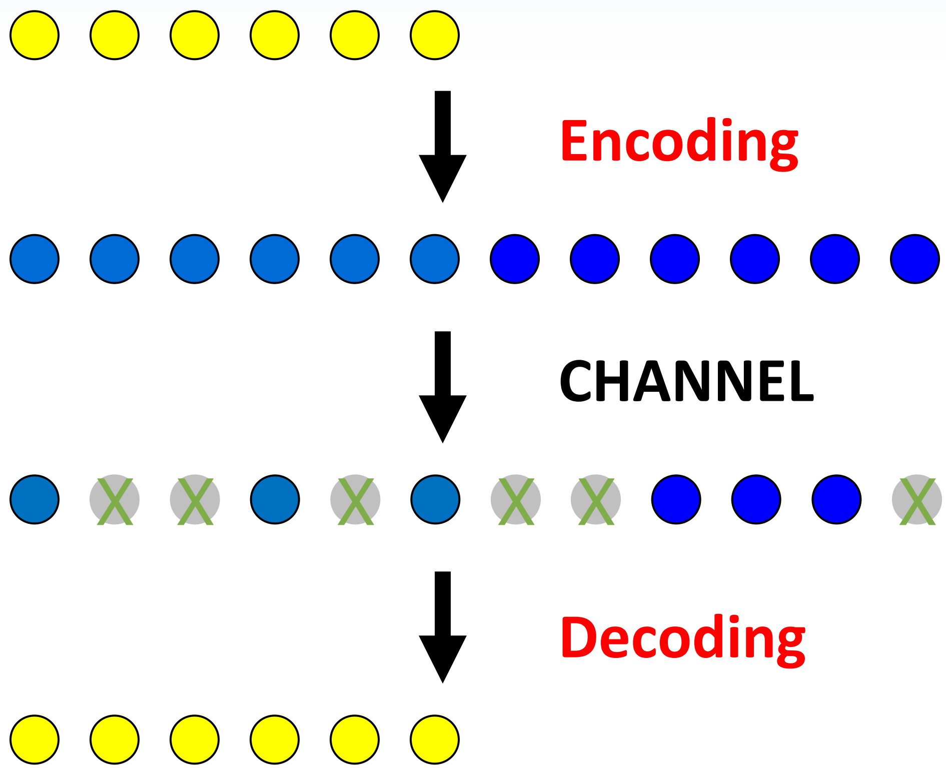
Classical Erasure codes

k message

$n = ck$ encoding

$\geq k$ collected

k message



Trivial Binary MDS Erasure codes

- Repetition coding: *(n, k=1)*

I	I	I	I	I	I	I	I

- Parity coding: *(n, k=n-1)*

I_1	I_2	I_3	I_4	I_5	I_6	I_7	P

$$P=\sum_{i=1}^{7} I_i \pmod 2$$

- There is no non-trivial binary MDS code.

Non-Trivial MDS Erasure codes

- One of the well known **non-trivial, non-binary** MDS code is Reed-Solomon (RS) codes. RS codes are defined over Galois Fields such as $GF(2^m)$.
- Construction is based on a polynomial evaluation.

$$\mathbf{m} = (m_0, m_1, \ldots, m_{k-1})^T$$

$$m(x) = m_0 + m_1 x + m_2 + \cdots + m_{k-1} x^{k-1}$$

- Evaluate $m(x)$ at n specific points to form the codeword:

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$$

- Encoding Complexity ~ $O(ck^2)$
- Decoding Complexity ~ $O(n \log^2(n) \log(\log(n)))$

Not very easy

- Complexity is also a strong function of the size of the Galois Field over which the code is defined. If RS code is defined over $GF(2^m)$ and $n = 2^m - 1$, decoding complexity can be approximated by

$$C = 2m^2(N_{multiplications} + N_{inversions}) + mN_{additions} \qquad [1]$$

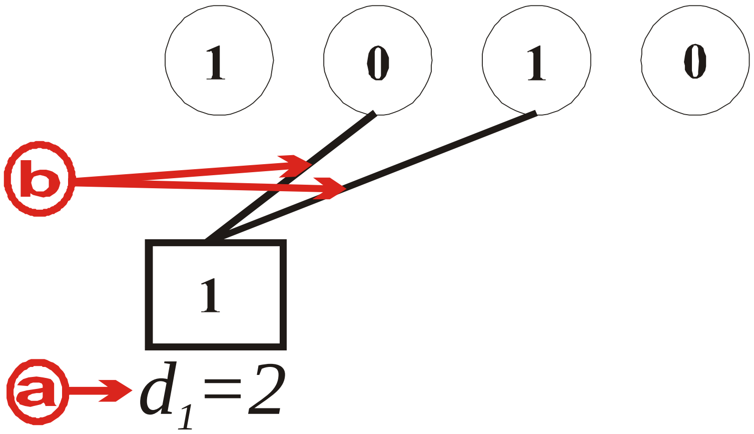
[1] N. Chen and Z. Yan, "Complexity analysis of reed-solomon decoding over GF(2m) without using syndromes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2008, Article ID 843634, 11 pages, 2008.

Issues with conventional erasure codes

- RS code are <u>fixed rate</u> code and thus, its rate must be fixed before transmission.
- In a broadcast scenario, the erasure rate of the channel is not known prior to transmission.
- RS codes are complex to implement, particularly for large block lengths n and rate r.
- If an erased symbol is to be reconstructed, all data must be read. In a storage scenario, if each data packet is stored in different nodes, all nodes must be accessed → increased bandwidth usage.
- We need a different paradigm for constructing erasure codes, possibly with MDS property!

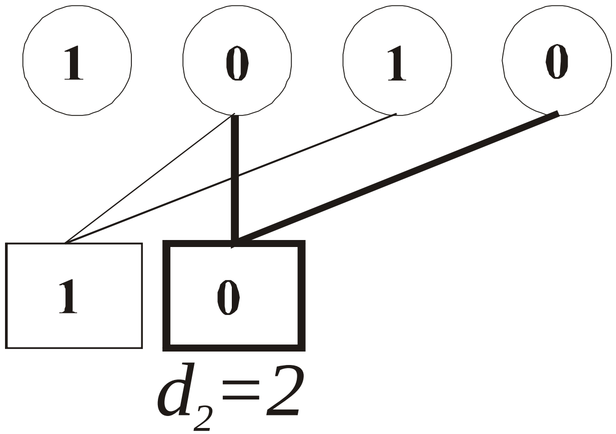
Rateless erasure codes

- Rate-less codes, i.e. there is no predetermined overhead $c=n/k$. One can generate as much code symbols as desired. An instantiation of such a construction is given by Luby in 2001, called Luby Transform (LT) codes.
- Asymptotically optimal: Only $n = (1+\epsilon)k$ coded symbols are enough to recover all k information symbols.
- **Simple Encoding:** encoded symbols are XORs of data symbols.
  - Pick a degree d from the appropriate degree distribution $\Omega(x)$ **(a)**
  - Randomly pick d data symbols. **(b)**
  - Encode them as encoded symbol by using XOR operations.



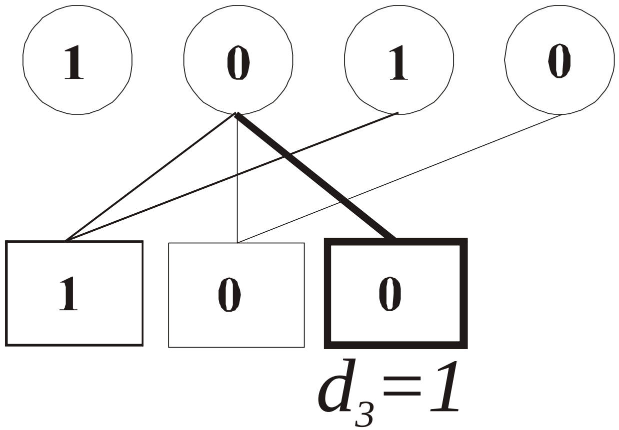
Rateless erasure codes

- Rate-less codes, i.e. there is no predetermined overhead $c=n/k$. One can generate as much code symbols as desired. An instantiation of such a construction is given by Luby in 2001, called LT codes.
- Asymptotically optimal: Only $n = (1+ \epsilon)k$ coded symbols are enough to recover all k information symbols.
- **Simple Encoding:** encoded symbols are XORs of data symbols.
  - Pick a degree d from the appropriate degree distribution $\Omega(x)$
  - Randomly pick d data symbols
  - Encode them as encoded symbol by using XOR operations.



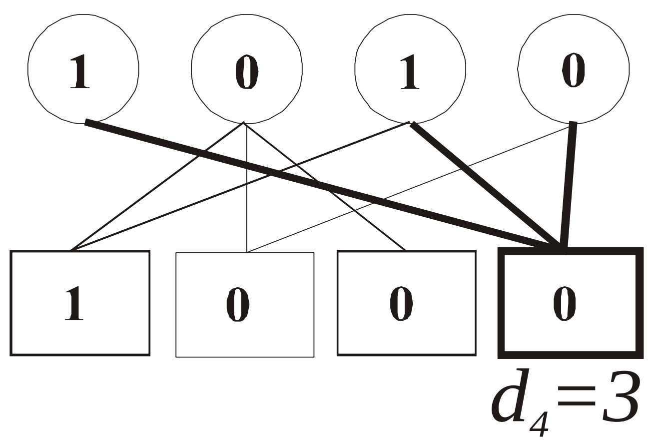
Rateless erasure codes

- Rate-less codes, i.e. there is no predetermined overhead $c=n/k$. One can generate as much code symbols as desired. An instantiation of such a construction is given by Luby in 2001, called LT codes.
- Asymptotically optimal: Only $n = (1+\epsilon)k$ coded symbols are enough to recover all k information symbols.
- **Simple Encoding:** encoded symbols are XORs of data symbols.
  - Pick a degree d from the appropriate degree distribution $\Omega(x)$
  - Randomly pick d data symbols
  - Encode them as encoded symbol by using XOR operations.



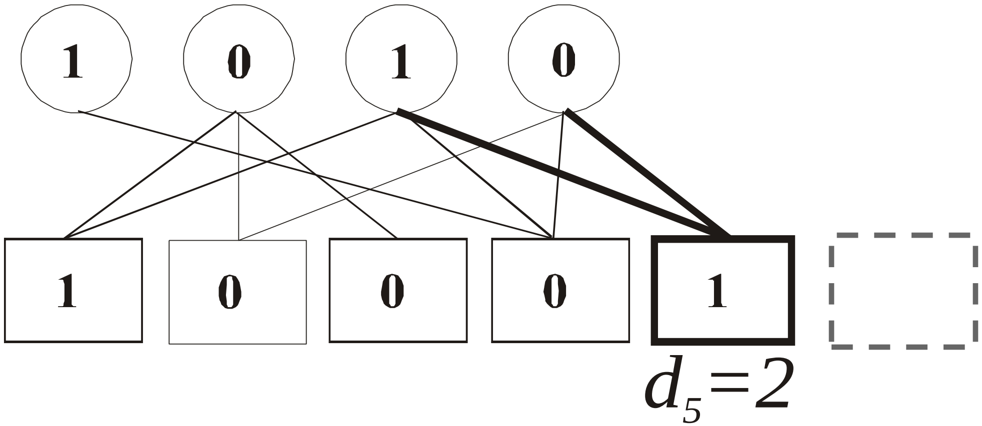
Rateless erasure codes

- Rate-less codes, i.e. there is no predetermined overhead $c=n/k$. One can generate as much code symbols as desired. An instantiation of such a construction is given by Luby in 2001, called LT codes.
- Asymptotically optimal: Only $n = (1+\epsilon)k$ coded symbols are enough to recover all k information symbols.
- **Simple Encoding:** encoded symbols are XORs of data symbols.
  - Pick a degree d from the appropriate degree distribution $\Omega(x)$
  - Randomly pick d data symbols
  - Encode them as encoded symbol by using XOR operations.



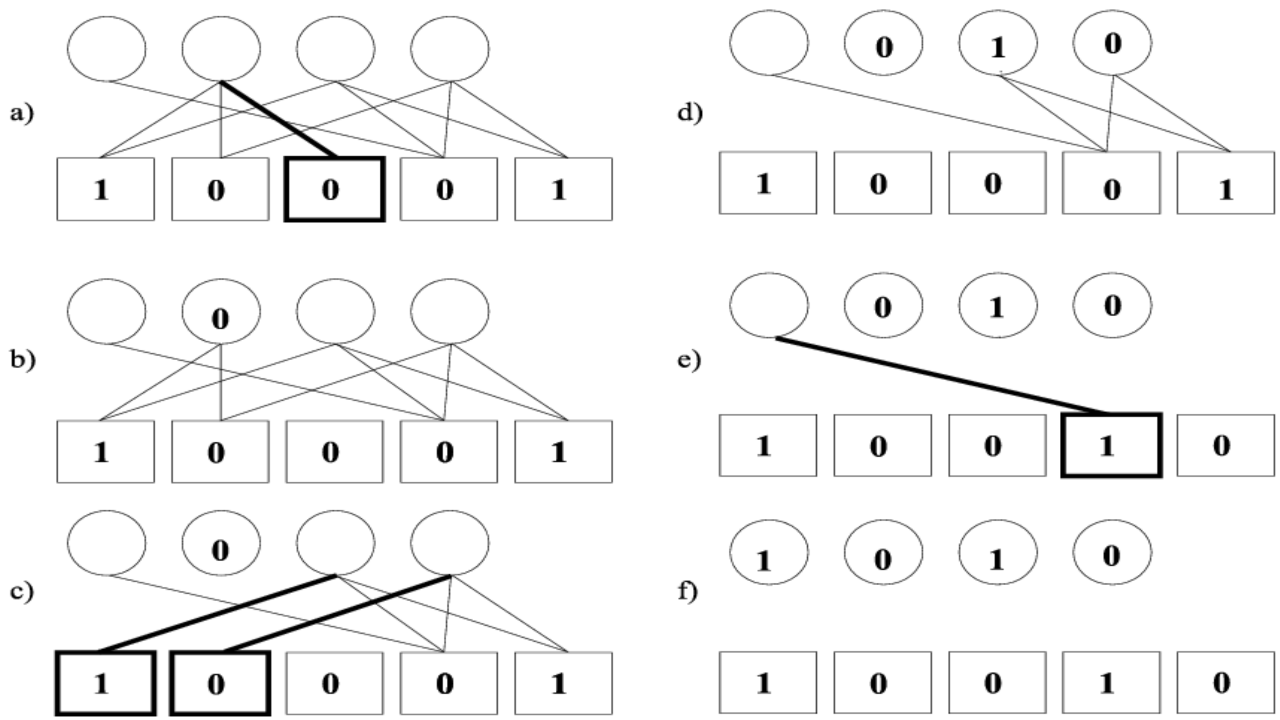
Rateless erasure codes

- Rate-less codes, i.e. there is no predetermined overhead $c=n/k$. One can generate as much code symbols as desired. An instantiation of such a construction is given by Luby in 2001, called LT codes.
- Asymptotically optimal: Only $n = (1+\epsilon)k$ coded symbols are enough to recover all k information symbols.
- **Simple Encoding:** encoded symbols are XORs of data symbols.
  - Pick a degree d from the appropriate degree distribution $\Omega(x)$
  - Randomly pick d data symbols
  - Encode them as encoded symbol by using XOR operations.



Simple Decoding

- Coded symbols are sent over a binary erasure channel.
- Decoder uses a Belief Propagation (BP) algorithm.



Degree distribution

Soliton Distribution

$$\underbrace{\qquad\qquad\qquad}$$

$$\rho(i) = \begin{cases} 1/k & i = 1 \\ 1/i(i-1) & i \in \{1..k\} \end{cases}$$

Robust Soliton Distribution (RSD)

$$\underbrace{\qquad\qquad\qquad}$$

$$R = c \cdot \ln(k/\delta)\sqrt{k}, \qquad c, \delta > 0$$

$$\tau(i) = \begin{cases} R/ik & i \in \{1 \dots k/R - 1\} \\ \dfrac{R \cdot \ln(R/\delta)}{k} & i = k/R \\ 0 & i \in \{k/R + 1..k\} \end{cases}$$

$$\beta = \sum_{i=1}^{k} \rho(i) + \tau(i)$$

$$\Rightarrow \mu(i) = \frac{\rho(i) + \tau(i)}{\beta}$$

- Degree distribution is chosen such that the decoding of the whole message block (k symbols) is ensured with high probability.

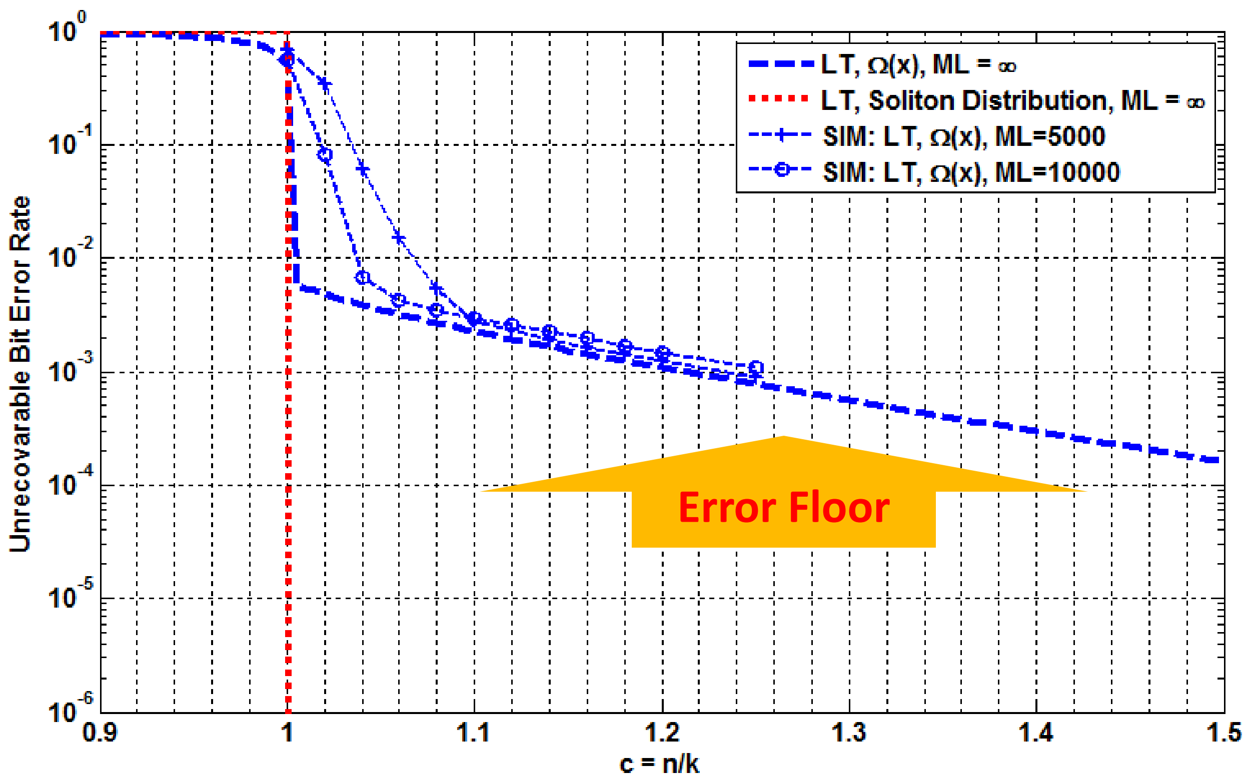
Efficiency & Complexity

- It is shown that using RSD, the probability that the decoding process will succeed after decoding $k + O(\sqrt{k}\ \ln^2(k/\gamma))$ is $1 - \gamma$. $\epsilon = O(\sqrt{k}\ \ln^2(k/\gamma))/k$
- The complexity of decoding is related to average number of XOR operations i.e., average number of edges in the graph $O\left(k \ln\left(\frac{k}{\gamma}\right)\right)$.
- **OBSERVATIONS:**
  - *Rateless construction.*
  - *Encoding/Decoding is <u>not linear</u> in k.*
  - *Asymptotically optimal (MDS) i.e., requires large n for vanishing ϵ.*
  - *We do not need to access all data symbols to resurrect a particular lost code symbol.*

How to achieve linear complexity?

- How about we have a distribution that the maximum degree is fixed, i.e., does not scale with increasing k.
- Let us assume we have the following degree distribution:

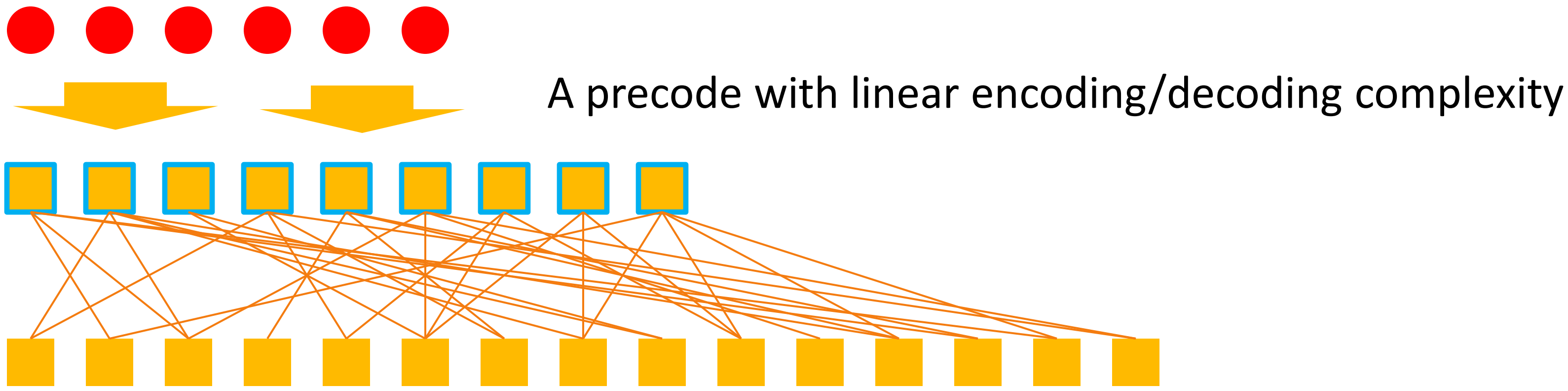
$$\Omega(x) = 0.007969x + 0.49357x^2 + 0.1662x^3 + 0.072646\,x^4 + 0.082558x^5 + 0.056058x^8$$
$$+0.037229x^9 + 0.05559x^{19} + 0.025023x^{65} + 0.003135x^{66}$$



[after erasures]

How to achieve linear complexity and no error floor?

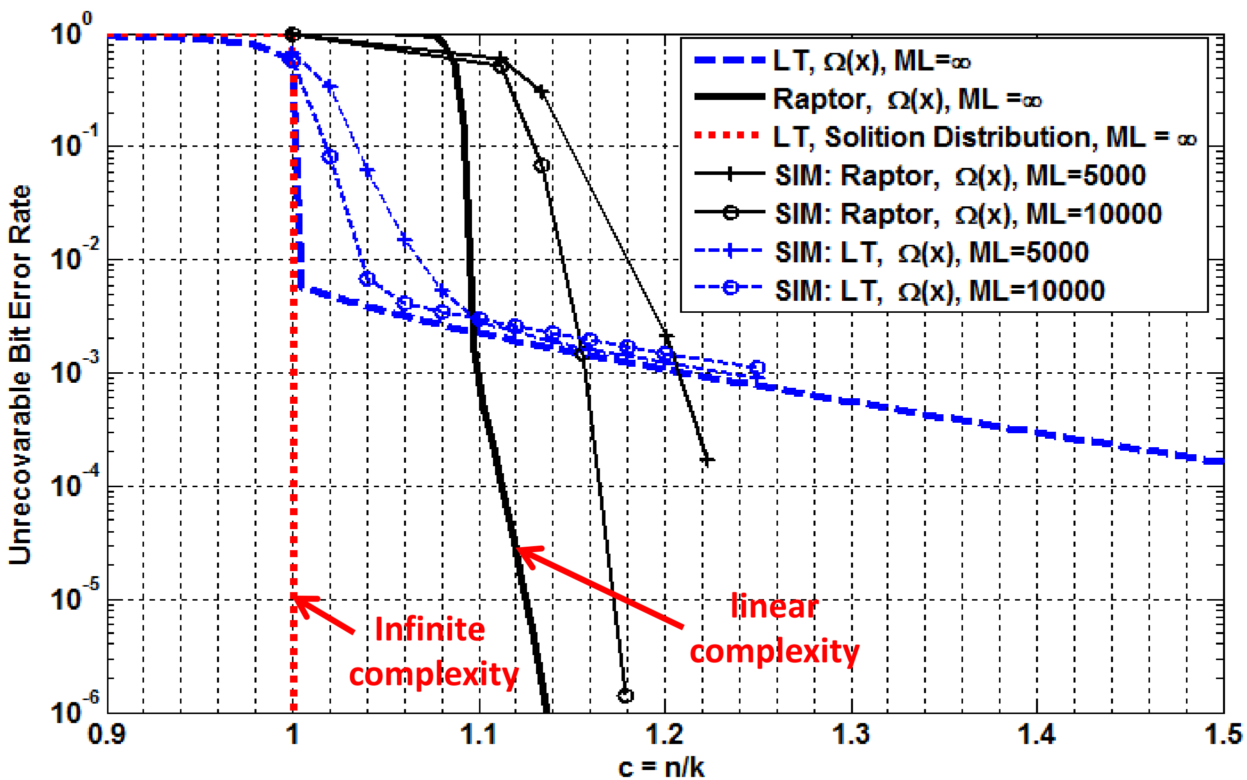
- The way to go is concatenation: Raptor Codes.



- Decoder for LT code decodes up to some fraction of the input symbols and then the rest of the erasures are corrected by the precode.
- The overall complexity is linear with k
- However, the overhead is increased due to the additional coding stage.

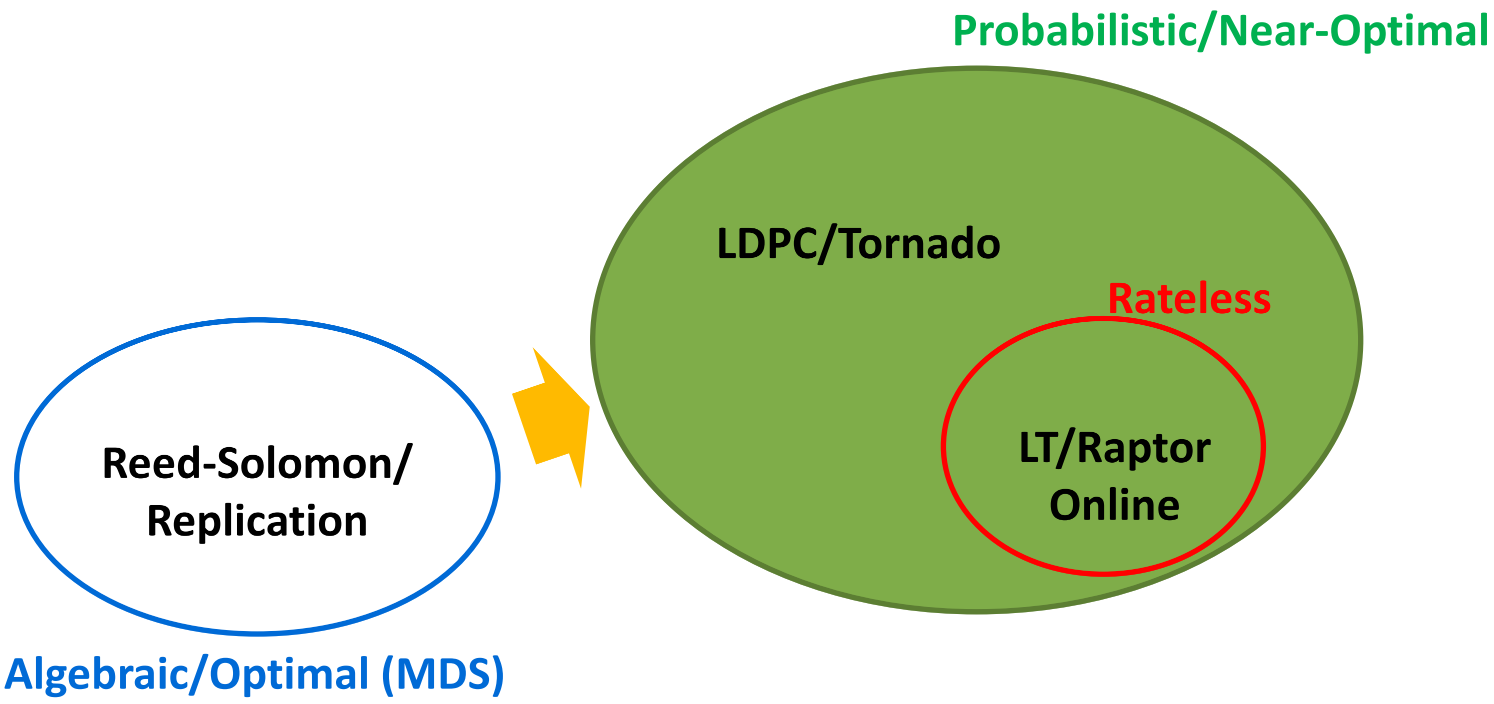
How to achieve linear complexity and no error floor?

- Raptor codes do not only achieve linear complexity, but also achieve low over head, near-optimal performance.
- They become to be part of 3rd Generation Partnership Project for use in mobile cellular wireless broadcast and also used by DVB-H standards for IP datacast to handheld devices



[after erasures]

Evolution of Erasure Codes



Quantum

ERASURE CODES FOR LARGE SCALE STORAGE

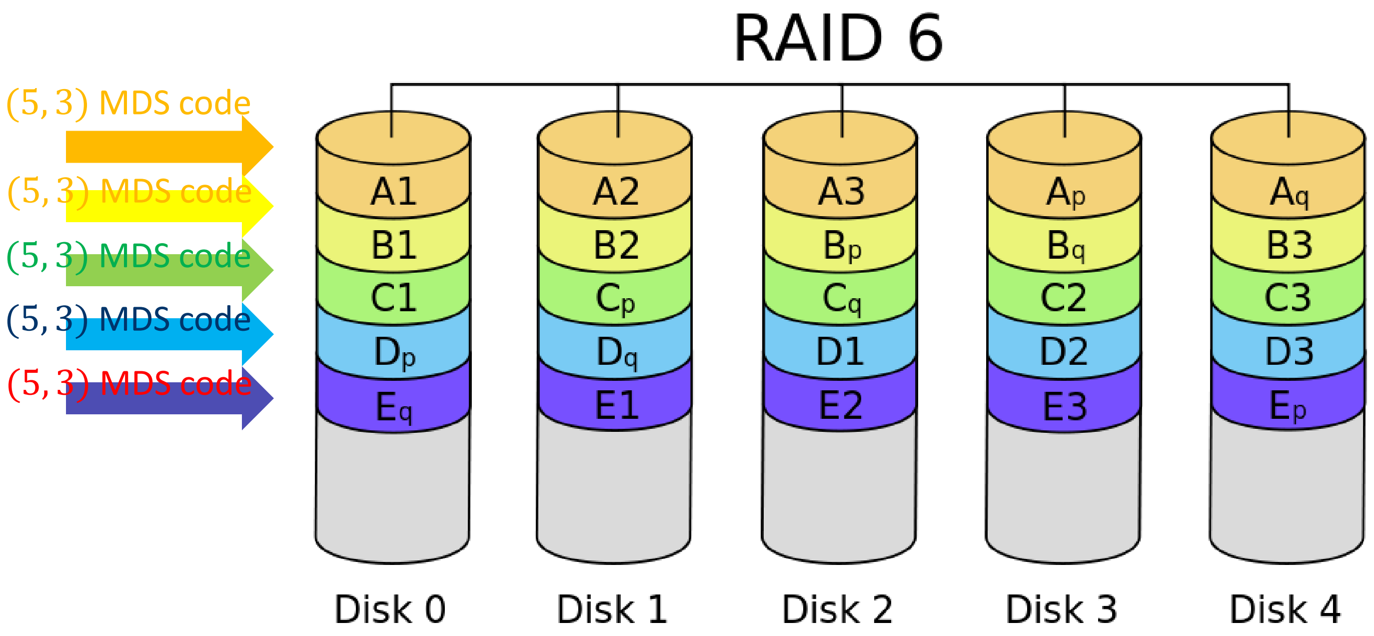
Suayb S. Arslan

suaybarslan@quantum.com

Quantum Corporation, 141 Innovation Dr., Irvine, CA, USA, 92612

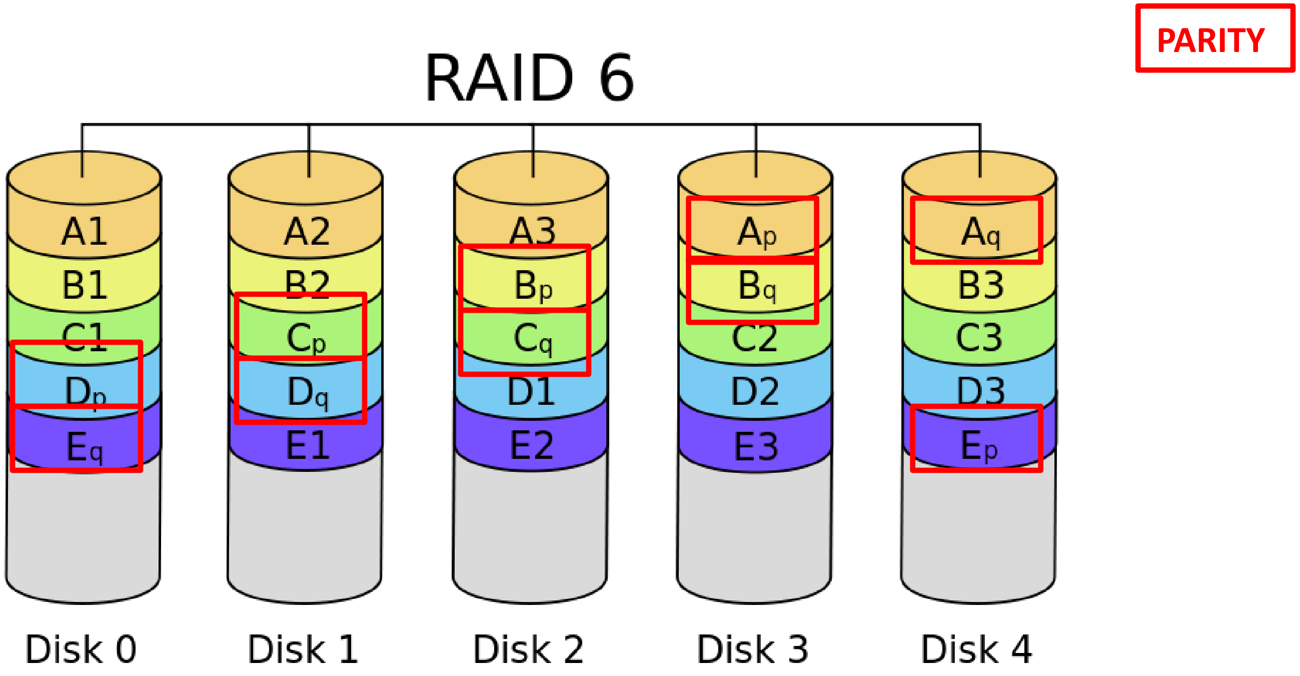
Evolution of erasure codes for data storage

- Classical MDS erasure codes are suboptimal for distributed storage networks because of the "*repair problem*"



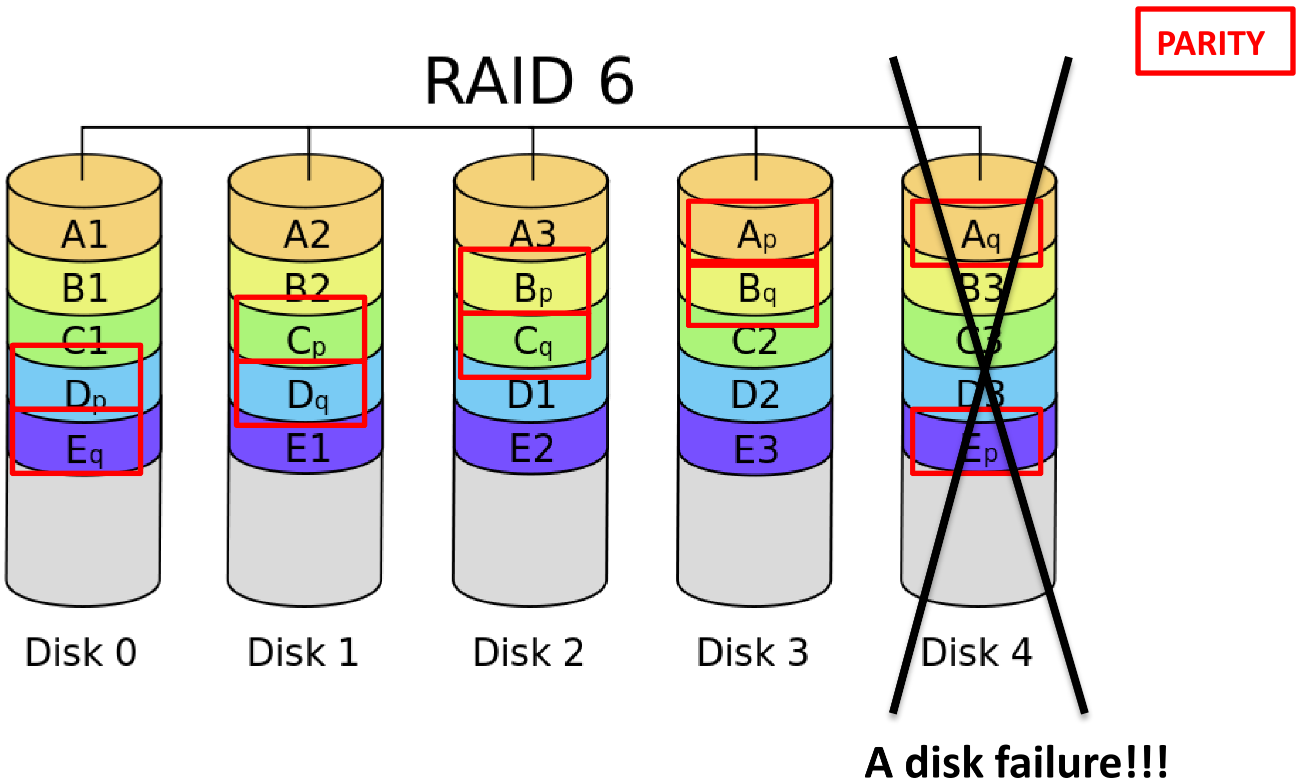
Evolution of erasure codes for data storage

- Classical MDS erasure codes are suboptimal for distributed storage networks because of the *“repair problem”*



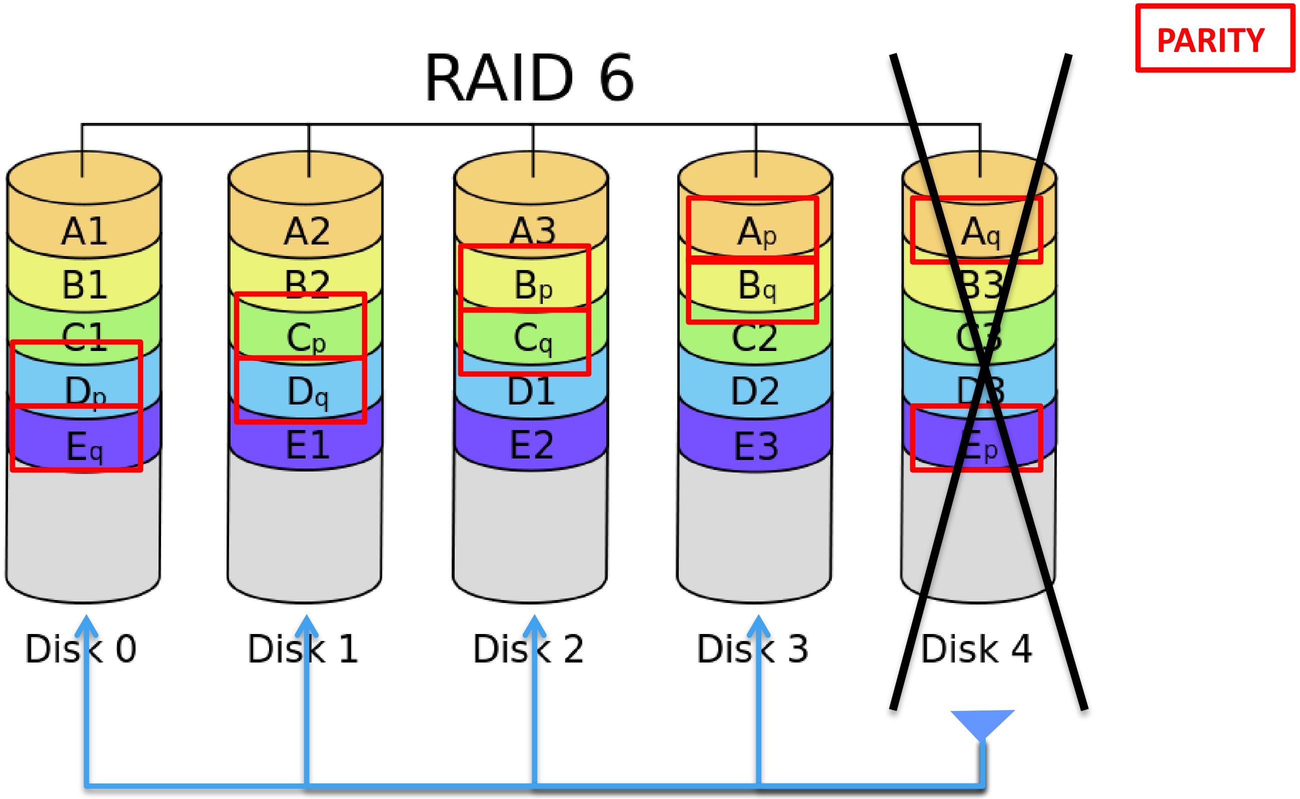
Evolution of erasure codes for data storage

- Classical MDS erasure codes are suboptimal for distributed storage networks because of the *"repair problem"*



Evolution of erasure codes for data storage

- Classical MDS erasure codes are suboptimal for distributed storage networks because of the *“repair problem”*



To repair the disk, we need to access all data in other disks: 4X more network repair overhead

Evolution of erasure codes for data storage

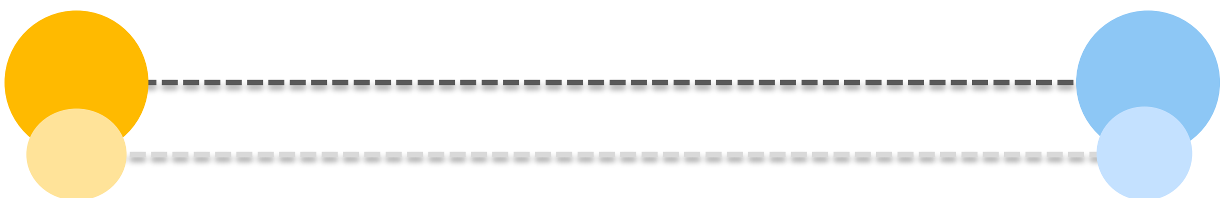
- Classical MDS erasure codes are suboptimal for distributed storage networks because of the *“repair problem”*.
- We need efficiently repairable erasure codes.

Least Efficient

Replication
(Repetition coding)

Most Efficient

Algebraic MDS codes

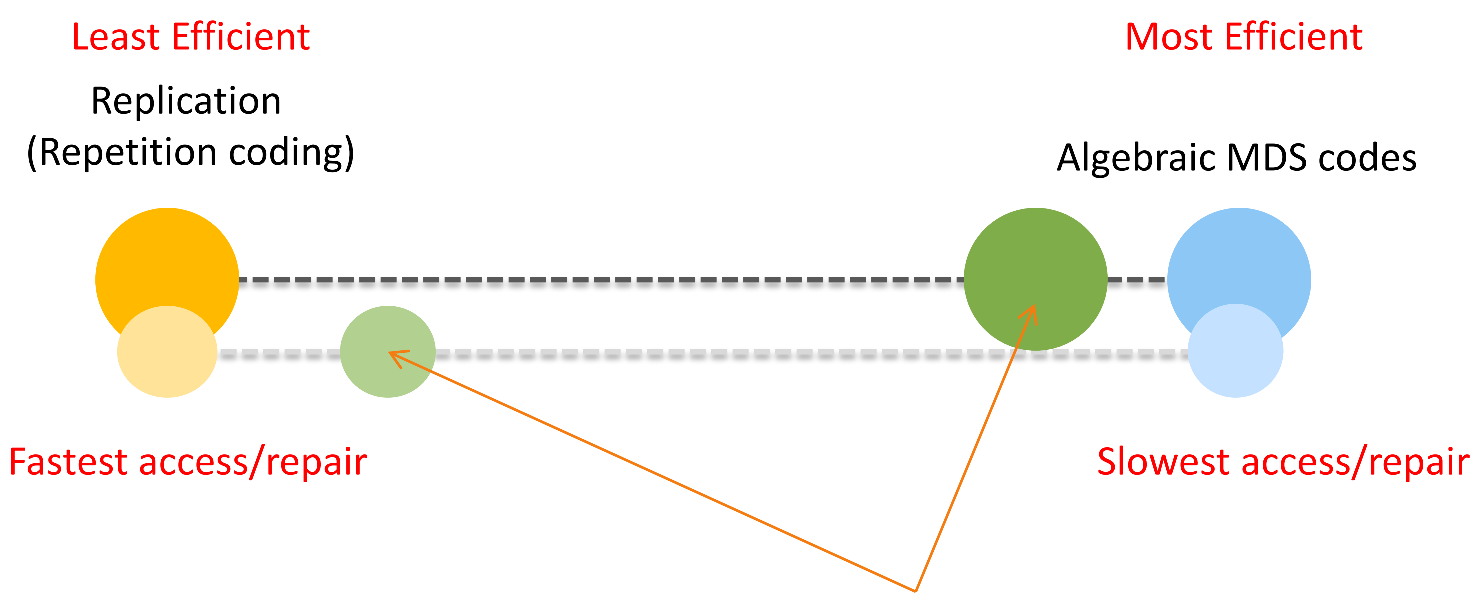


Fastest access/repair

Slowest access/repair

Evolution of erasure codes for data storage

- Classical MDS erasure codes are suboptimal for distributed storage networks because of the *"repair problem"*.
- We need efficiently repairable erasure codes.

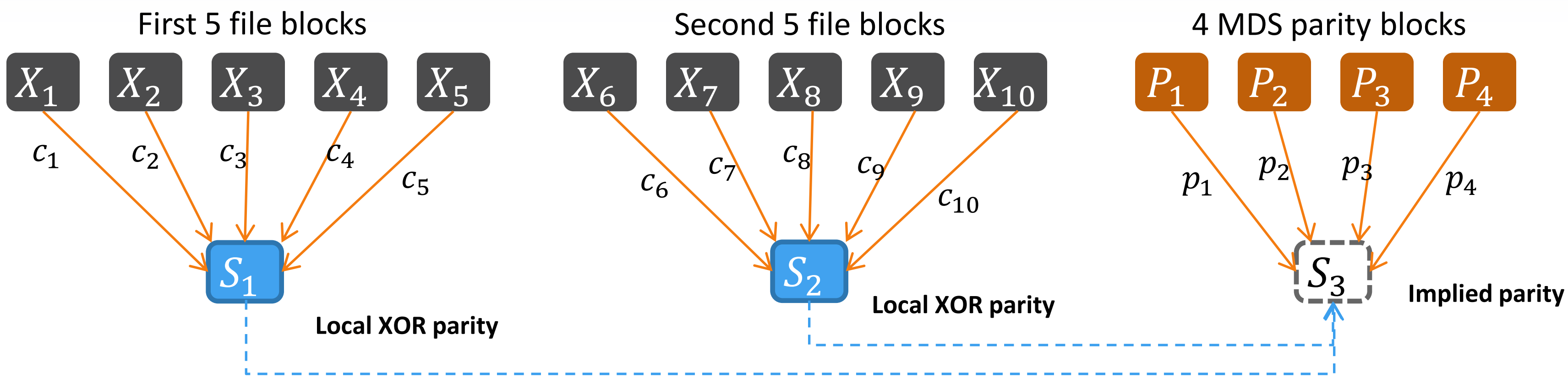


**With the evolution of erasure codes,
A small sacrifice in performance leads to efficiently
repairable good codes.**

A Case study – Facebook File system

[Locally repairable codes]

- Let us use a standard (14,10) MDS code.



- Calculate local parities:
  - $S_1 = c_1X_1 + c_2X_2 + c_3X_3 + c_4X_4 + c_5X_5$
  - $S_2 = c_6X_6 + c_7X_7 + c_8X_8 + c_9X_9 + c_{10}X_{10}$
- Also choose coefficients such that
  - $S_1 + S_2 = S_3$ [Implied parity – Not stored!]
- DISADVANTAGE : **Extra Storage Requirement!!!**

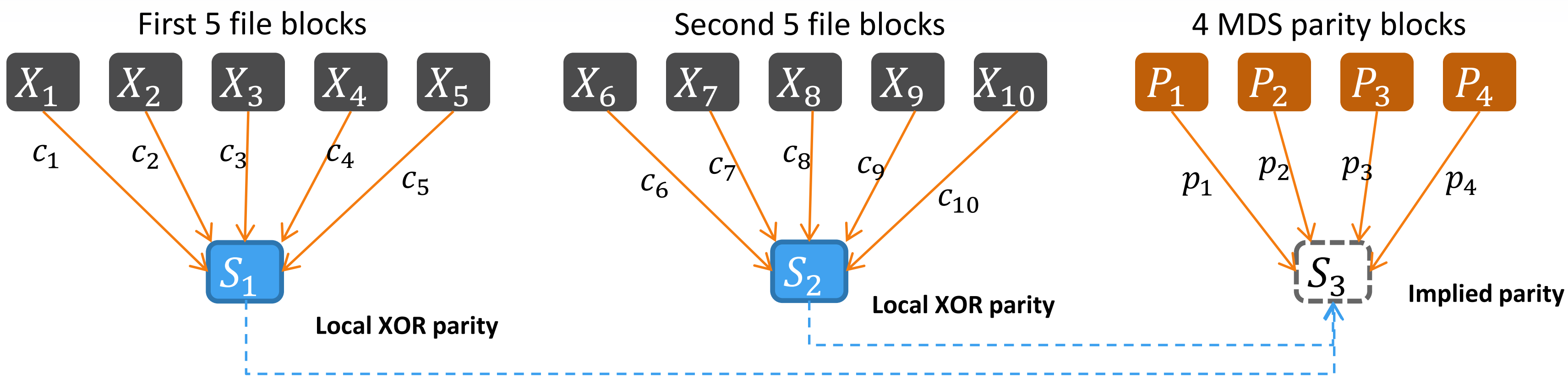
Overhead (RS) = 14/10 = 1.4

Overhead (LRC)= 14/10 = 1.6

A Case study – Facebook File system

[Locally Repairable Codes - LRC]

- Let us use a standard (14,10) MDS code.



- Calculate local parities:
  - $S_1 = c_1X_1 + c_2X_2 + c_3X_3 + c_4X_4 + c_5X_5$
  - $S_2 = c_6X_6 + c_7X_7 + c_8X_8 + c_9X_9 + c_{10}X_{10}$
- Also choose coefficients such that
  - $S_1 + S_2 = S_3$ [Implied parity – Not stored!]
- DISADVANTAGE : **Extra Storage Requirement!!!**

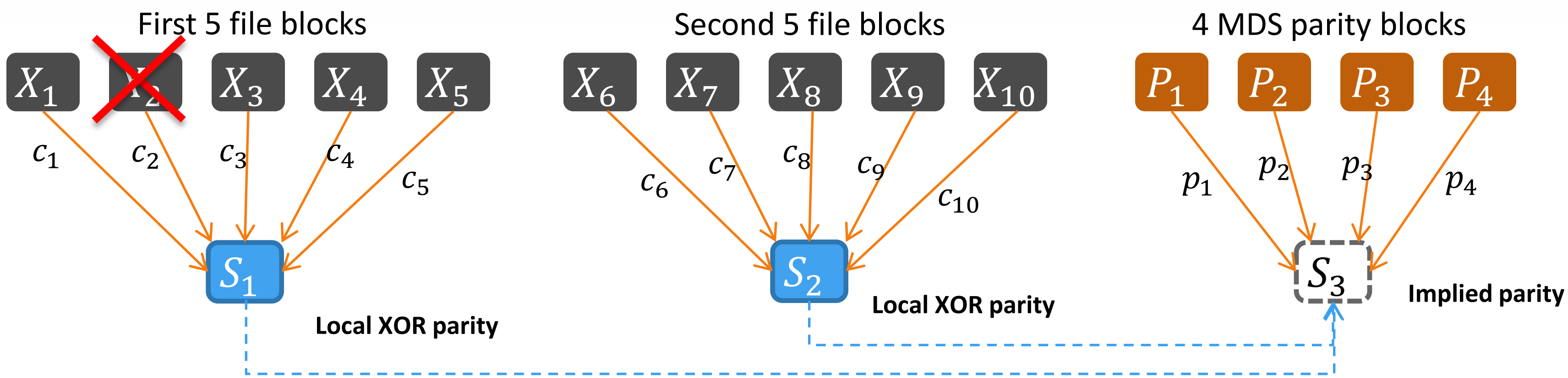
Overhead (RS) = 14/10 = 1.4

Overhead (LRC)= 16/10 = 1.6

A Case study – Facebook File system

[Locally Repairable Codes - LRC]

- **Suppose a failure occurs!**

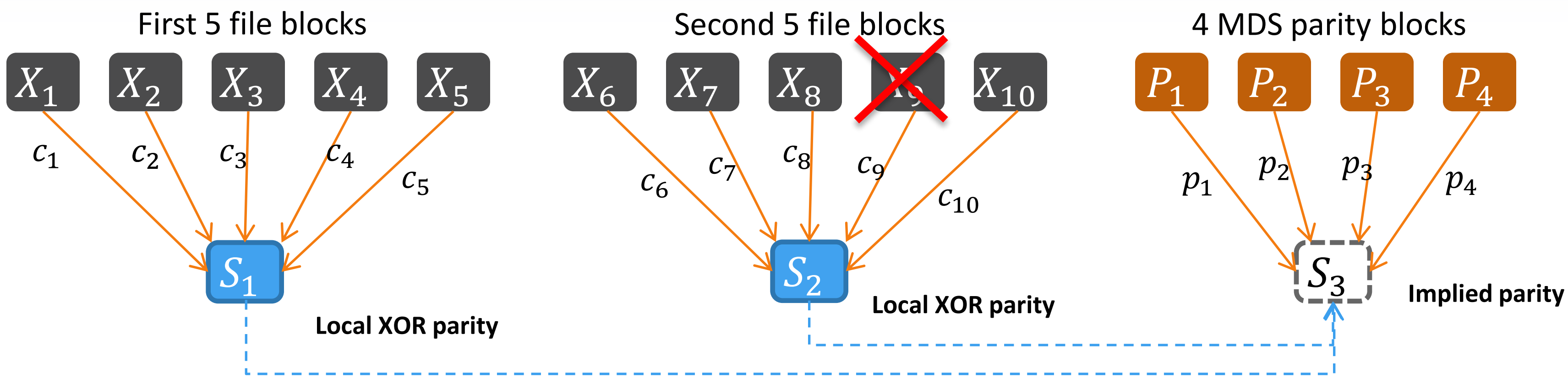


- Recovery equation:
  - $X_2 = {c_2}^{-1}(S_1 - c_1X_1 - c_3X_3 - c_4X_4 - c_5X_5)$
- Number of blocks that need to be accessed and read: 5!

A Case study – Facebook File system

[Locally Repairable Codes - LRC]

- **Suppose a failure occurs!**

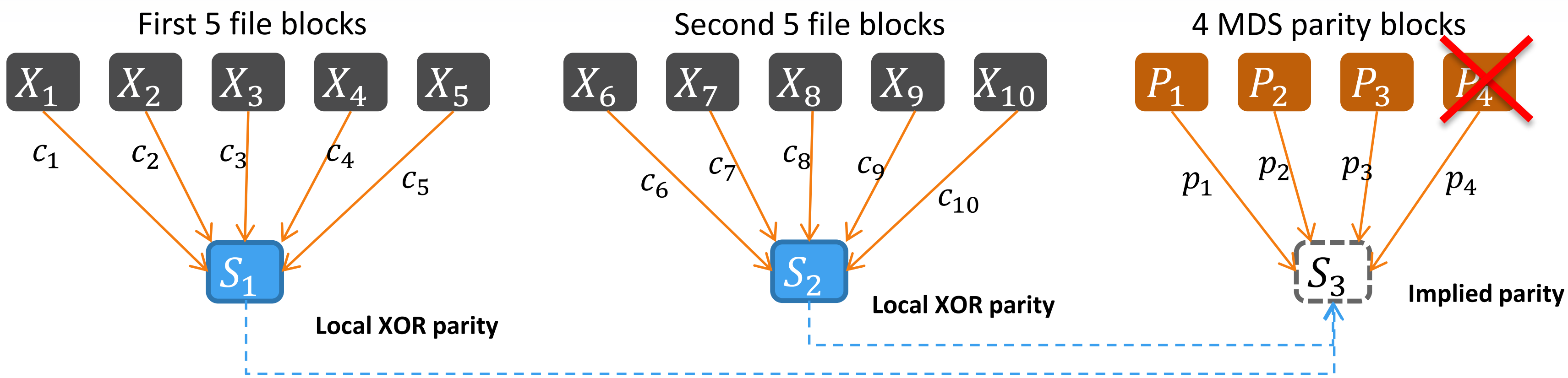


- Recovery equation:
  - $X_9 = c_9^{-1}(S_2 - c_6X_6 - c_7X_7 - c_8X_8 - c_{10}X_{10})$
- Number of blocks that need to be accessed and read: 5!

A Case study – Facebook File system

[Locally Repairable Codes - LRC]

- Suppose a failure occurs!

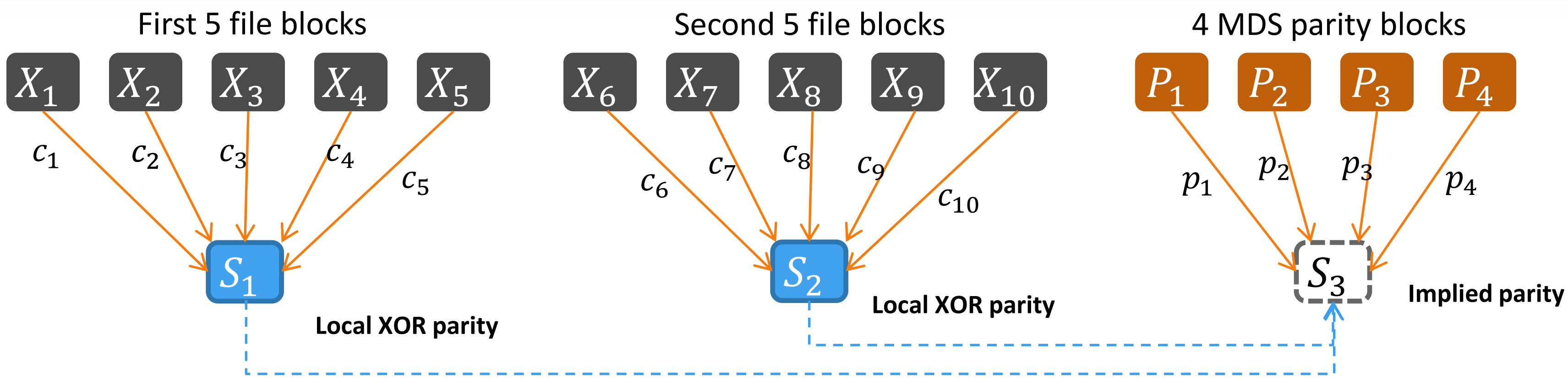


- Recovery equation:
  - $P_4 = {p_4}^{-1}(-S_1 - S_2 - p_1 P_1 - p_2 P_2 - p_3 P_3)$
- Number of blocks that need to be accessed and read: 5!

A Case study – Facebook File system

[Locally Repairable Codes - LRC]

- **Suppose a failure occurs!**



- Single failure : MIN:5 , MAX:5
- Double failures: MIN:9, MAX: 12
- Tripple failures: MIN: 12, MAX:12

LRC (16, 10)

- Single failure : MIN:12, MAX:12
- Double failures: MIN:12, MAX: 12
- Tripple failures: MIN: 12, MAX:12

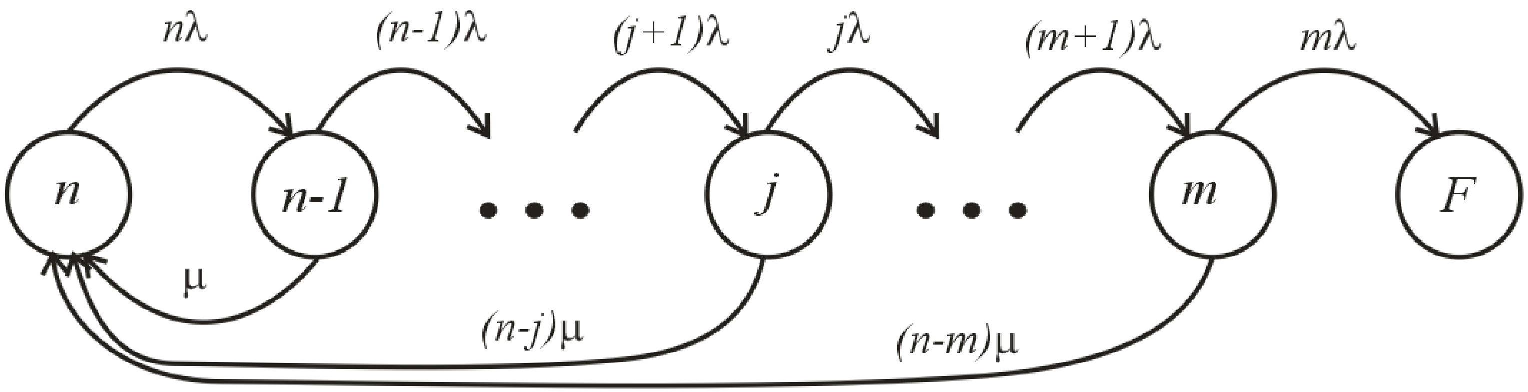
RS (14, 10)

Block locality

- **Definition**: (**Block locality**) An (n, k) code has a block locality l, when each block/unit is a function of at most l other blocks.
- Example: An (n, k) RS code has a block locality of k.
- We desire erasure codes with block locality $l \ll k$
- **Theorem 3**: (**locally repairable codes (LRC)**) There exists (n, k) locally repairable codes with block locality $\ln(k)$ that can correct $n - (1 + \epsilon)k$ erasures where $\epsilon = \frac{1}{\ln(k)} - \frac{1}{k}$.
  - **Example:** LT codes has an average symbol degree of $\ln(k)$ and therefore has an average block locality of $\ln(k)$ while achieving an optimal performance asymptotically.

How all these measures reflect to system performance?

- A storage system's reliability is usually measured in terms of mean time to failure (MTTDL) values.
- Assume we have $\boldsymbol{n}$ disks, $\boldsymbol{m}$ of which are used for data storage and $\boldsymbol{c} = \boldsymbol{n} - \boldsymbol{m}$ are used parity (failure protection).
- Conventionally, a Markov model is used (with some correction factors) to predict the MTTDL values.



- Each state represents the number of operational disks in the array. Transitions happen with each component having constant failure and repair rates λ and μ, respectively.

Reliability and the Markov Model

- This model assumes an $(n,\ m)$ MDS code that can correct up to $c\ =\ n-m$ erasures.
  - ASSUMPTIONS:
    1) Disk failures are independent
    2) Each disk failure and repair happens based on an exponential distribution (Poisson random process).
- **MTTDL is the expected time to enter state F.**

$$P_i(t)\text{: } probability\ of\ being\ in\ state\ i\ at\ time\ t$$

Reliability function

$$R(t) = \sum_{j=m}^{m+c} P_j(t)$$

MTTDL

$$MTTDL = \int_0^{\infty} R(t)dt$$

- Slight changes (a single disk repair at a time) can be made to the model, however these changes only slightly effect the MTTDL value.

Reliability and the Markov Model

- Due to assumption 1 and 2,
  - $MTTF = 1/\lambda$ and $MTTR = 1/\mu$
- Let $\omega = \dfrac{\mu}{\lambda} = \dfrac{MTTF}{MTTR}$.
- We have,

$$MTTDL = \frac{\omega^c}{\lambda\, m \binom{m+c}{c}}$$ **[1]**

- For $c = 1$ (RAID 5)

$$MTTDL = \frac{\mu}{\lambda^2\, m\,(m+1)} = \frac{MTTF^2}{m\,(m+1)MTTR}$$

- For $c = 2$ (RAID 6)

$$MTTDL = \frac{\mu^2}{\lambda^3\, \frac{1}{2} m\,(m+1)(m+2)} = \frac{MTTF^3}{\frac{1}{2} m\,(m+1)(m+2)MTTR^2}$$

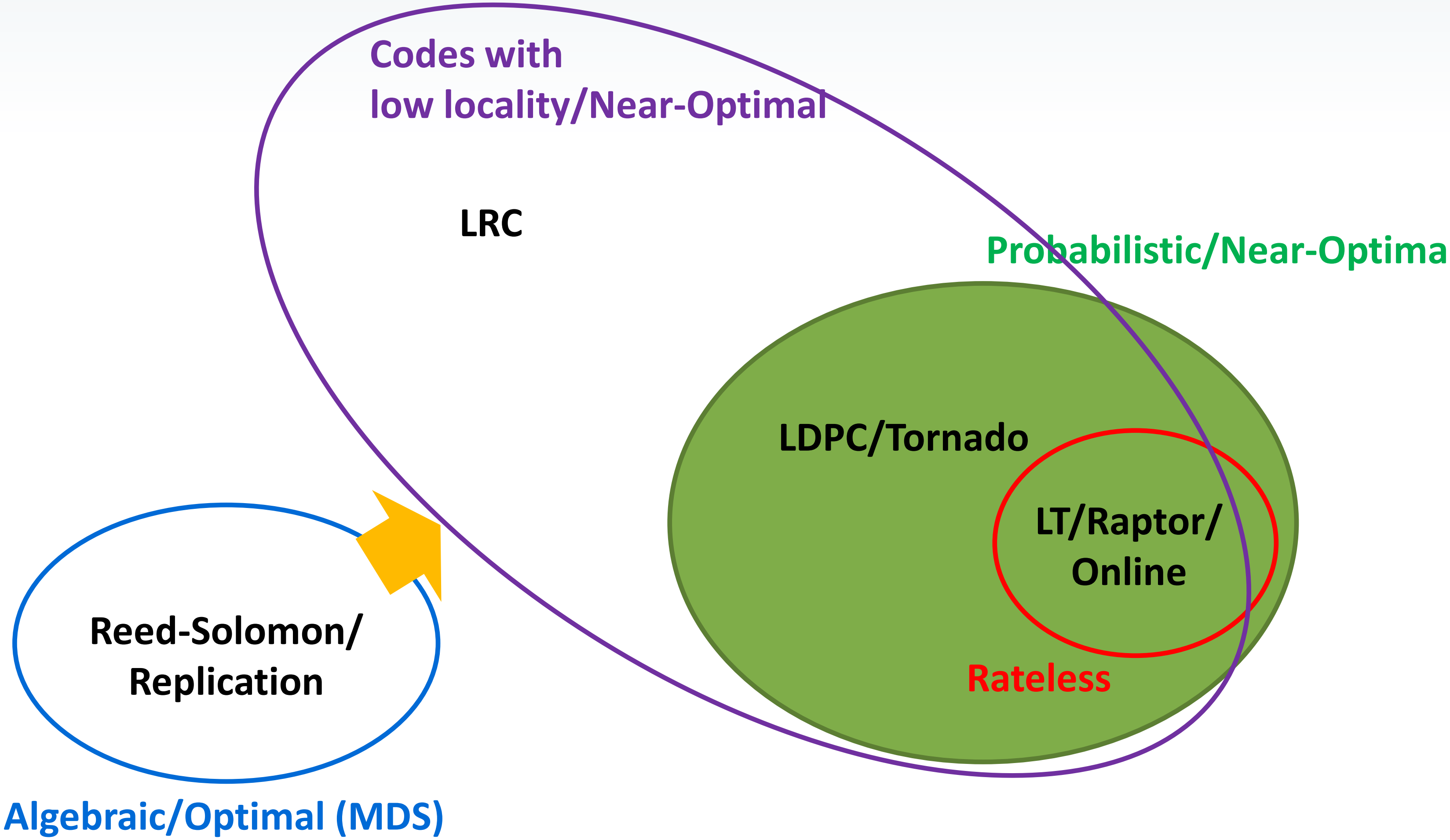
[2] W. Burkhard, and J. Menon, "Disk array storage system reliability". *Proceedings of the International Symposium on Fault-tolerant Computing*, pgs.432-441, 1993..

Numerical Results [2]

Erasure Code	Storage overhead	Repair traffic	MTTDL (days)
Replication (3, 1)	3x	1x	2.3e+10
RS (14, 10) – optimal	1.4x	10x	3.3e+10
LRC (16, 10)	1.6x	5x	1.2e+15

[3] M. Sathiamoorthy, M. Asteris, D.S. Papailiopoulos, A.G. Dimakis, R. Vadali, S. Chen, and D. Borthakur. Xoring elephants: Novel erasure codes for big data. In Proceedings of the VLDB Endowment, 2013

Evolution of Erasure Codes



OPTIMIZATION OF LT CODES FOR IMAGE TRANSFER

Suayb S. Arslan

suaybarslan@quantum.com

Quantum Corporation, 141 Innovation Dr., Irvine, CA, USA, 92612

Source quality assessment: Image compression

-Basics

- Given two images ***I*** and ***I'*** (original and the noisy version), the distortion will be measured by Mean Square Error (MSE):

$$MSE = \frac{1}{L_x \times L_y} \sum_{y=1}^{L_x} \sum_{x=1}^{L_y} [I(x,y) - I'(x,y)]^2$$

where L_x and L_y are dimensions of the image.

- Peak Signal to Noise Ratio (PSNR in dB) is defined to be

$$PSNR = 10 \times \log_{10} \left(\frac{I_{max}^2}{MSE} \right)$$

where I_{max} is the maximum possible intensity value of the image.

- For monochromatic gray scale image: $I_{max} = 255$
- Lower MSE (larger PSNR) means better image quality.
- "Source rate" means the average number of bits spent per pixel (bpp). For a given PSNR value, the lower the source rate is, the better the compression will be.

Progressive Source Compression

-Introduction

Progressive bit stream

0.01bpp, PSNR=22.55dB

- 4% gives you only a low quality representation of the source.

Progressive Source Compression

-Introduction

Progressive bit stream

0.01bpp, PSNR=22.55dB

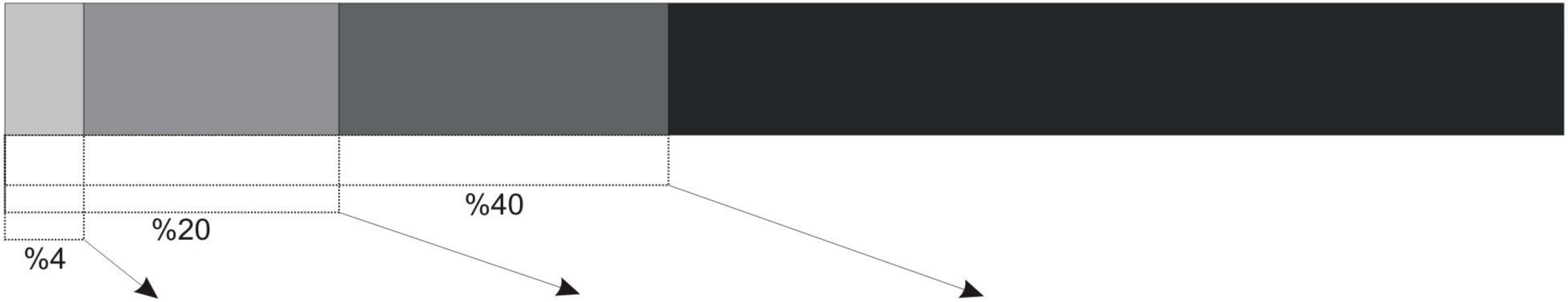
0.05bpp, PSNR=27.17dB

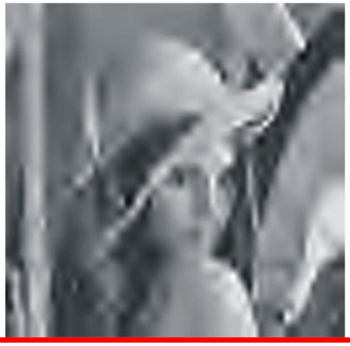
- 20% gives better image quality compared to 4% case.

Progressive Source Compression

-Introduction

Progressive bit stream





0.01bpp, PSNR=22.55dB

0.05bpp, PSNR=27.17dB

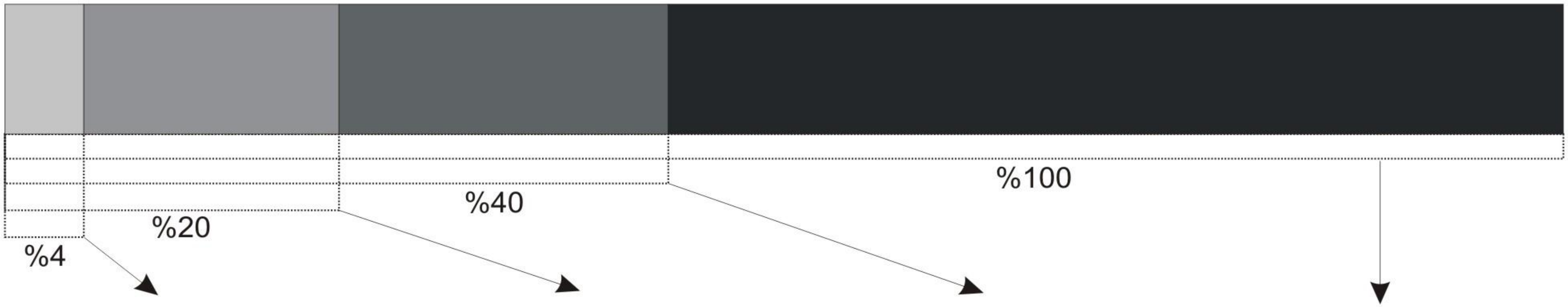
0.1bpp, PSNR=29.81dB

- Using only 40% of the total bit stream, a good quality image is obtained.

Progressive Source Compression

-Introduction

Progressive bit stream



0.01bpp, PSNR=22.55dB

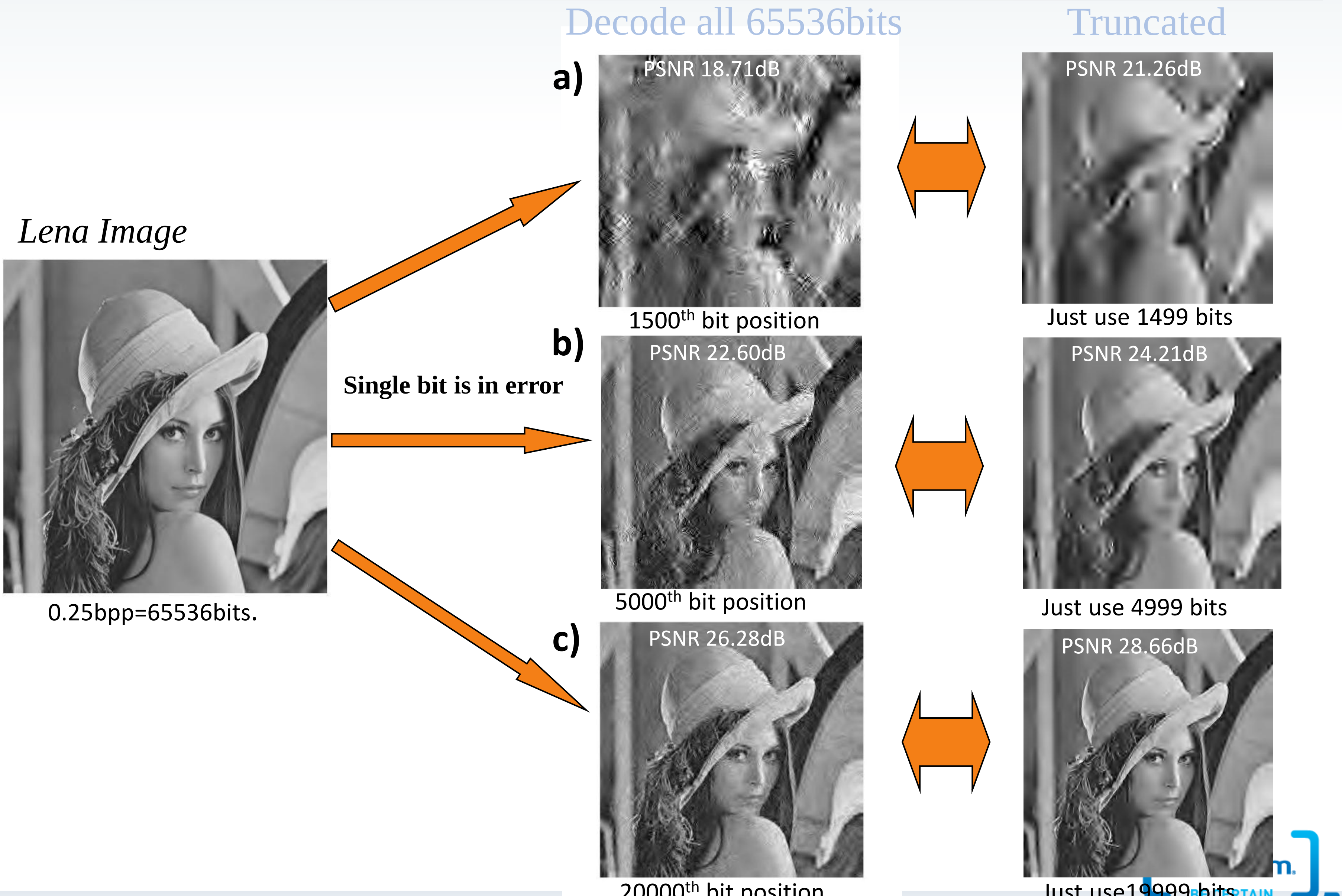
0.05bpp, PSNR=27.17dB

0.1bpp, PSNR=29.81dB

0.25bpp, PSNR=33.68dB

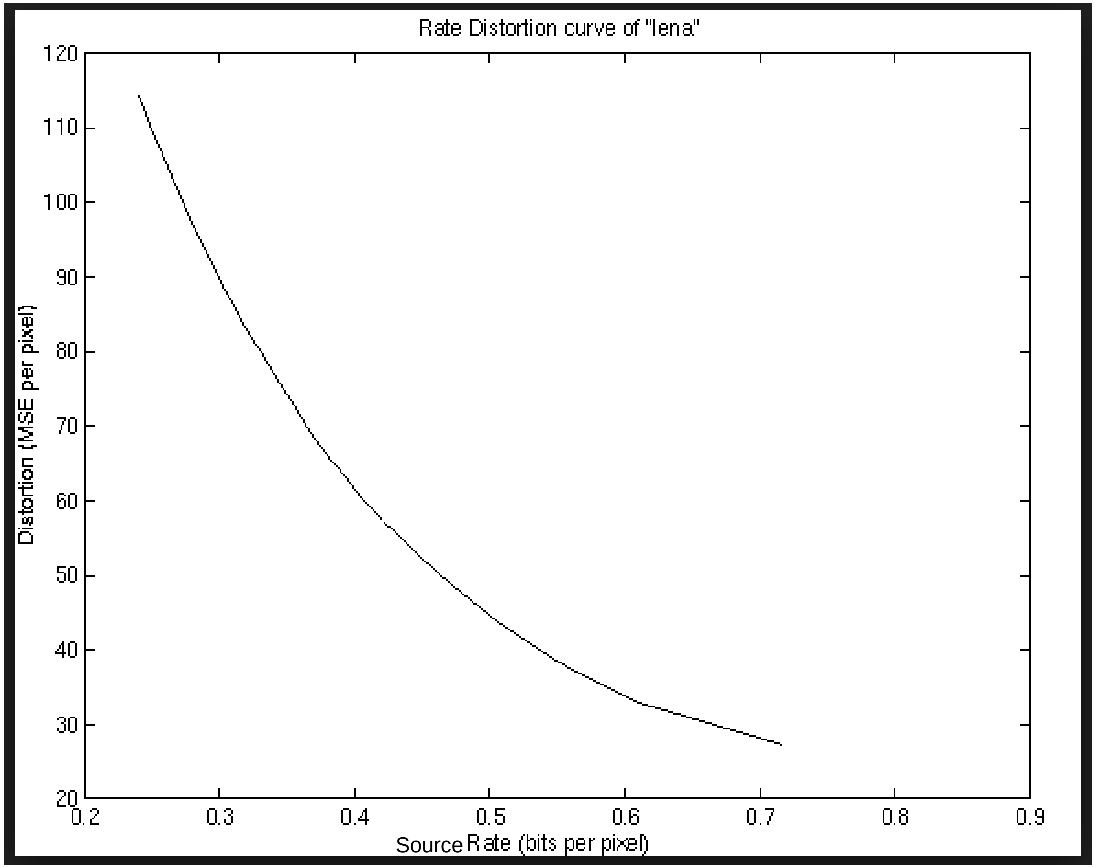
- At 100%, the image quality is improved further but no major difference from 40%.
- Examples: SPIHT, EZW, JPEG2000 etc.
- Disadvantage: Very sensitive to bit errors.
- Unequal Error Protection (UEP) is achieved by channel coding.

Error Propagation



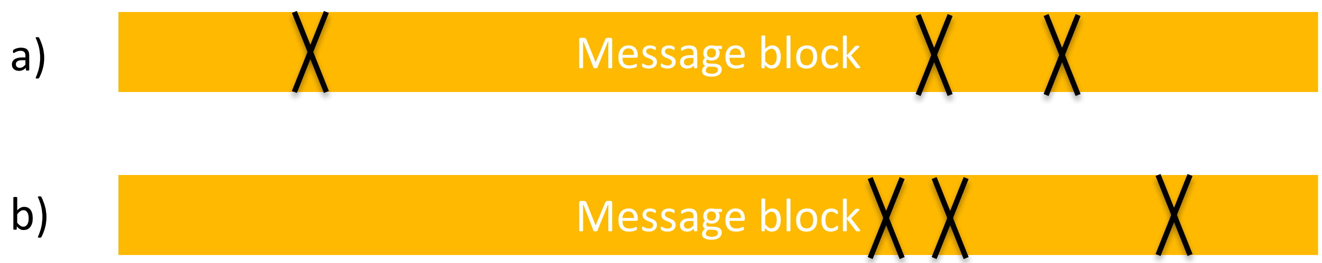
Rate-Distortion Curve

- In a lossy data compression, R-D curve pictures the relationship between the source rate and the distortion for a given source and encoder/decoder pair.



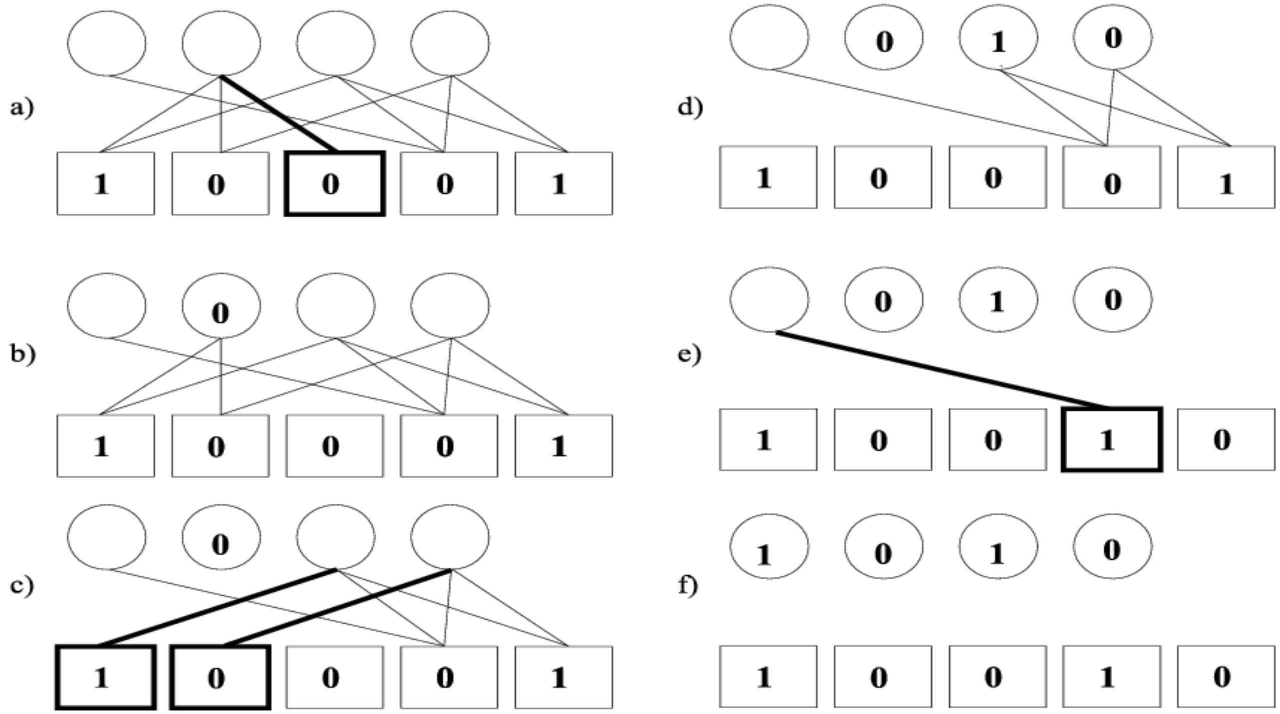
Design objective of the erasure code

- Decoding the whole message block? If $c < 1$, this is not even possible with optimal codes!
- Decoding a fraction of the message block? What fraction?



- Both have the same number of unrecoverable errors. However **b)** will provide better image quality!
- Need unequal protection/unequal recovery time.

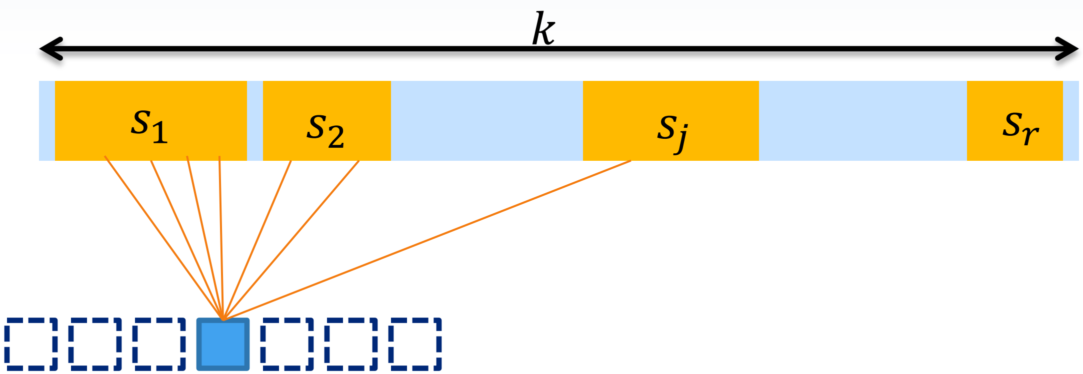
Belief Propagation



- Let us observe the following:
  - *Decoding stage 1:* A degree-1 check node decodes an information symbol.
  - *Decoding stage 2:* Some of the degree-2 check nodes decode two information symbols.
  - *Decoding stage 3:* A degree-3 check node decodes an information symbol.
- **Conclusion:** low degree coded symbols decode information symbols earlier (early iterations) in the decoding algorithm.
- This can be used for prioritized decoding.

Idea

- First step: divide the message block into multiple subblocks (r subblocks).



$d = 7$

- Second step: For each symbol generated: Choose a degree according to a suitable degree distribution.
- Let $p_{j,i}$ be the conditional probability of choosing any information symbol in s_j given the degree of the coded symbol is i.

$$\mathbf{P}_{r\times k} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,k} \\ \vdots & \vdots & \cdots & \vdots \\ p_{r-1,1} & p_{r-1,2} & \cdots & p_{r-1,k} \\ p_{r,1} & p_{r,2} & \cdots & p_{r,k} \end{bmatrix}$$

[4] S. S. Arslan, P. Cosman, and L. Milstein, “Generalized unequal error protection LT codes for progressive data transmission,” *IEEE Trans. Image Processing*, vol. 21, no. 8, pp. 3586–3597, Aug. 2012.

- Second step: Choose edges according to $\mathbf{P}_{r\ x\ k}$

Idea

- Number of unknowns: $(r-1)k$ i.e., it scales with k.
- To reduce the number of unknowns, we introduce an exponential dependence:

Motivation: *exponential-like RD characteristics*

choose $p_{j,i}$ to be an exponential function of the degree number i for $j = 1, 2, \ldots, r-1$ as follows:

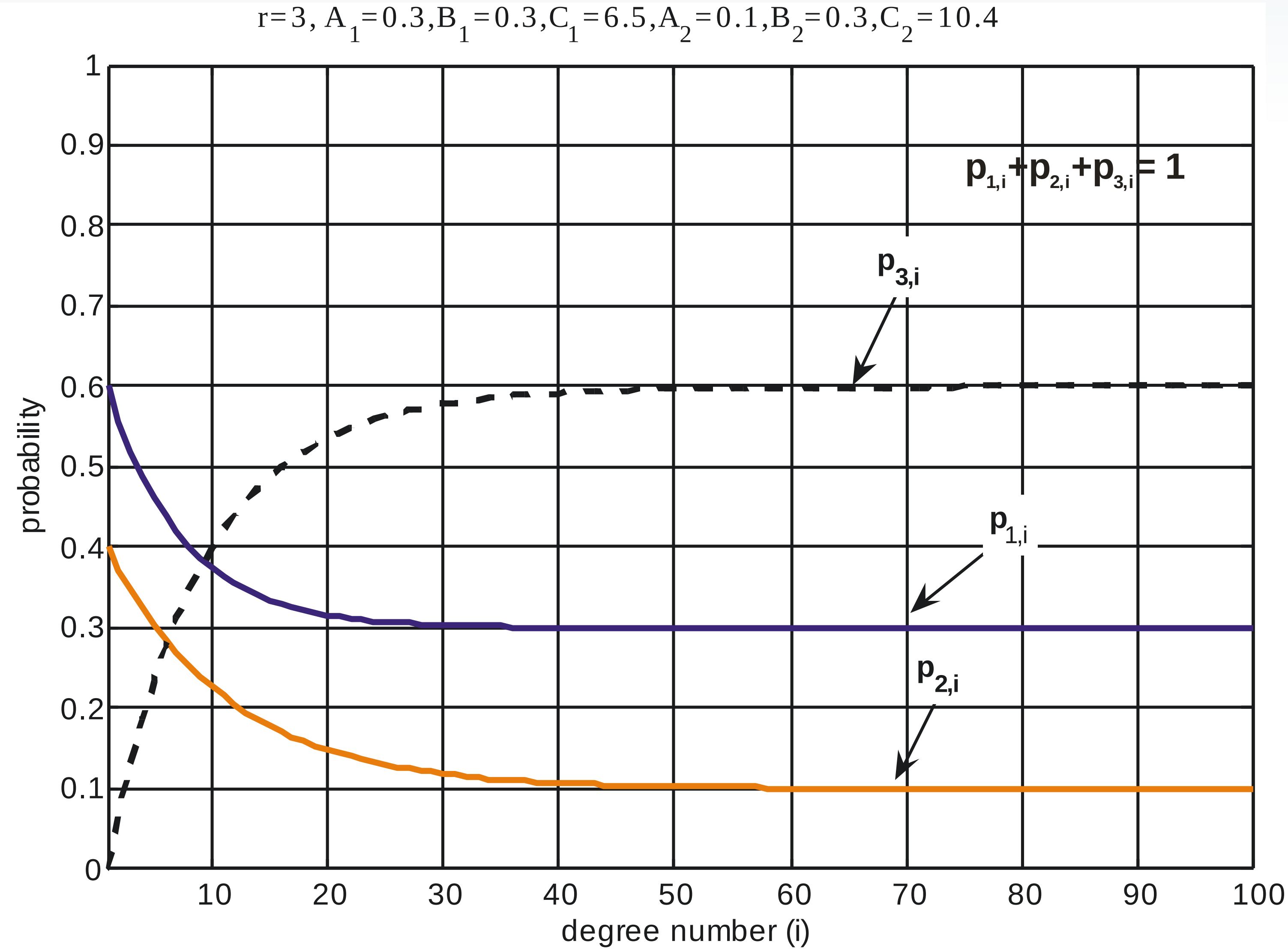
Definition *Exponential* SD

- $$p_{j,i} = A_j + B_j \times \exp\left\{-\frac{i-1}{C_j}\right\} \text{ for } i = 1, 2, \ldots, k$$

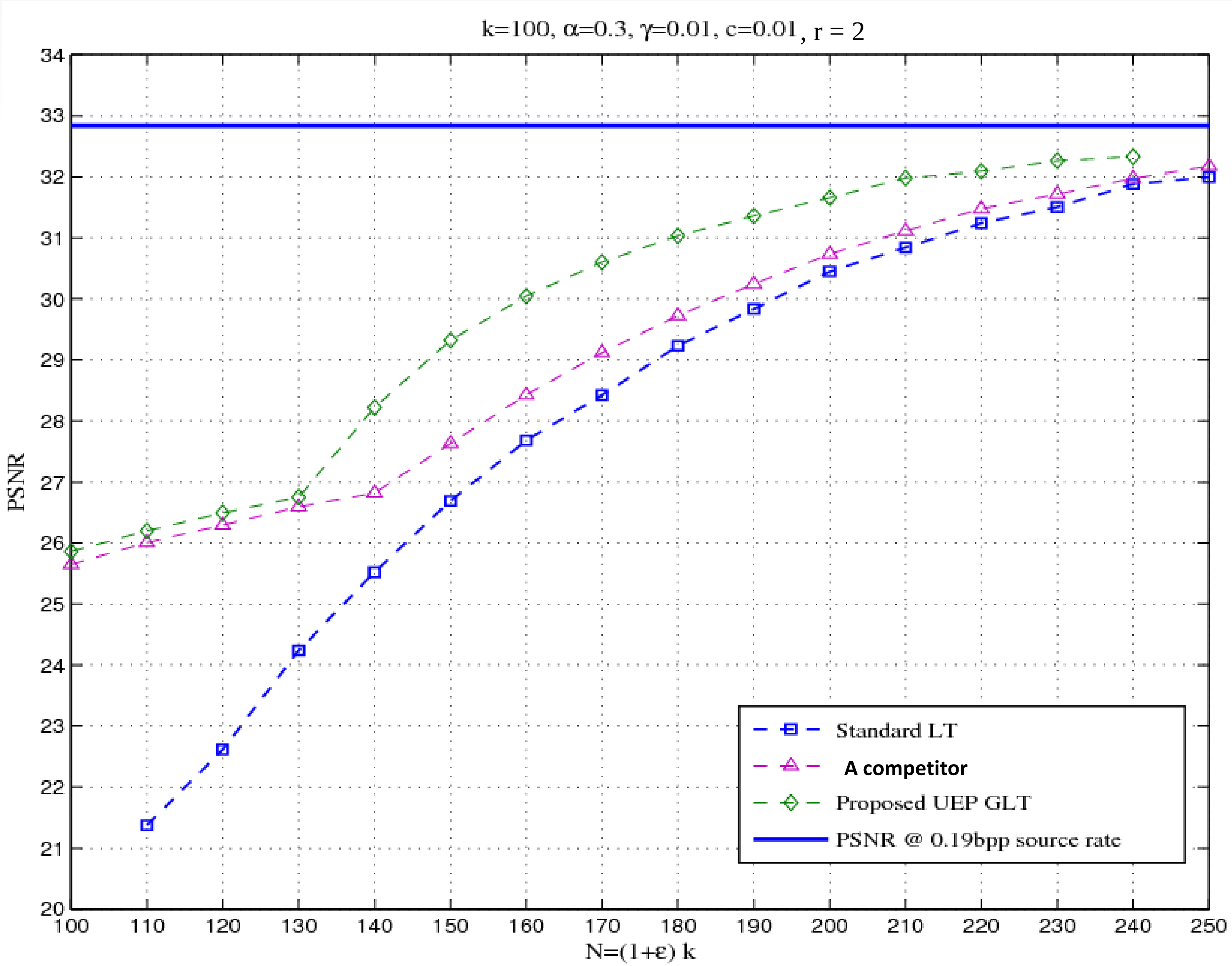
where $\{A_j \geq 0, B_j \geq 0, C_j \geq 0\}_{j=1}^{r-1}$ are design parameters satisfying $\sum_{j=1}^{r} p_{j,i} = 1$ for all i.

- Number of parameters are reduced to $3(r-1)$.

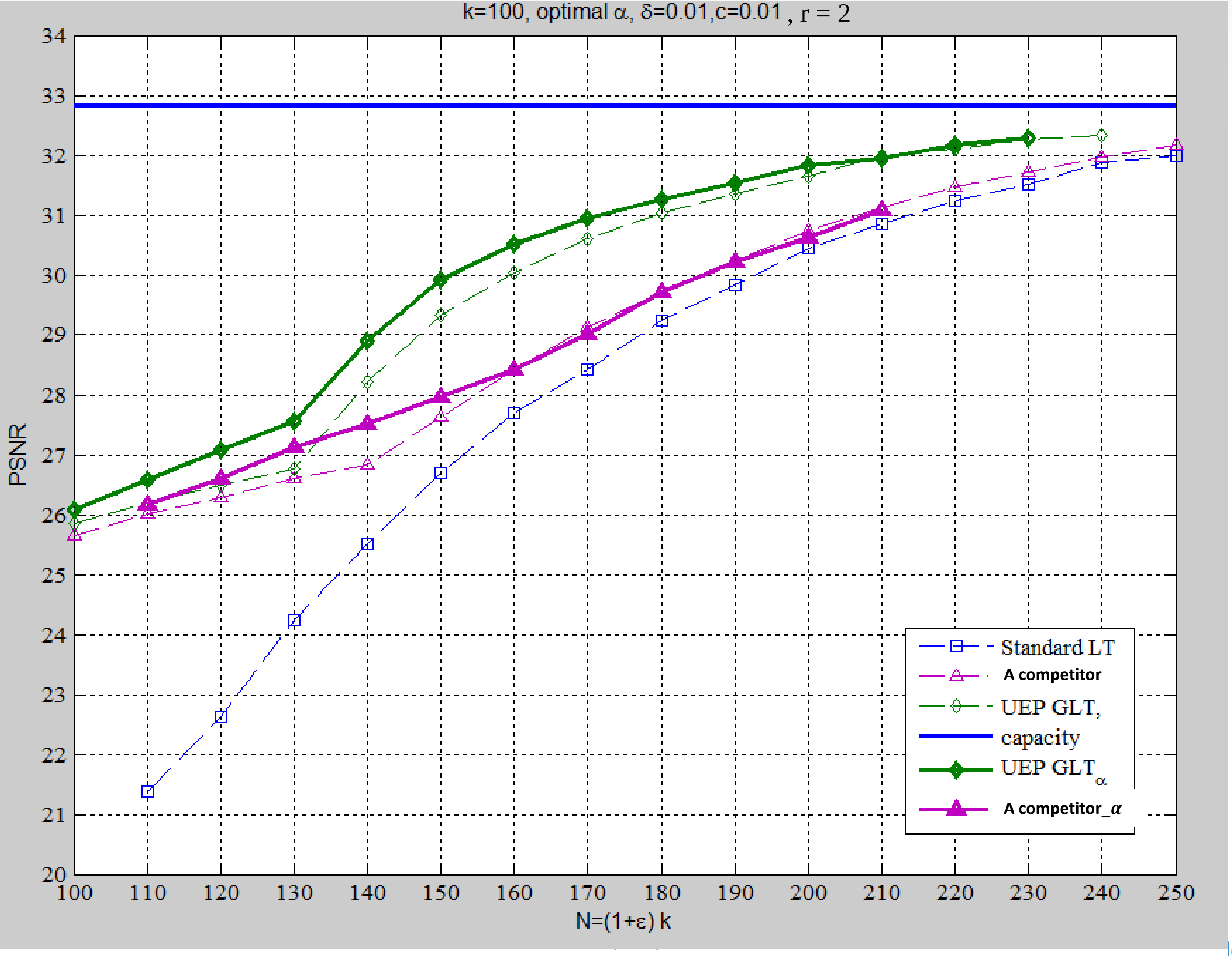
Idea


r=3, A_1=0.3,B_1=0.3,C_1=6.5,A_2=0.1,B_2=0.3,C_2=10.4
p_1,i+p_2,i+p_3,i= 1
p_3,i
p_1,i
p_2,i
probability
degree number (i)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80
90
100

Simulation result



Simulation result



Conclusions

- Depending on the application, the evolution of erasure codes have taken different directions.
- Different types of erasure codes are considered for different types of applications.
- For storage applications, main trend is to design codes with near-optimal performance in terms of efficiency with reduced bandwidth requirements while making sure that the error probabilities are under some target.
- For multimedia applications, main trend is to maximize the transfer multimedia quality or minimize the distortion.
- Optimization of the parameters of the erasure code can increase performance.

THANKS!

BE CERTAIN