



# Coding and Allocation for Distributed Data Storage: Fundamental Tradeoffs, Coding Schemes and Allocation Patterns

Iryna Andriyanova

## ► To cite this version:

Iryna Andriyanova. Coding and Allocation for Distributed Data Storage: Fundamental Tradeoffs, Coding Schemes and Allocation Patterns. Doctoral. Tutorial at the Swedish Communications Technologies Workshop 2013, Gothenburg, Sweden. 2013. cel-01263287

**HAL Id: cel-01263287**

**<https://hal.science/cel-01263287>**

Submitted on 27 Jan 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# Coding and Allocation for Distributed Data Storage

Fundamental Tradeoffs, Coding Schemes and Allocation Patterns

---

Iryna Andriyanova

ETIS Lab, ENSEA/ University of Cergy-Pontoise/ CNRS  
Cergy-Pontoise, France



August 21, 2013

# Coding for Distributed Storage: Why Do We Care

---

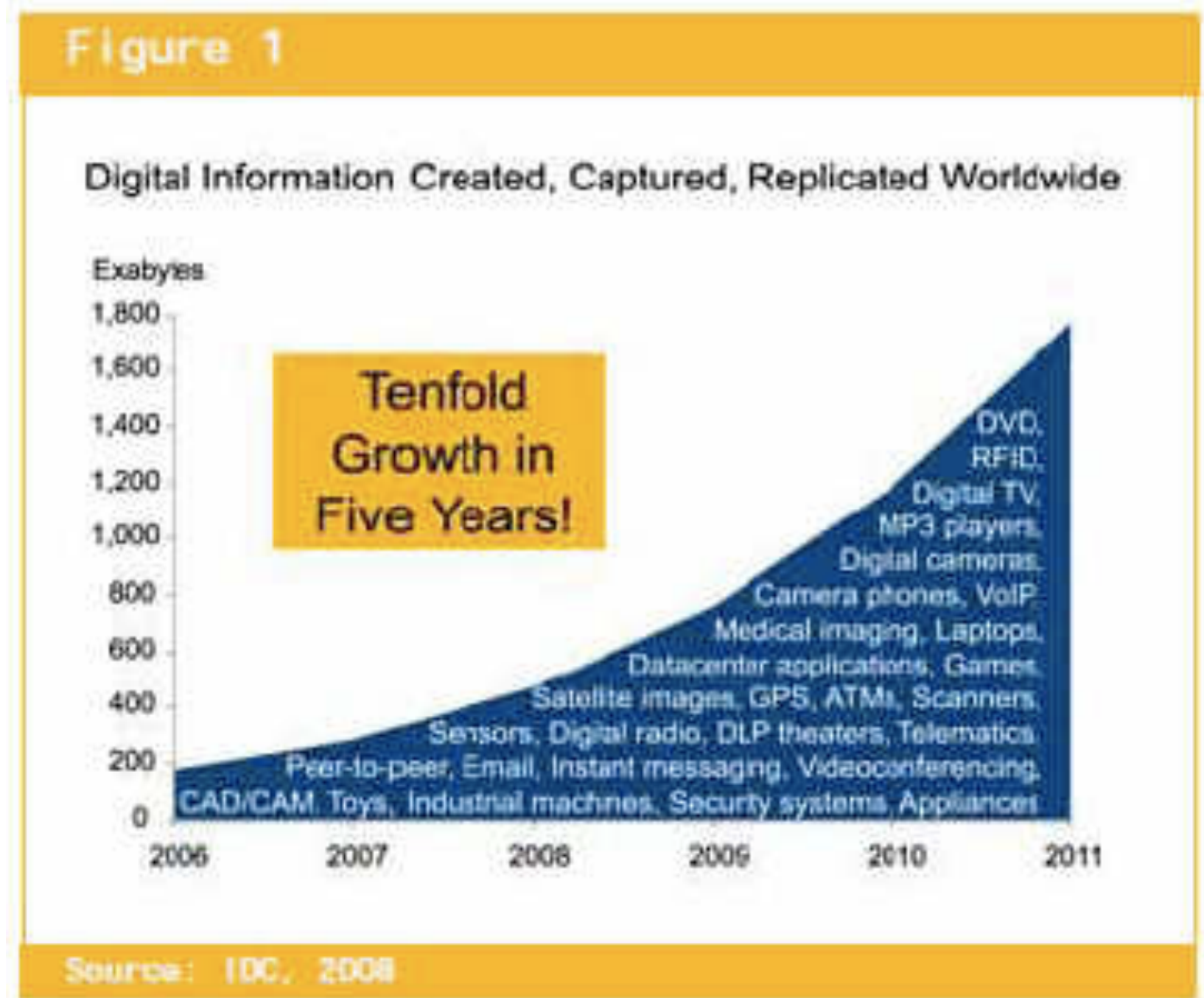
1) Shift of technology from expensive, high-end centralized NAS/SAN to a network of cheap storage devices



Issue: to provide a fault tolerance solution  
(storage-efficient)

# Coding for Distributed Storage: Why Do We Care

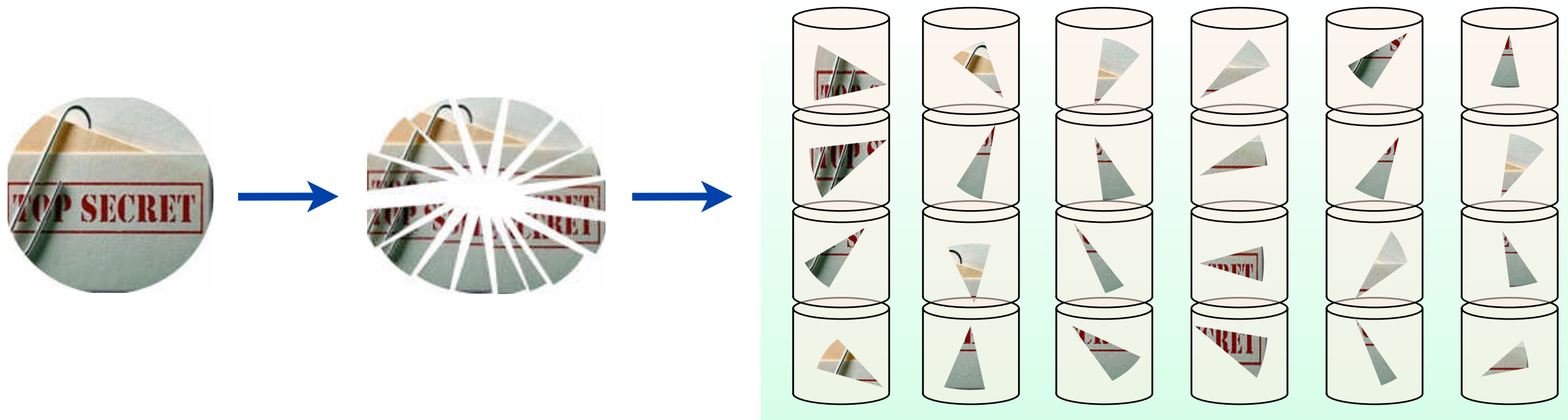
## 2) Arrival of big data



Issue: to provide low cost solutions for data storing, accessing, repairing, updating

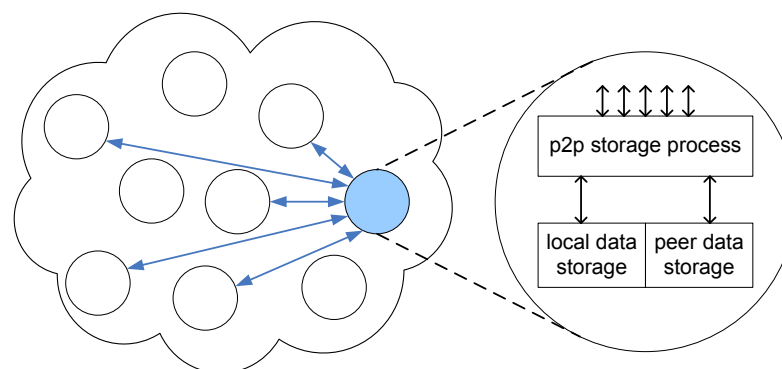
# When It Started

1987: RAID by Patterson, Gibson, Katz



**Redundant Array of Independent Disks**

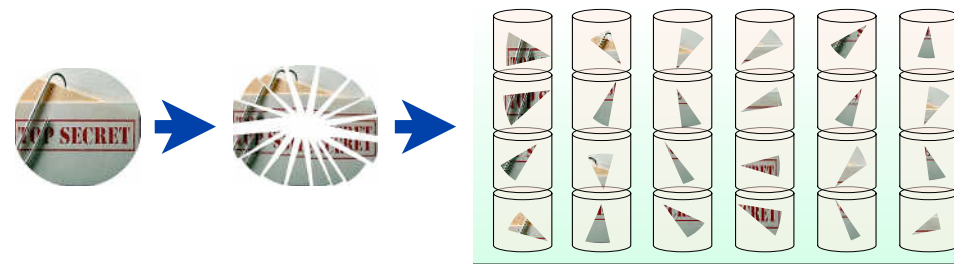
Decentralized solutions.





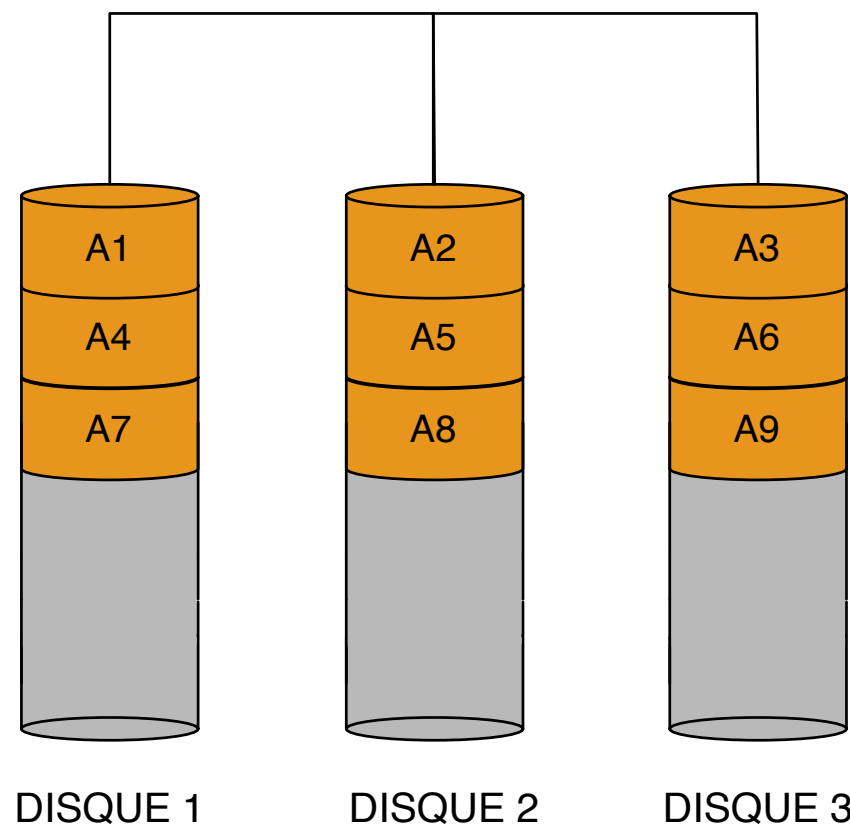
# When It Started

1987: by Patterson, Gibson, Katz



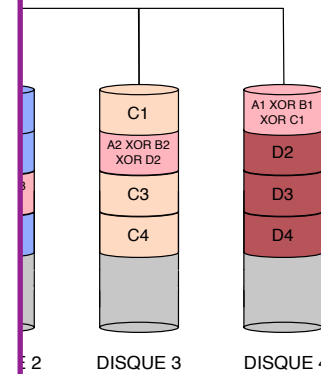
## RAIDs:

### RAID 0

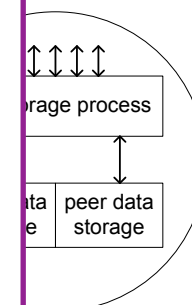
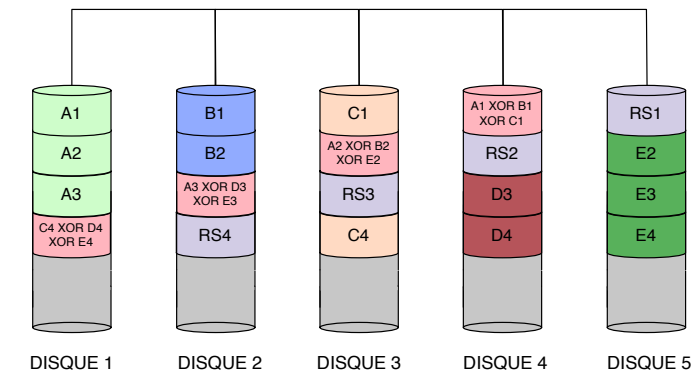


**no coding**

### RAID 5

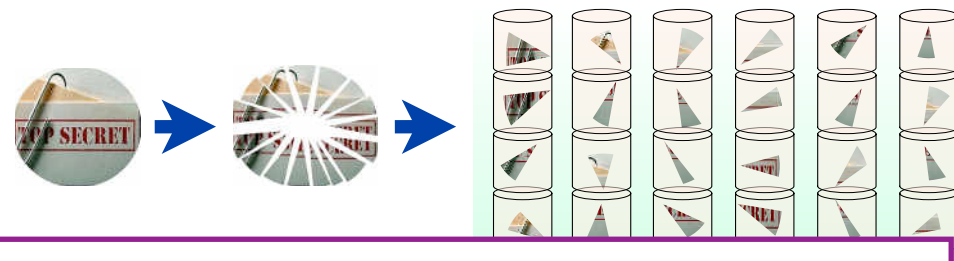


### RAID 6

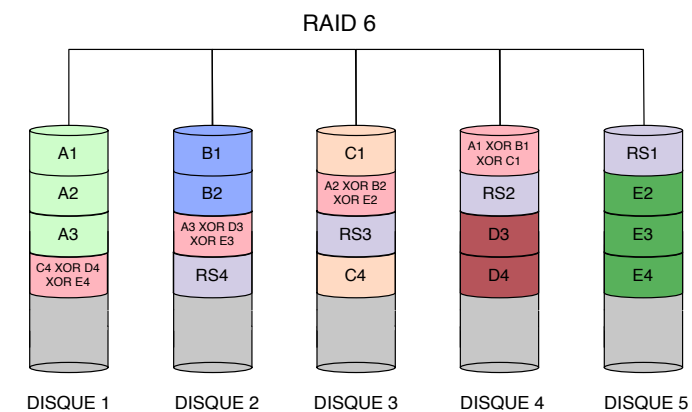
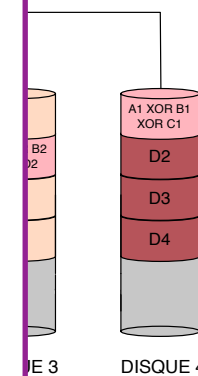
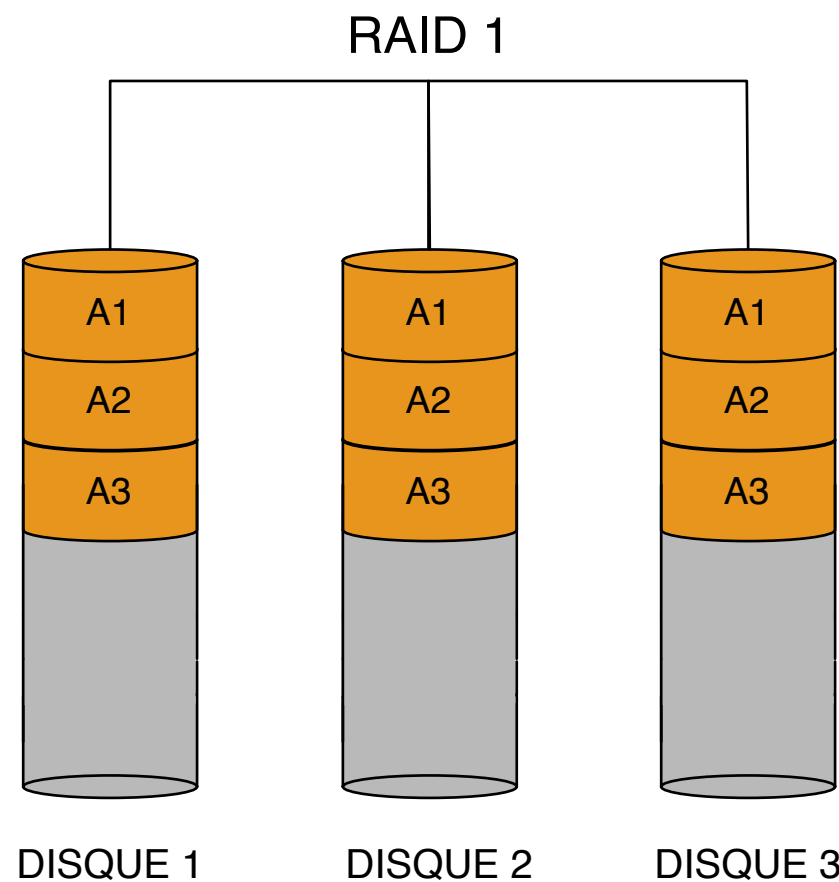
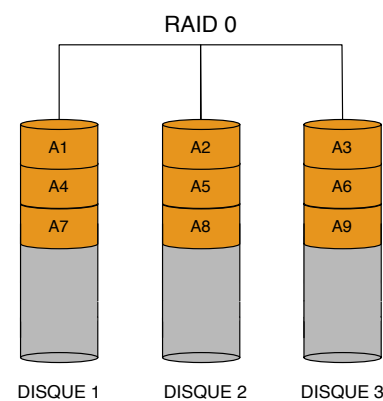


# When It Started

1987: by Patterson, Gibson, Katz



## RAIDs:



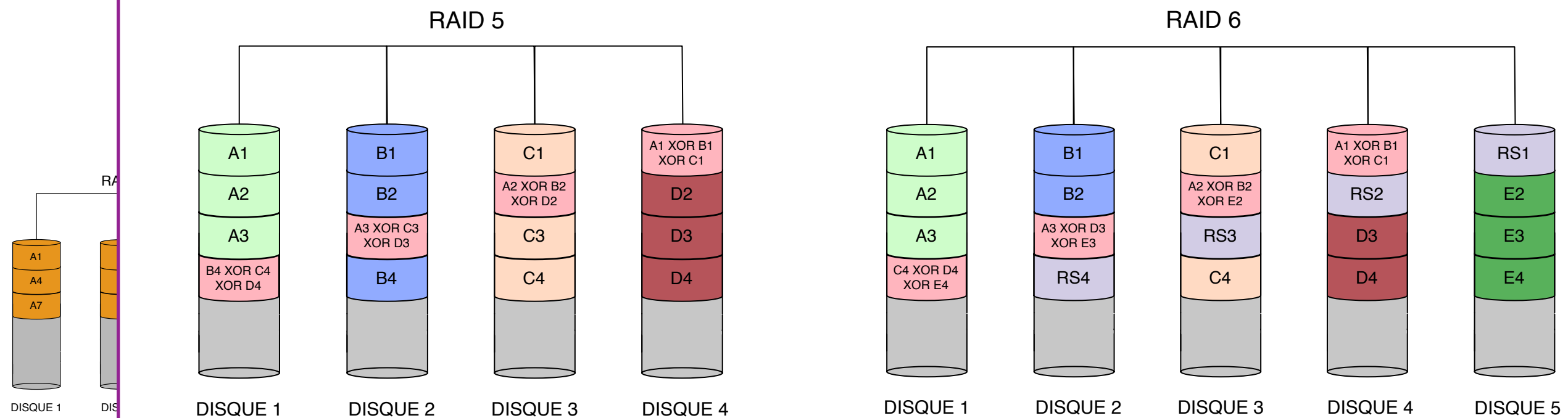
Decentral

replication

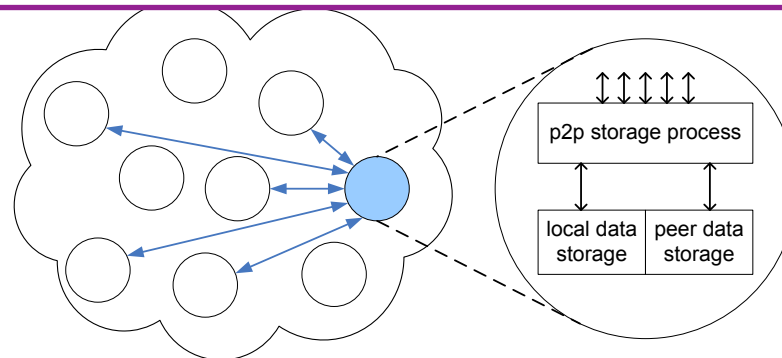
# When It Started

1987 by Patterson Gibson Katz

## RAIDs:



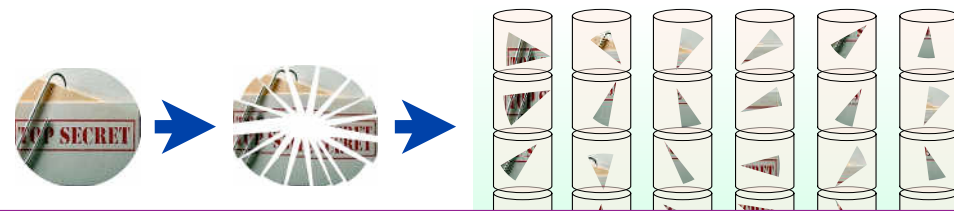
**erasure-correction coding**





# When It Started

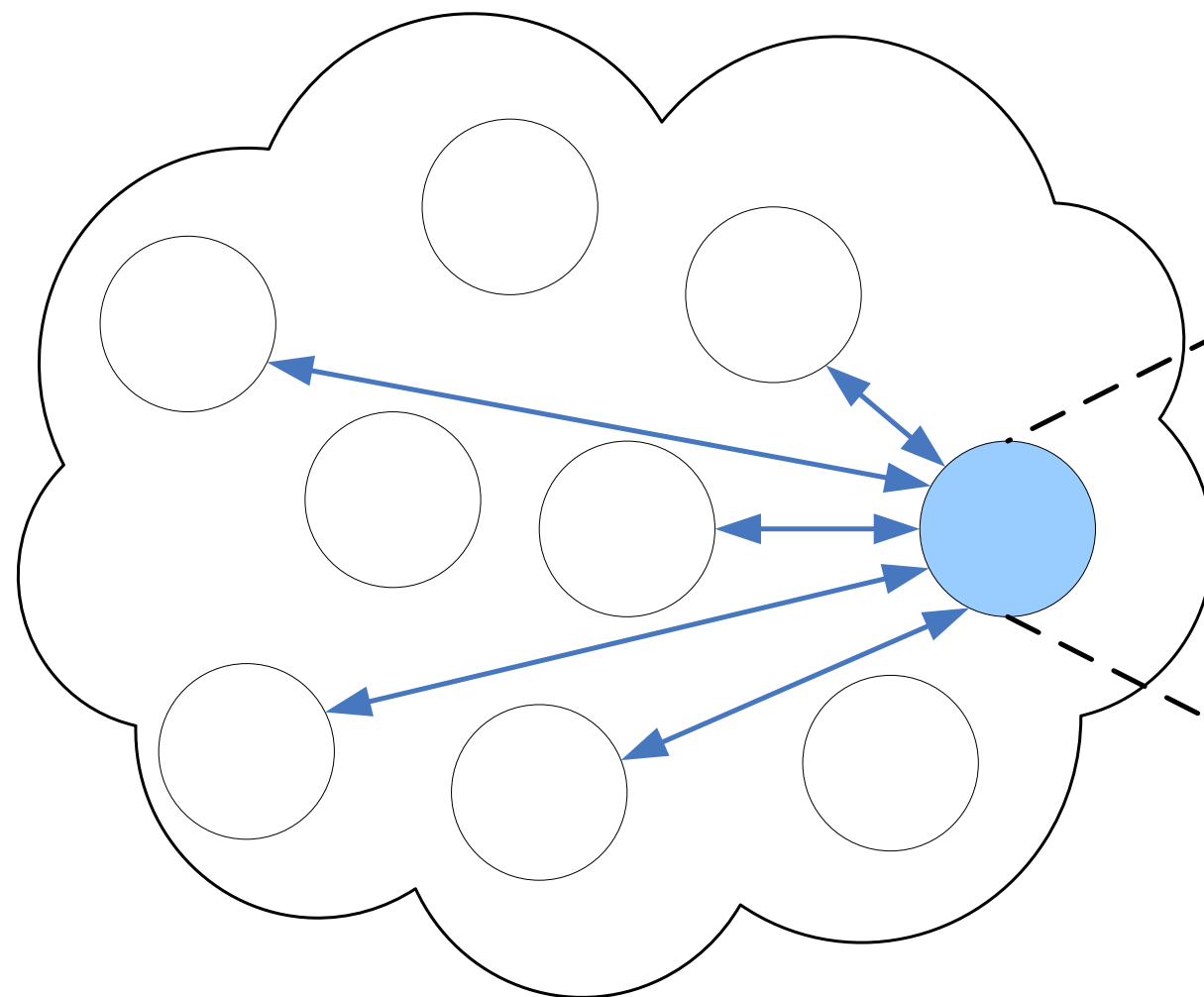
1987: by Patterson, Gibson, Katz



## Decentralized solutions:

R

DISC



**erasure-correction coding**

RS1  
E2  
E3  
E4  
DISQUE 5

# Distributed Storage (DS) Platforms and Services

---

**Platforms** of distributed file systems (DFSs):

GFS, Hadoop HDFS, Azure, ...

**DS products:**

Google's Drive, Apple's iCloud, Dropbox, Microsoft's SkyDrive, SugarSync, Amazon Cloud Drive, OVH's Hubic, Canonical's Ubuntu One, Carbonite, Symform, ADrive, EMC's Mozy, Wuala,...

**“Giants of storage”:**

Google, Facebook, Amazon, ....

# Open Issues

---

- \* Storing large amounts of data in an efficient way
- \* Design of efficient network protocols, reducing the total network load
- \* Dealing with the data already stored
- \* Flexibility of storage parameters
- \* Efficient data update
- \* Security issues
- \* Queueing, quality of service

# About This Tutorial

---

Most of open problems can be rewritten in terms of parameters of underlying erasure-correction code and allocation protocol.

Hence, we focus on:

- **coding** for distributed storage
- **allocation** in DS networks

P.S. Credits to Alan Julé (ETIS Lab) for his help.

# Plan

---

1. Data storage models and parameters
2. Coding for distributed storage (DS)
3. Allocation problem
4. Some new problems in DS

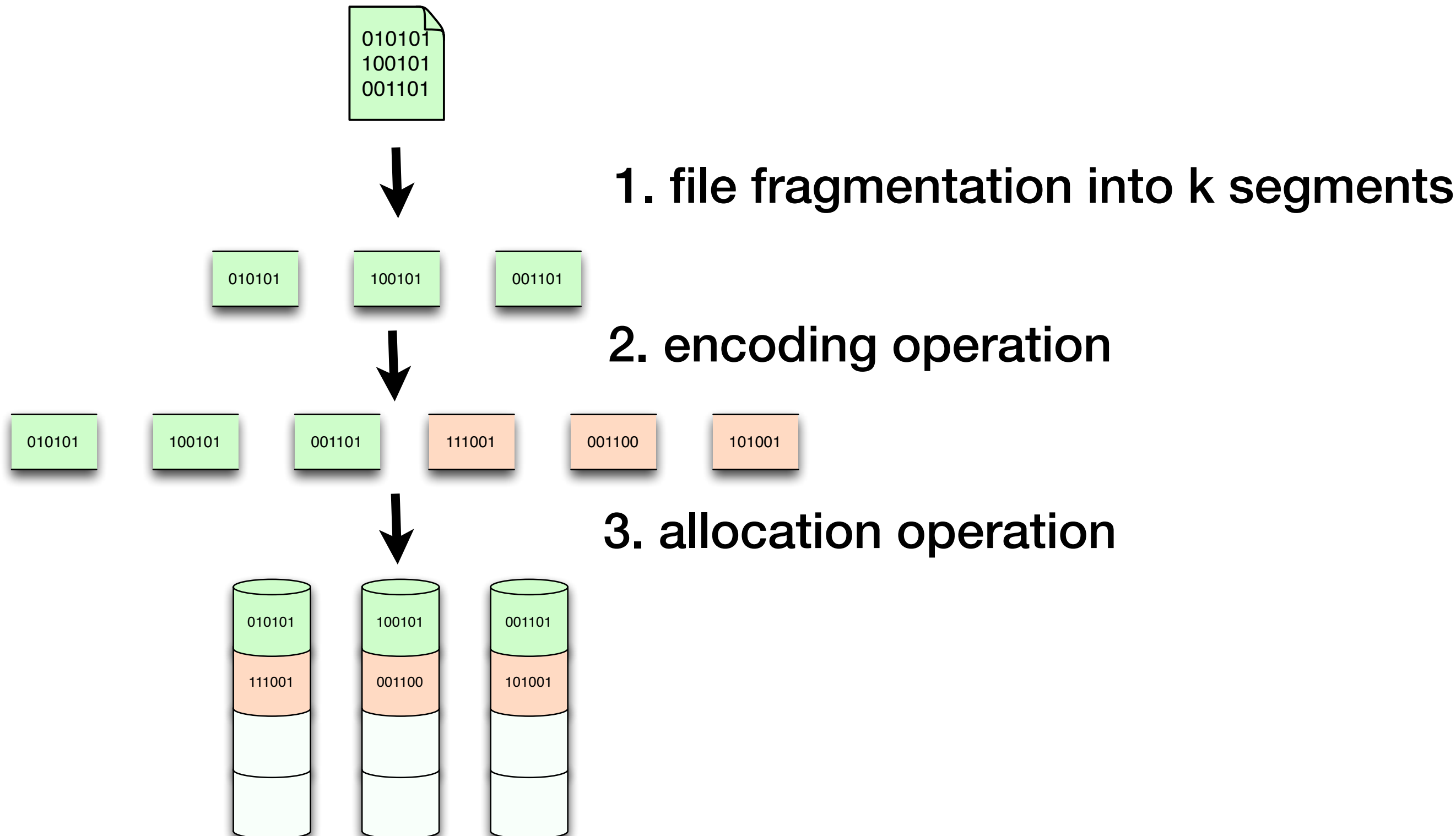
Discussion

# Part 1: Data Storage Models

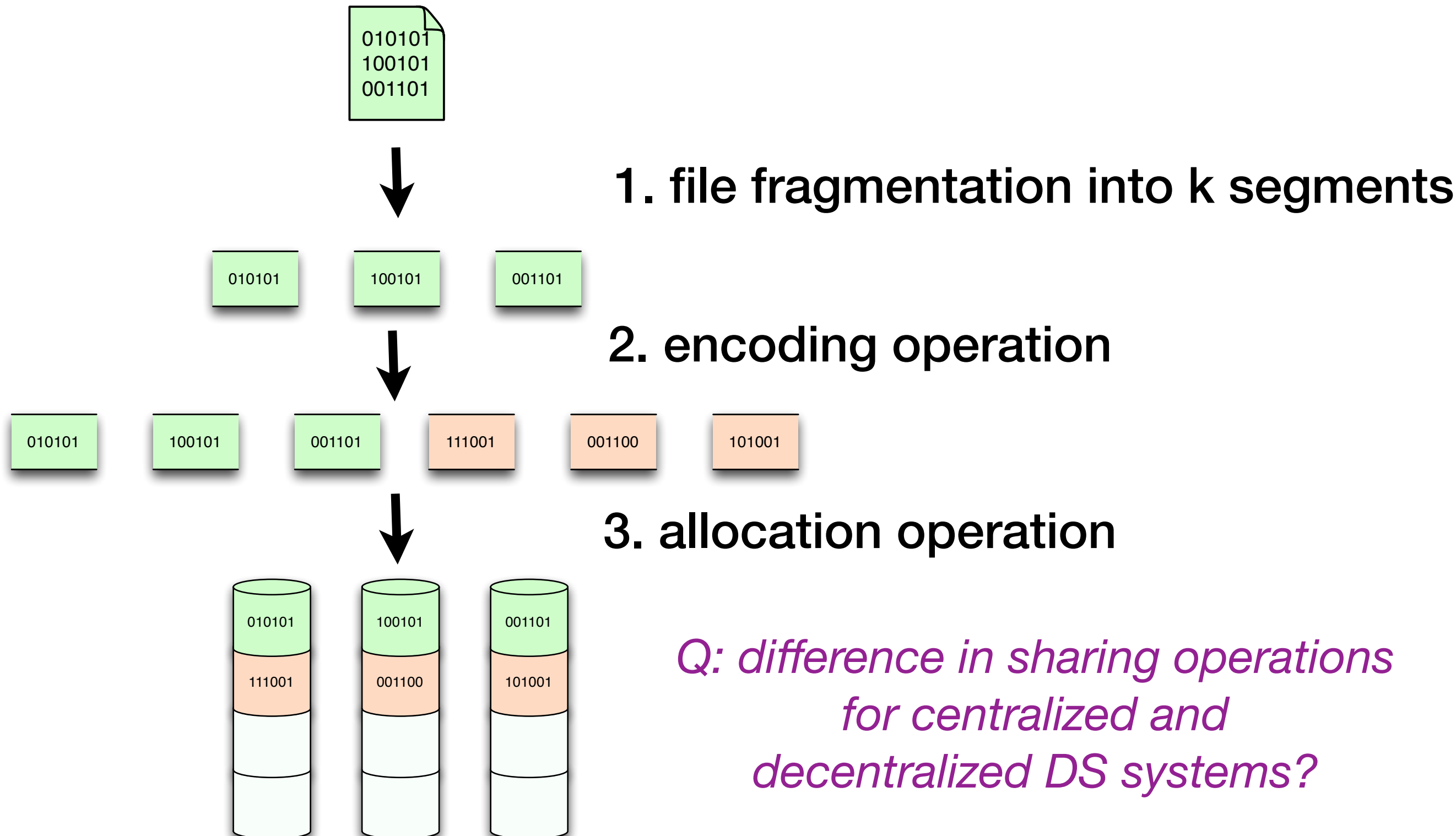


# Data Upload to the DS Network (DSN)

---

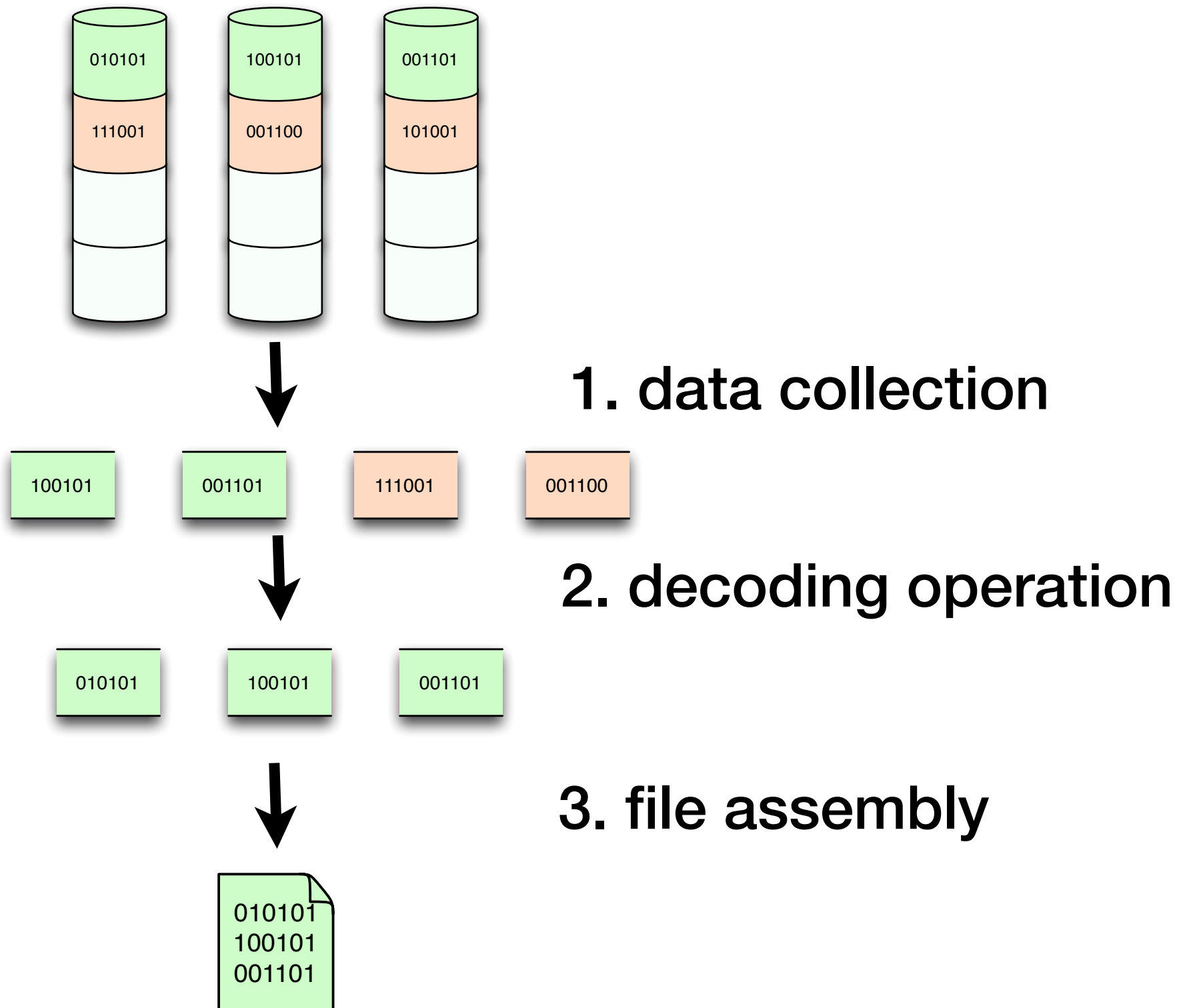


# Data Upload to the DS Network (DSN)



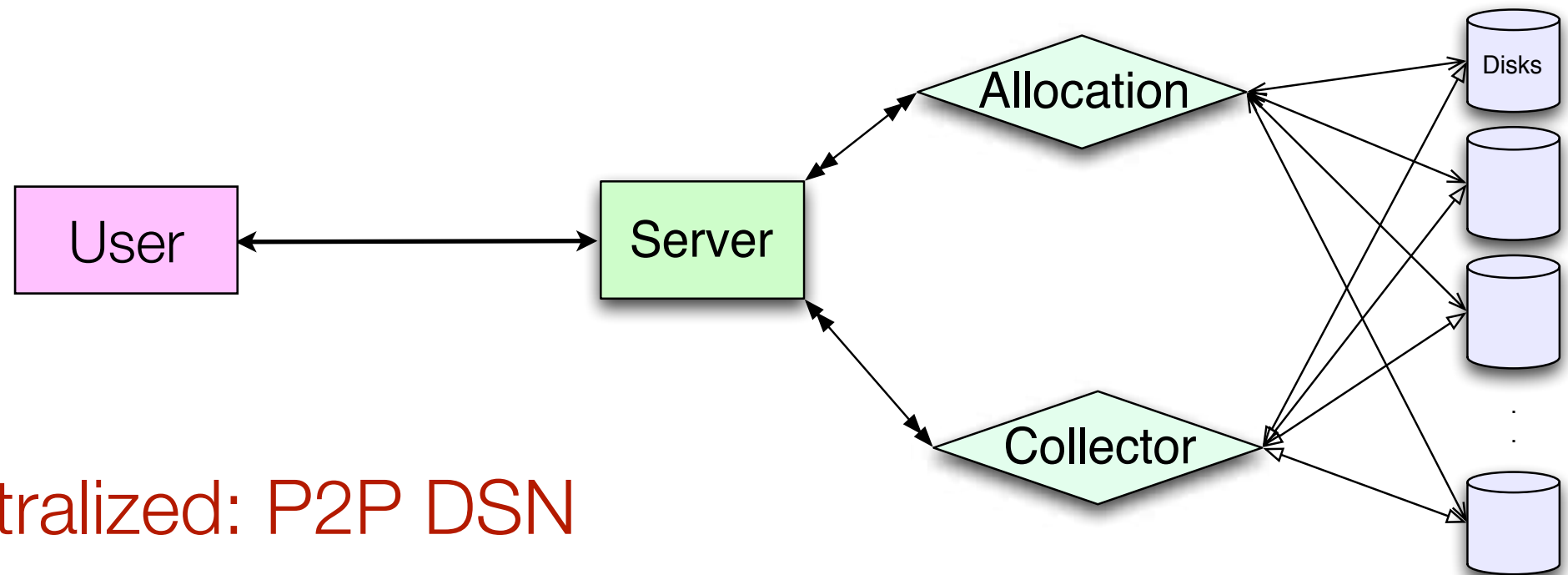
# Data Download from the DSN

---

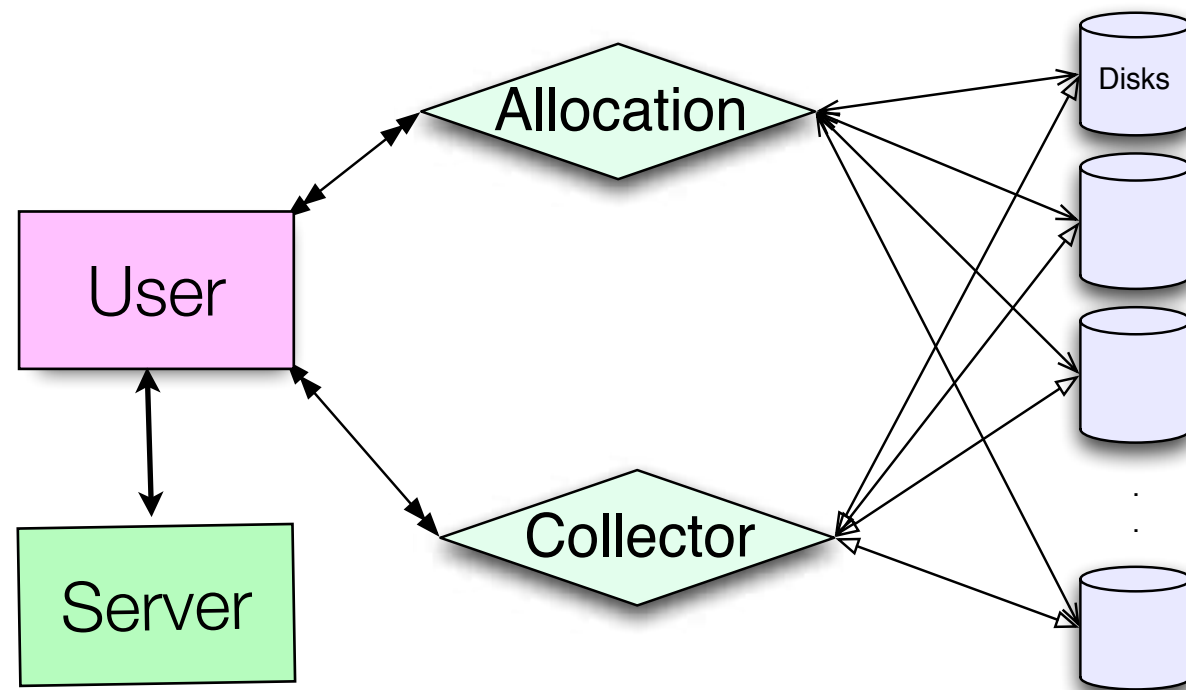


# Upload/Download

centralized: Data Center Network (DCN)



decentralized: P2P DSN



# Upload/Download in DSNs: Access Models

---

Type of allocation/collection:  
deterministic/deterministic

random/deterministic

deterministic/random

random/random

?

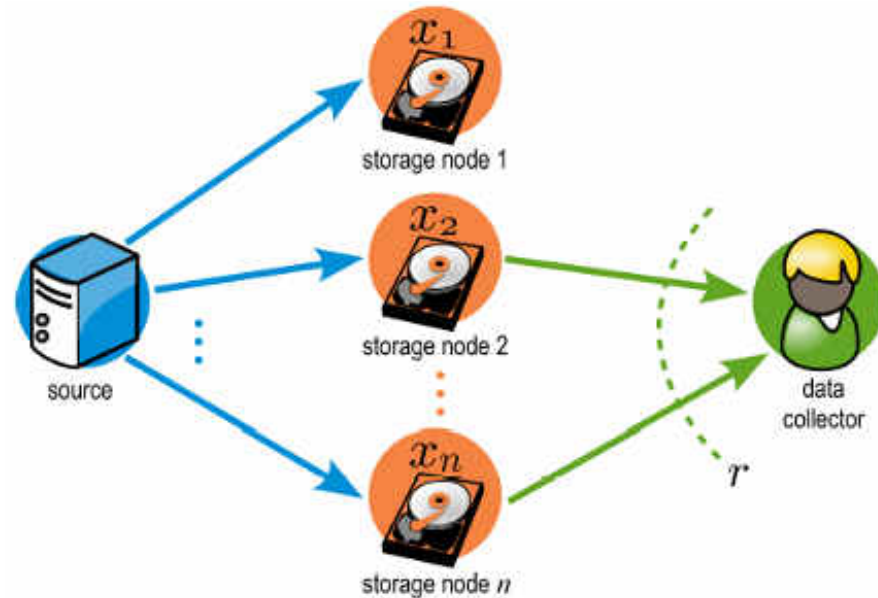
centralized DSNs  
(DCNs)

decentralized DSNs

# Upload/Download in DSNs: Access Models

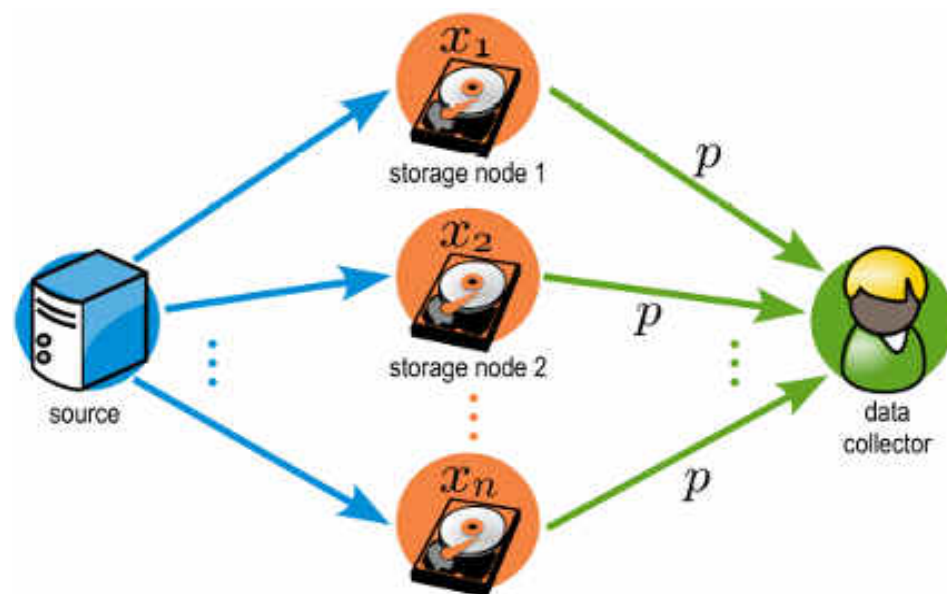
Type of allocation/collection:

deterministic/deterministic



centralized DSNs  
(DCNs)

deterministic/random

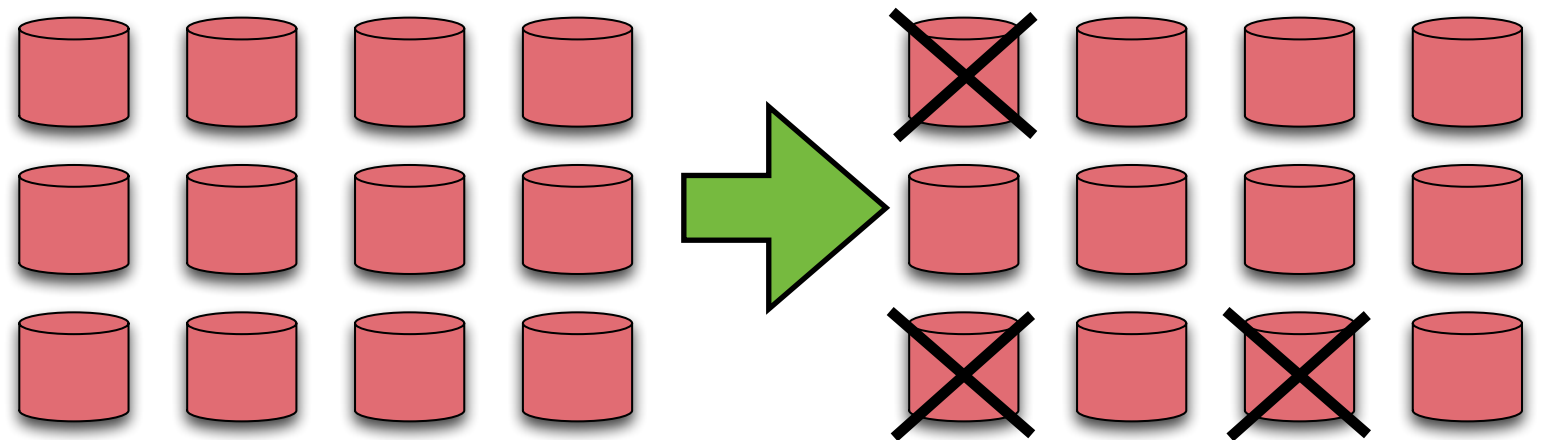


decentralized DSNs



# Data Loss Models, Failure Repair

---



- independent failure model:

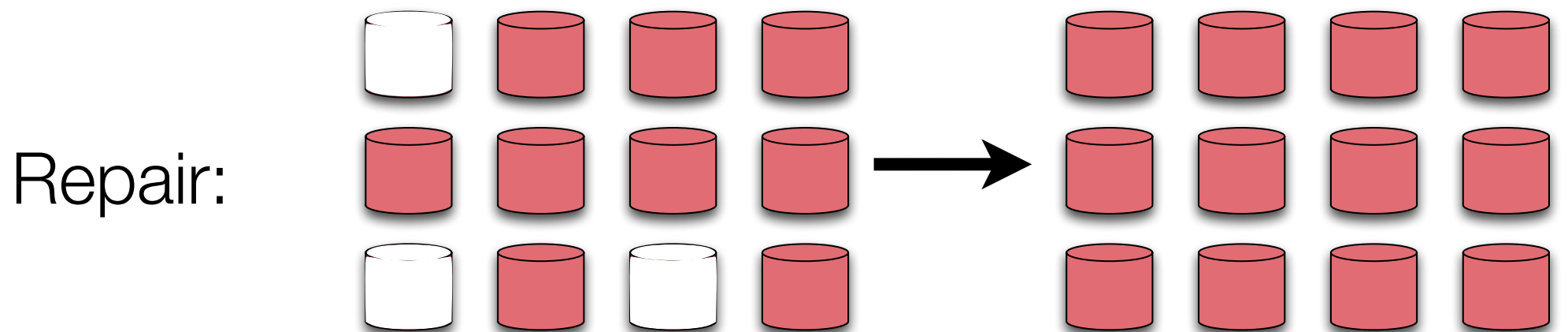
a storage node fails with some probability  $p$  (DCNs) or  
a storage node is not accessible with some probability  $p$  (P2P)

- dependent failure model:

a storage node  $i$ , given a set of failed nodes  $S$ , fails with  
probability  $p(i|S)$

# Data Loss Models, Failure Repair

---

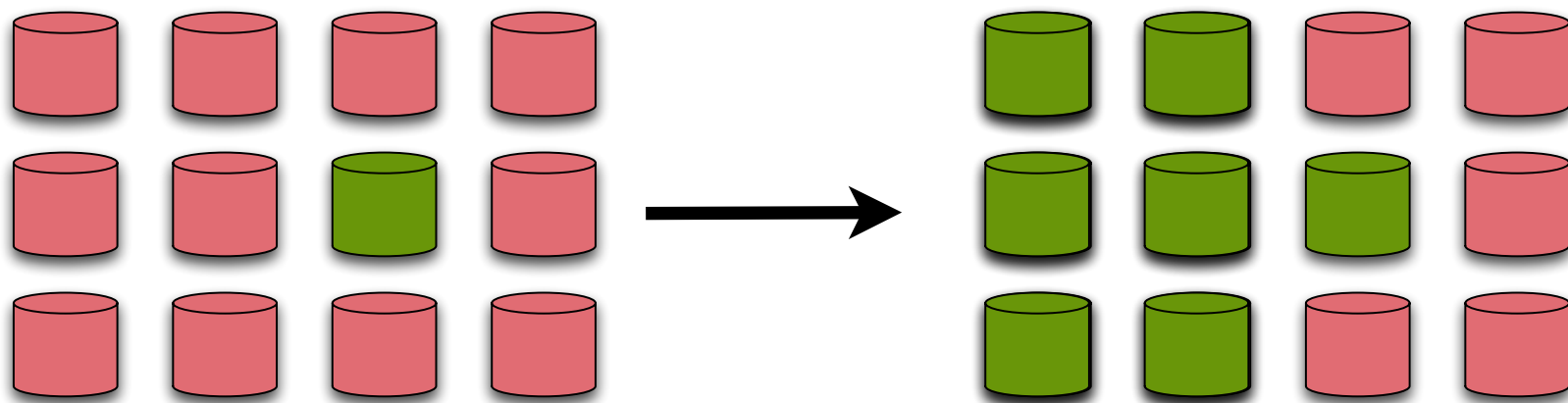


**(local) decoding/ linear combination of available data segments (code symbols)**

- exact repair:  
recovered data segment = lost one
- functional repair:  
recovered data segment  $\neq$  lost one

# Data Update Model

---



**re-calculation of data segments (code symbols)  
depending on the changed data**

- Useful for:  
frequently modified data

# Parameters of DSNs

---

- Number of disks  $n$

# Parameters of DSNs

---

- Number of disks  $n$
- Fault tolerance  $f$  = maximum number of failed disks with no data loss/  
that one can always repair

# Parameters of DSNs

---

- Number of disks  $n$
- Fault tolerance  $f$
- Storage efficiency  $R = \text{user data} / \text{total stored data}$



# Parameters of DSNs

---

- Number of disks  $n$
- Fault tolerance  $f$
- Storage efficiency  $R$
  
- Encoding/decoding complexity

# Parameters of DSNs

---

- Number of disks  $n$
- Fault tolerance  $f$
- Storage efficiency  $R$
- Encoding/decoding complexity
- Repair access  $R(t)$  / bandwidth  $B_R(t)$  = #disks/ amount of data to access/download in order to repair  $t$  failures

# Parameters of DSNs

---

- Number of disks  $n$
- Fault tolerance  $f$
- Storage efficiency  $R$
  
- Encoding/decoding complexity
- Repair access  $R(t)$  / bandwidth  $B_R(t)$
- Download access  $D(t)$  / bandwidth  $B_D(t) = \text{\#disks} / \text{amount of data to access/download in order to decode the data}$

# Parameters of DSNs

---

- Number of disks  $n$
- Fault tolerance  $f$
- Storage efficiency  $R$
  
- Encoding/decoding complexity
- Repair access  $R(t)$  / bandwidth  $B_R(t)$
- Download access  $D(t)$  / bandwidth  $B_D(t)$
- Update access

# Parameters of DSNs

---

(Fundamental *storage* tradeoff, code parameters)

- Number of disks  $n$
- Fault tolerance  $f$
- Storage efficiency  $R$

(code-allocation parameters)

- Encoding/decoding complexity
- Repair access  $R(t)$  / bandwidth  $B_R(t)$
- Download access  $D(t)$  / bandwidth  $B_D(t)$
- Update access

# On Code-Allocation Dependence

---

Code and allocation are dependent in general.

Code : storage and network parameters of the scheme

Allocation : “channel model” of the “storage communication”



# On Code-Allocation Dependence

---

Code and allocation are dependent in general.

Code : storage and network parameters of the scheme

Allocation : “channel model” of the “storage communication”

To simplify the design task,

- code can be made independent of allocation
- allocation can be made independent of code (ex. RAID)

# On Code-Allocation Dependence

---

Code and allocation are dependent in general.

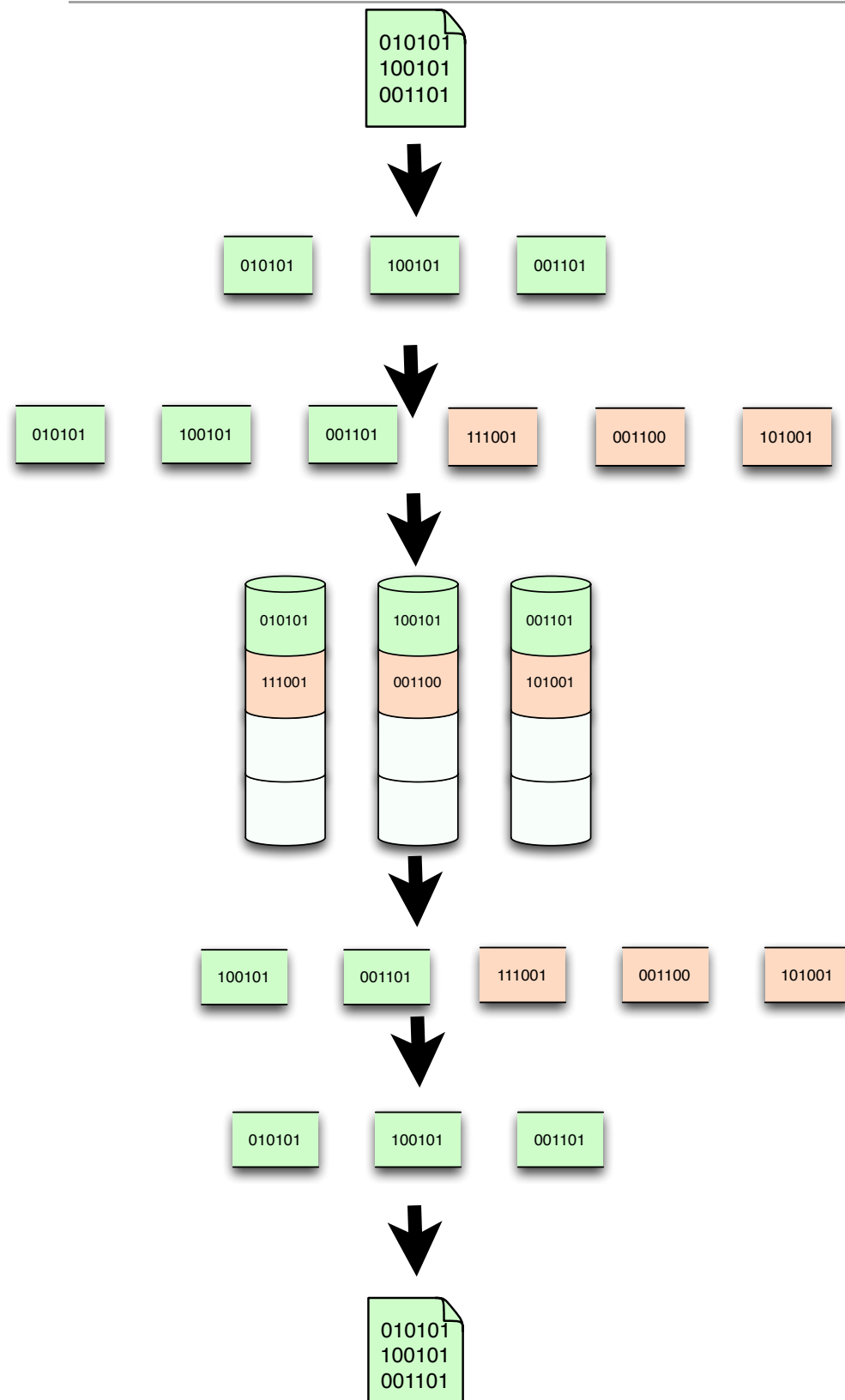
Code : storage and network parameters of the scheme

Allocation : “channel model” of the “storage communication”

To simplify the design task,

- code can be made independent of allocation
  - ~~allocation can be made independent of code~~ (ex. RAID)
- in a real DSN, allocation is always dependent on code (#users)

# Our Toy Data Storage Model for the Next Part



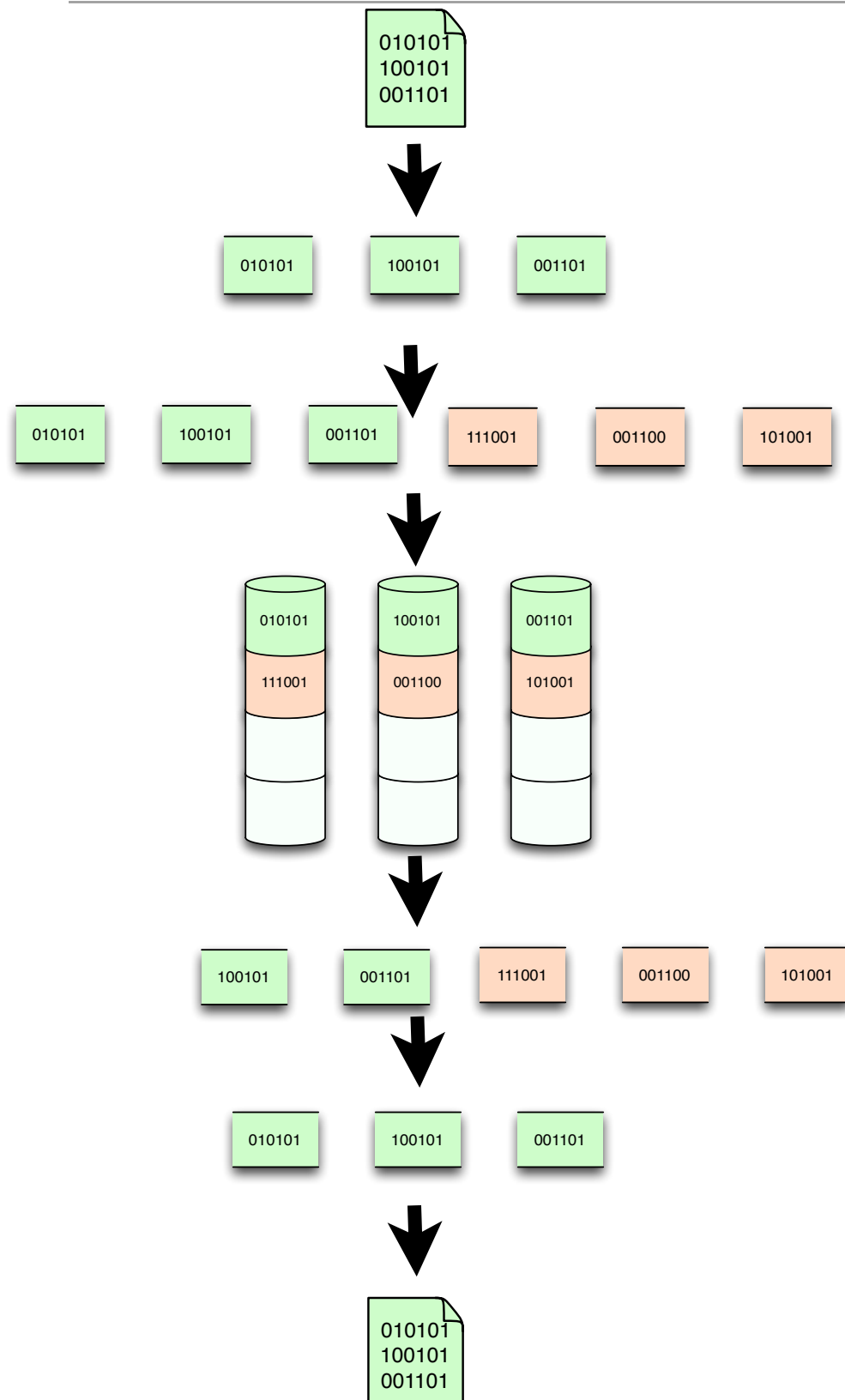
# segments =  $k$

# encoded segments = # disks =  $n$

1 segment  $\rightarrow$  1 separate disk

independent loss model, failure probability =  $p$

# Our Toy Data Storage Model for the Next Part



# segments =  $k$

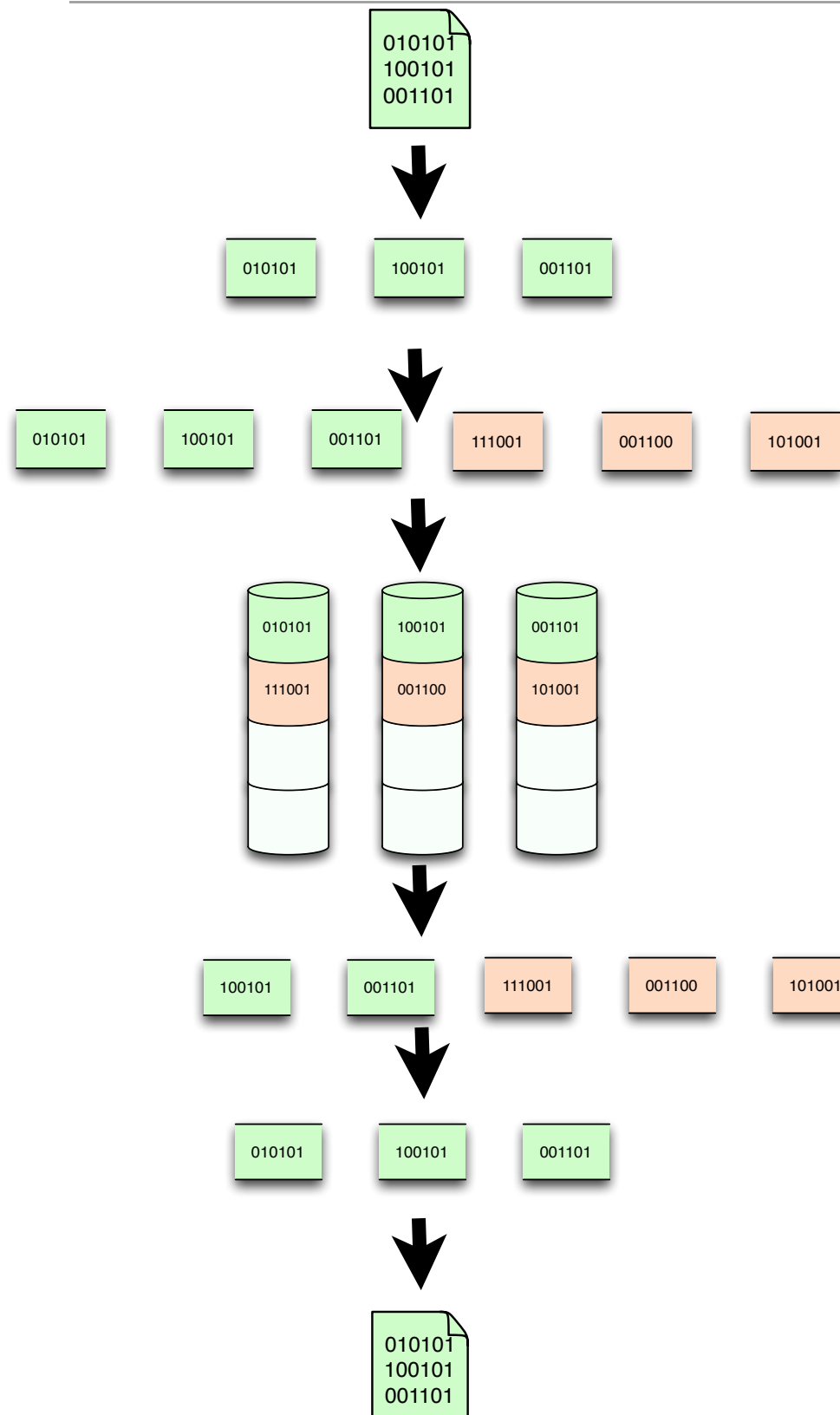
# encoded segments = # disks =  $n$

1 segment  $\rightarrow$  1 separate disk

independent loss model, failure probability =  $p$

(i) segment of size  $m$  = symbol over  $GF(2^m)$

# Our Toy Data Storage Model for the Next Part




# segments =  $k$

# encoded segments = # disks =  $n$

1 segment  $\rightarrow$  1 separate disk

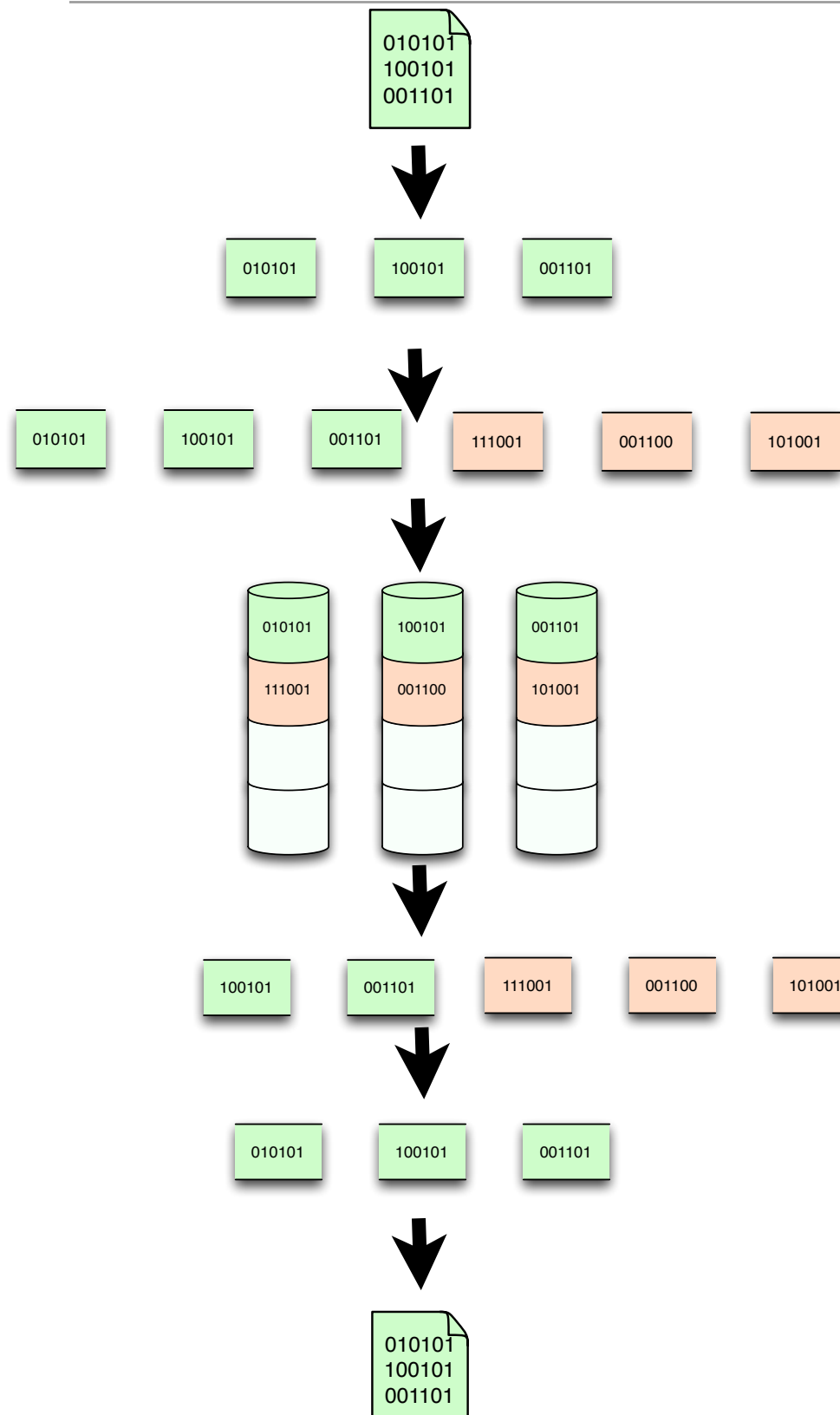
independent loss model, failure probability =  $p$

(i) segment of size  $m$  = symbol over  $GF(2^m)$

(ii)  gives us the channel model, which is symbol erasure channel (binary erasure channel) with probability  $p$

## Part 2: Codes for DSNs and Storage Tradeoffs

# Our Toy Data Storage Model for the Next Part




# segments =  $k$

# encoded segments = # disks =  $n$

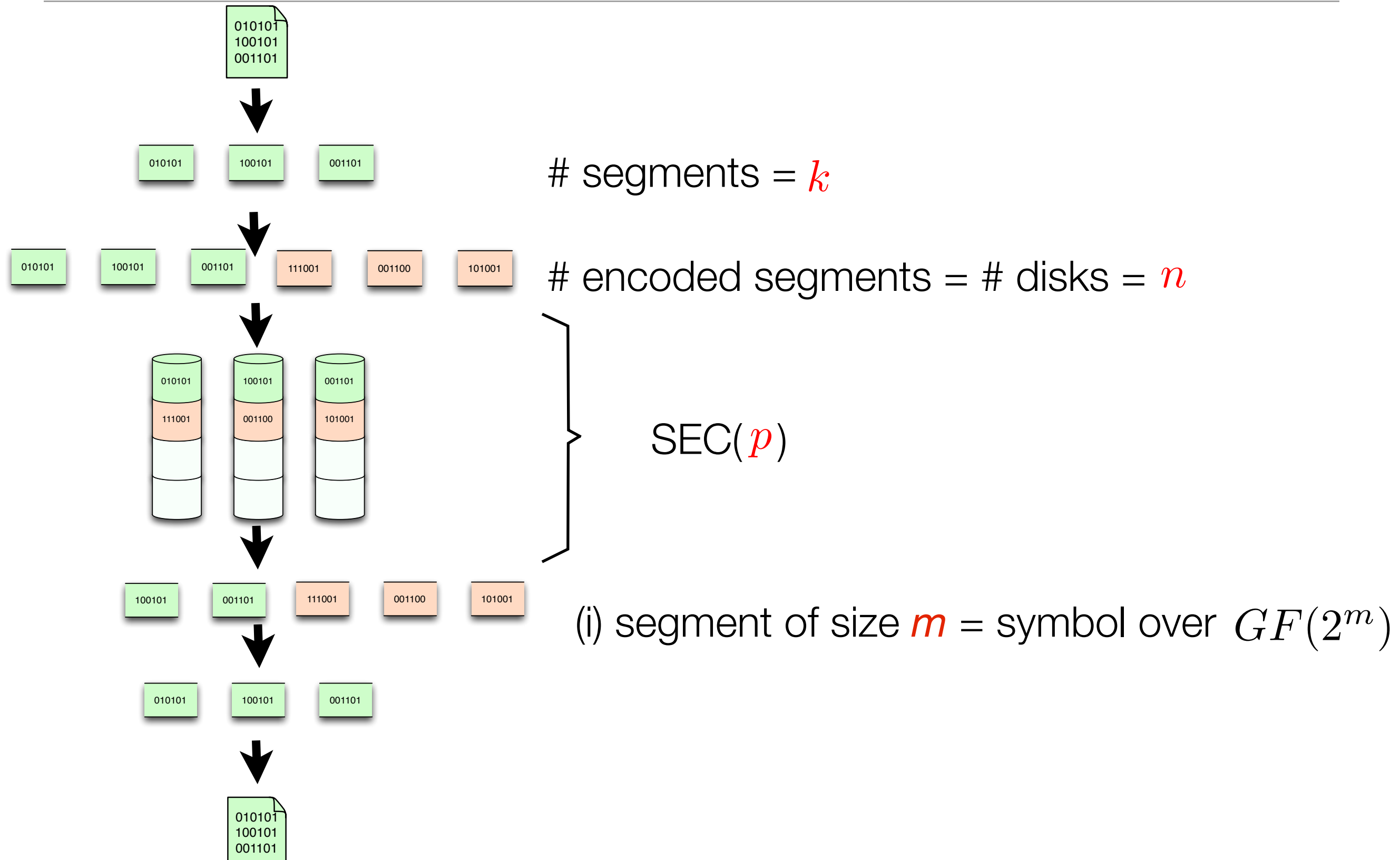
1 segment  $\rightarrow$  1 separate disk

independent loss model, failure probability =  $p$

(i) segment of size  $m$  = symbol over  $GF(2^m)$

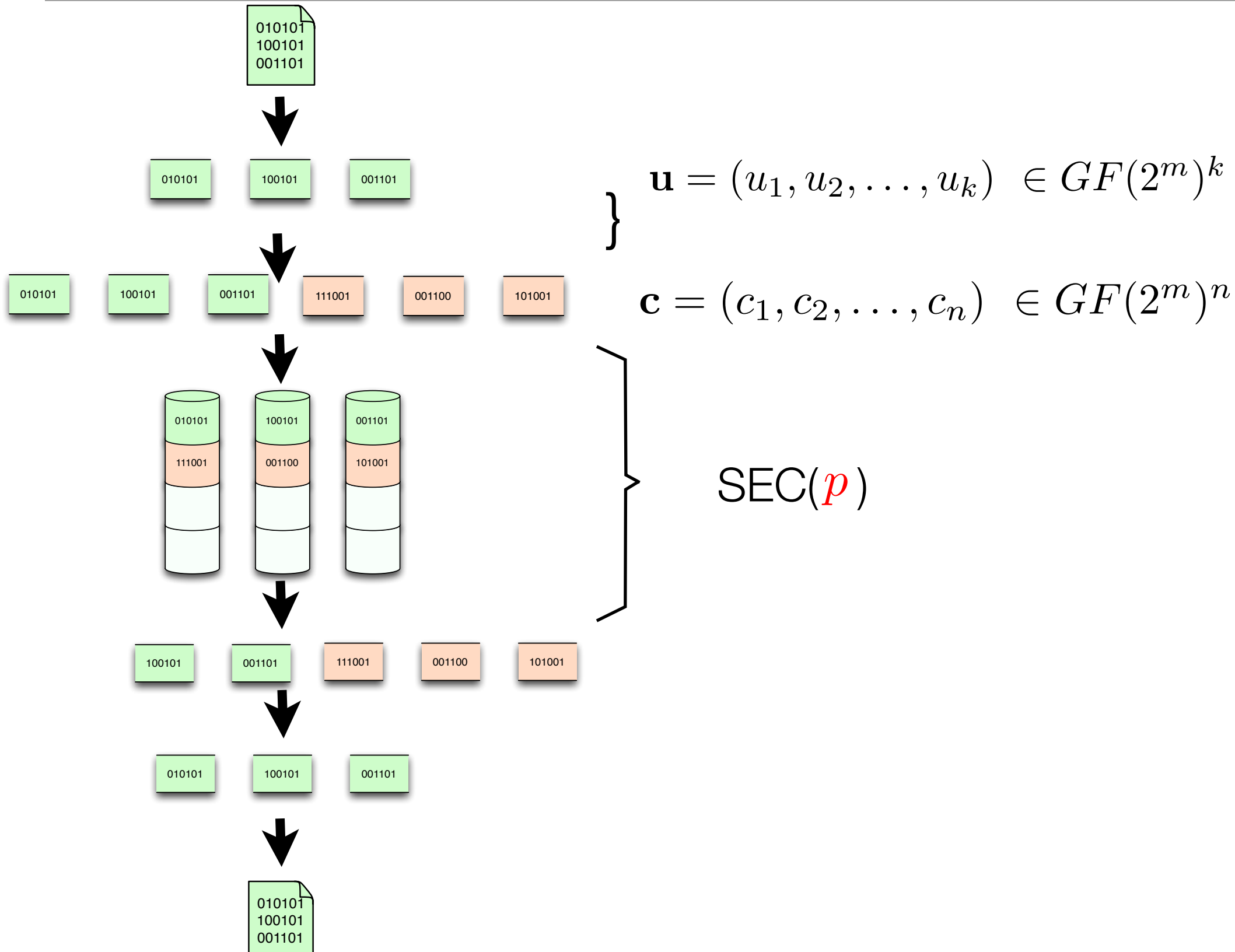
(ii)  gives us the channel model, which is symbol erasure channel (binary erasure channel) with probability  $p$

# Our Toy Data Storage Model for the Next Part

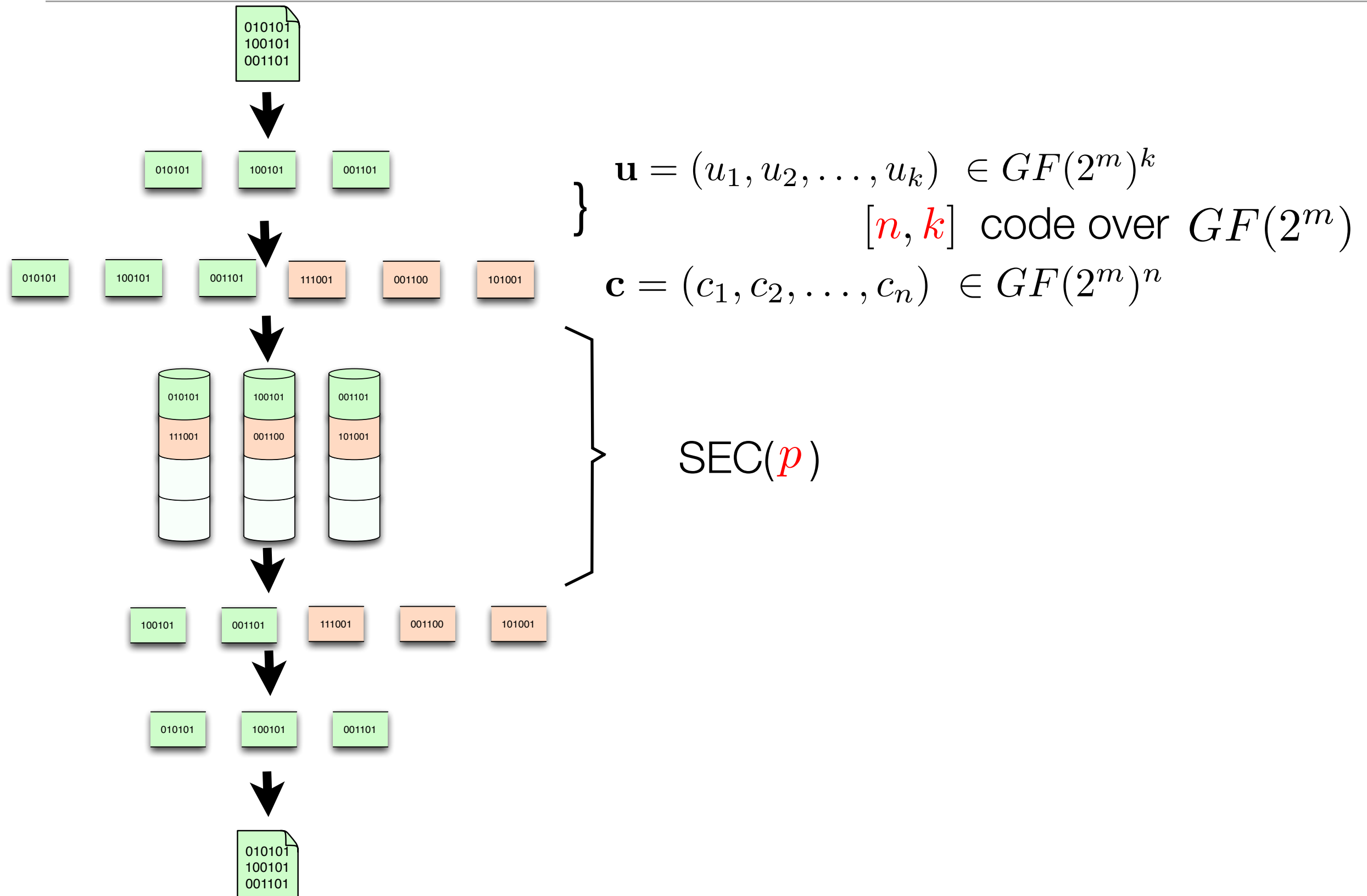




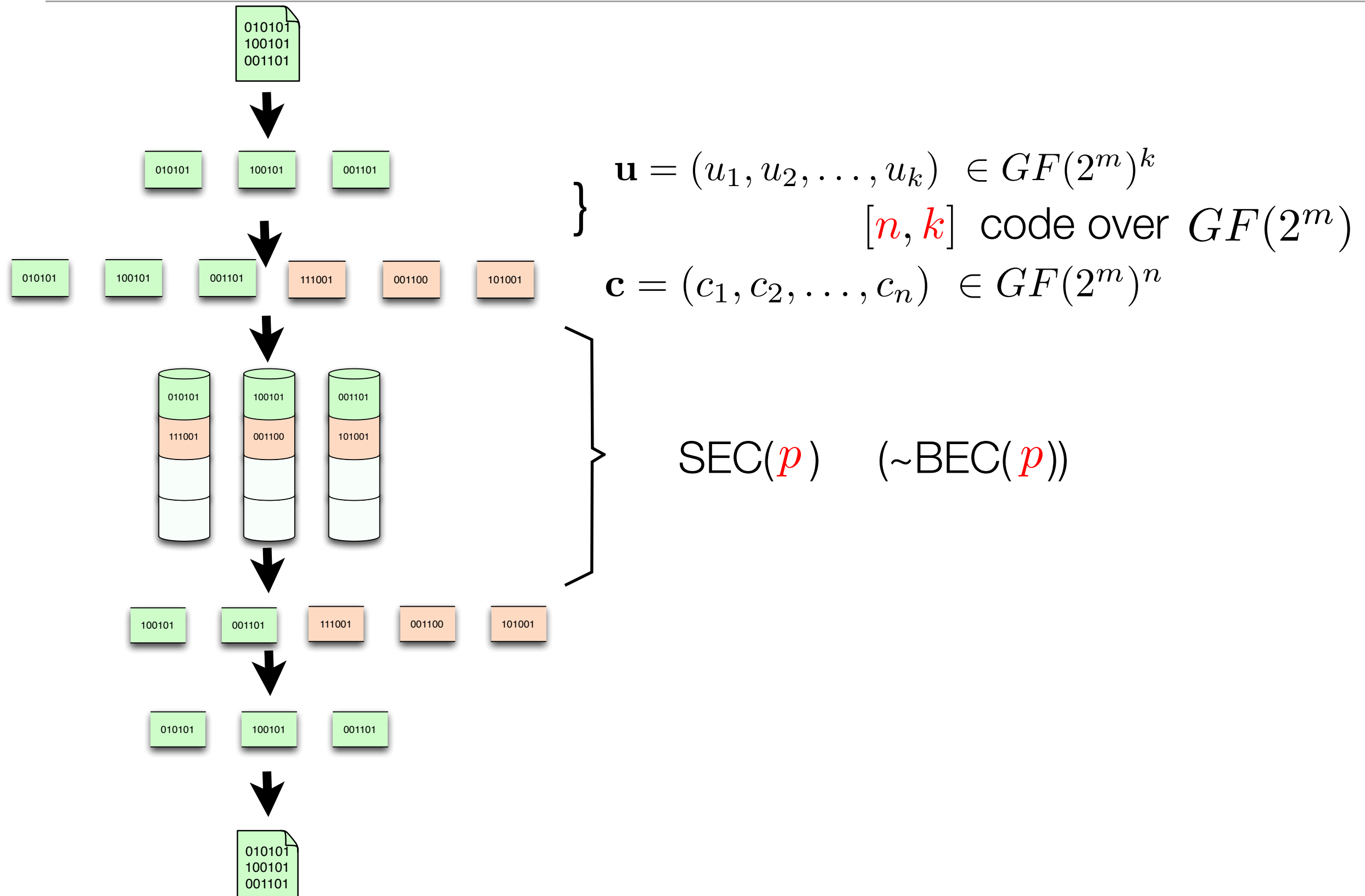
# Our Toy Data Storage Model for the Next Part



# Our Toy Data Storage Model for the Next Part



# Our Toy Data Storage Model for the Next Part



# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## *Definition*

*Hamming minimum distance  $d$  = minimum number of differences between any two codewords from  $C$  (there are  $2^k$  in total).*

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## *Definition*

*Hamming minimum distance  $d$  = minimum number of differences between any two codewords from  $C$  (there are  $2^k$  in total).*

## *Definition*

*Erasure-correcting capability  $t$  = maximum number of erased symbols that can always be corrected by decoding*

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## *Definition*

*Hamming minimum distance  $d$  = minimum number of differences between any two codewords from  $C$  (there are  $2^k$  in total).*

## *Definition*

*Erasure-correcting capability  $t$  = maximum number of erased symbols that can always be corrected by decoding*

*Simple important fact*

$$t = d - 1$$

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## Definition

*Hamming minimum distance  $d$  = minimum number of differences between any two codewords from  $C$  (there are  $2^k$  in total).*

## Definition

*Erasure-correcting capability  $t$  = maximum number of erased symbols that can always be corrected by decoding*

*Simple important fact*

$$t = d - 1$$

## Definition

Code rate:  $R_c = \frac{k}{n}$   $(0 \leq R_c \leq 1)$

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## *Definition*

*Generator matrix  $G = k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .*



# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## *Definition*

*Generator matrix  $G = k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .*

*Encoding operation:*

$$\mathbf{c} = \mathbf{u}G$$

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## Definition

**Generator matrix**  $G = k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .

Encoding operation:

$$\mathbf{c} = \mathbf{u}G$$

## Definition

**Systematic code** = code for which the generator matrix is of form  $G = (I_k \ P)$

$$\mathbf{c} = \mathbf{u} (I_k \ P) = (u_1, u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## Definition

**Generator matrix  $G$**  =  $k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .

Encoding operation:

$$\mathbf{c} = \mathbf{u}G$$

## Definition

**Systematic code** = code for which the generator matrix is of form  $G = (I_k \ P)$

$$\mathbf{c} = \mathbf{u} (I_k \ P) = (u_1, u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

## Definition

**Parity matrix  $H$**  =  $(n-k) \times n$  matrix such that  $H\mathbf{c}^T = \mathbf{0}$

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## Definition

**Generator matrix**  $G$  =  $k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .

Encoding operation:

$$\mathbf{c} = \mathbf{u}G$$

## Definition

**Systematic code** = code for which the generator matrix is of form  $G = (I_k \ P)$

$$\mathbf{c} = \mathbf{u} (I_k \ P) = (u_1, u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

## Definition

**Parity matrix**  $H$  =  $(n-k) \times n$  matrix such that  $H\mathbf{c}^T = \mathbf{0}$

Q: What is the form of  $H$  for the systematic code?

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## Definition

Generator matrix  $G = k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .

Encoding operation:

$$\mathbf{c} = \mathbf{u}G$$

## Definition

Systematic code = code for which the generator matrix is of form  $G = (I_k \ P)$

$$\mathbf{c} = \mathbf{u} (I_k \ P) = (u_1, u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

## Definition

Parity matrix  $H = (n-k) \times n$  matrix such that  $H\mathbf{c}^T = \mathbf{0}$

Q: What is the form of  $H$  for the systematic code?  $H = (-P^T \ I_{n-k})$

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## Definition

Generator matrix  $G = k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .

Encoding operation:

$$\mathbf{c} = \mathbf{u}G$$

## Definition

Systematic code = code for which the generator matrix is of form  $G = (I_k \ P)$

$$\mathbf{c} = \mathbf{u} (I_k \ P) = (u_1, u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

## Definition

Parity matrix  $H = (n-k) \times n$  matrix such that  $H\mathbf{c}^T = \mathbf{0}$

Q: What is the form of  $H$  for the systematic code?  $H = (-P^T \ I_{n-k})$

Q: What is the decoding operation to recover erased symbols?

# Some Coding Facts

---

For a  $[n, k]$  code  $C$  over  $GF(2^m)$ :

## Definition

Generator matrix  $G = k \times n$  matrix, the lines of which are base vectors, spanning the vector subspace of  $C$ .

Encoding operation:

$$\mathbf{c} = \mathbf{u}G$$

## Definition

Systematic code = code for which the generator matrix is of form  $G = (I_k \ P)$

$$\mathbf{c} = \mathbf{u} (I_k \ P) = (u_1, u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

## Definition

Parity matrix  $H = (n-k) \times n$  matrix such that  $H\mathbf{c}^T = \mathbf{0}$

Q: What is the form of  $H$  for the systematic code?  $H = (-P^T \ I_{n-k})$

Q: What is the decoding operation to recover erased symbols?

$$H_{erased} (\mathbf{c}_{erased})^T = H_{known} (\mathbf{c}_{known})^T \quad \text{or} \quad \mathbf{u} = (G_{known})^{-1} \mathbf{c}_{known}$$

# Link Between DSN and Code Parameters

---

| DSN                          | Code  |
|------------------------------|---|
| Fault tolerance $f$          | Erasure-correcting capability $t = d - 1$                     |
| Storage efficiency $R$       | Code rate $R_c$   |
| Repair access $R(1)$         | $\frac{1}{n - k} \sum_{i=1}^{n-k} (\text{weight}_H(h_i) - 1)$ |
| Download access $D(k)$       | $\min  c_{known} $ s. t. the solution exists                  |
| Encoding/decoding complexity | Encoding/decoding complexity                                  |



# Link Between DSN and Code Parameters

| DSN                          | Code  |
|------------------------------|---|
| Fault tolerance $f$          | Erasure-correcting capability $t = d - 1$                   |
| Storage efficiency $R$       | Code rate $R_c$   |
| Repair access $R(1)$         | $\frac{1}{n-k} \sum_{i=1}^{n-k} (\text{weight}_H(h_i) - 1)$ |
| Download access $D(k)$       | $\min  c_{known} $ s. t. the solution exists                |
| Encoding/decoding complexity | Encoding/decoding complexity                                |

Is it possible?  $f(d) \uparrow$   $R(R_c) \uparrow$   $R(1) \downarrow$   $D(k) \downarrow$  complexity  $\downarrow$

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_C) + 1$$

## *Definition*

*Maximum distance separable (MDS) code* = code that achieves the Singleton bound.

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## *Definition*

*Maximum distance separable (MDS) code = code that achieves the Singleton bound.*

*Q: What is  $D(k)$  for an MDS code?*

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## *Definition*

*Maximum distance separable (MDS) code = code that achieves the Singleton bound.*

*Q: What is  $D(k)$  for an MDS code?  $D(k)=k$ .*

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## *Definition*

*Maximum distance separable (MDS) code* = code that achieves the Singleton bound.

*Q: What is  $D(k)$  for an MDS code?  $D(k)=k$ .*

*Q: What is the code having the lowest decoding complexity?*

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## *Definition*

*Maximum distance separable (MDS) code* = code that achieves the Singleton bound.

*Q: What is  $D(k)$  for an MDS code?  $D(k)=k$ .*

*Q: What is the code having the lowest decoding complexity? A systematic one.*

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## *Definition*

*Maximum distance separable (MDS) code = code that achieves the Singleton bound.*

*Q: What is  $D(k)$  for an MDS code?  $D(k)=k$ .*

*Q: What is the code having the lowest decoding complexity? A systematic one.*

Examples of used systematic MDS codes:

## **Repetition code (RAID1)**

$$(c_1, c_2, \dots, c_n) = (u_1, u_1, \dots, u_1)$$

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## Definition

*Maximum distance separable (MDS) code* = code that achieves the Singleton bound.

*Q: What is  $D(k)$  for an MDS code?  $D(k)=k$ .*

*Q: What is the code having the lowest decoding complexity? A systematic one.*

Examples of used systematic MDS codes:

## Repetition code (RAID1)

$$(c_1, c_2, \dots, c_n) = (u_1, u_1, \dots, u_1)$$

$$R = \frac{1}{n}, \quad d = n$$



# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## Definition

*Maximum distance separable (MDS) code* = code that achieves the Singleton bound.

*Q: What is  $D(k)$  for an MDS code?  $D(k)=k$ .*

*Q: What is the code having the lowest decoding complexity? A systematic one.*

Examples of used systematic MDS codes:

## Repetition code (RAID1)

$$(c_1, c_2, \dots, c_n) = (u_1, u_1, \dots, u_1)$$

$$R = \frac{1}{n}, \quad d = n$$

## Parity [3,2] code (RAID5)

$$(c_1, c_2, \dots, c_n) = (u_1, u_2, \dots, u_{n-2}, \bigoplus_{i=1}^{n-1} u_i)$$

$$R = \frac{n-1}{n}, \quad d = 2$$

# A Storage Tradeoff: $f$ , $R$ and $D(k)$

---

Singleton bound:

$$d \leq n - k + 1 = n(1 - R_c) + 1$$

## Definition

*Maximum distance separable (MDS) code* = code that achieves the Singleton bound.

*Q: What is  $D(k)$  for an MDS code?  $D(k)=k$ .*

*Q: What is the code having the lowest decoding complexity? A systematic one.*

Examples of used systematic MDS codes:

**Reed-Solomon (RS) codes (RAID6)**

$$\mathbf{c} = \mathbf{u} \begin{pmatrix} I_k & P \end{pmatrix} = (u_1, u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

All elements of  $P$  are non-zero.  $n \leq$  field size.

# Distributed Storage with MDS Codes

---

- \* Achieving the tradeoff between the **fault tolerance**, **storage efficiency** and **download access**
- \* **Decoding complexity**  $\leq O(k^3)$ . For systematic codes and deterministic collection - 0.

But:

- Encoding complexity:  $O(n^2)$
- Repair access:  $R(1) = k \sim O(k)$
- Maximum codelength  $n$  cannot be high

# Distributed Storage with MDS Codes

---

- \* Achieving the tradeoff between the **fault tolerance**, **storage efficiency** and **download access**
- \* **Decoding complexity**  $\leq O(k^3)$ . For systematic codes and deterministic collection - 0.

But:

- Encoding complexity:  $O(n^2)$
- Repair access:  $R(1) = k \sim O(k)$
- Maximum codelength  $n$  cannot be high

*High repair cost - inevitable price for erasure coding?*

# Distributed Storage with Non-MDS Codes

---

## Code constructions based on modifications of MDS codes:

- pyramid codes, generalized pyramid codes
- local reconstruction codes

## Other code constructions:

- regenerating codes (based on network coding)
- LDPC codes (graph codes)

# Modifications of MDS Codes: Pyramid Codes

---

**Remark:** for the repetition code,  $R(1)=1$ .

**Idea:** construct “pyramids”

$$H = \begin{matrix} & & & & & & P_1 & P_2 & P_3 \\ \left( \begin{array}{ccccccccc} p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & 1 & 0 & 0 \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} & 0 & 1 & 0 \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} & 0 & 0 & 1 \end{array} \right) \end{matrix}$$

$$R(1) = 6$$

# Modifications of MDS Codes: Pyramid Codes

**Remark:** for the repetition code,  $R(1)=1$ .

**Idea:** construct “pyramids” (“check-node splitting”)

$$H = \begin{pmatrix} p_{11} & p_{12} & p_{13} & 0 & 0 & 0 & \underbrace{1 \ 1}_{P_1} & 0 & 0 \\ 0 & 0 & 0 & p_{14} & p_{15} & p_{16} & 0 & 1 & 0 & 0 \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} & 0 & 0 & 1 & 0 \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R(1) = 4 \cdot \frac{3}{9} + 3 \cdot \frac{4}{9} + 6 \cdot \frac{2}{9} = 4$$

# Modifications of MDS Codes: Pyramid Codes

---

**Remark:** for the repetition code,  $R(1)=1$ .

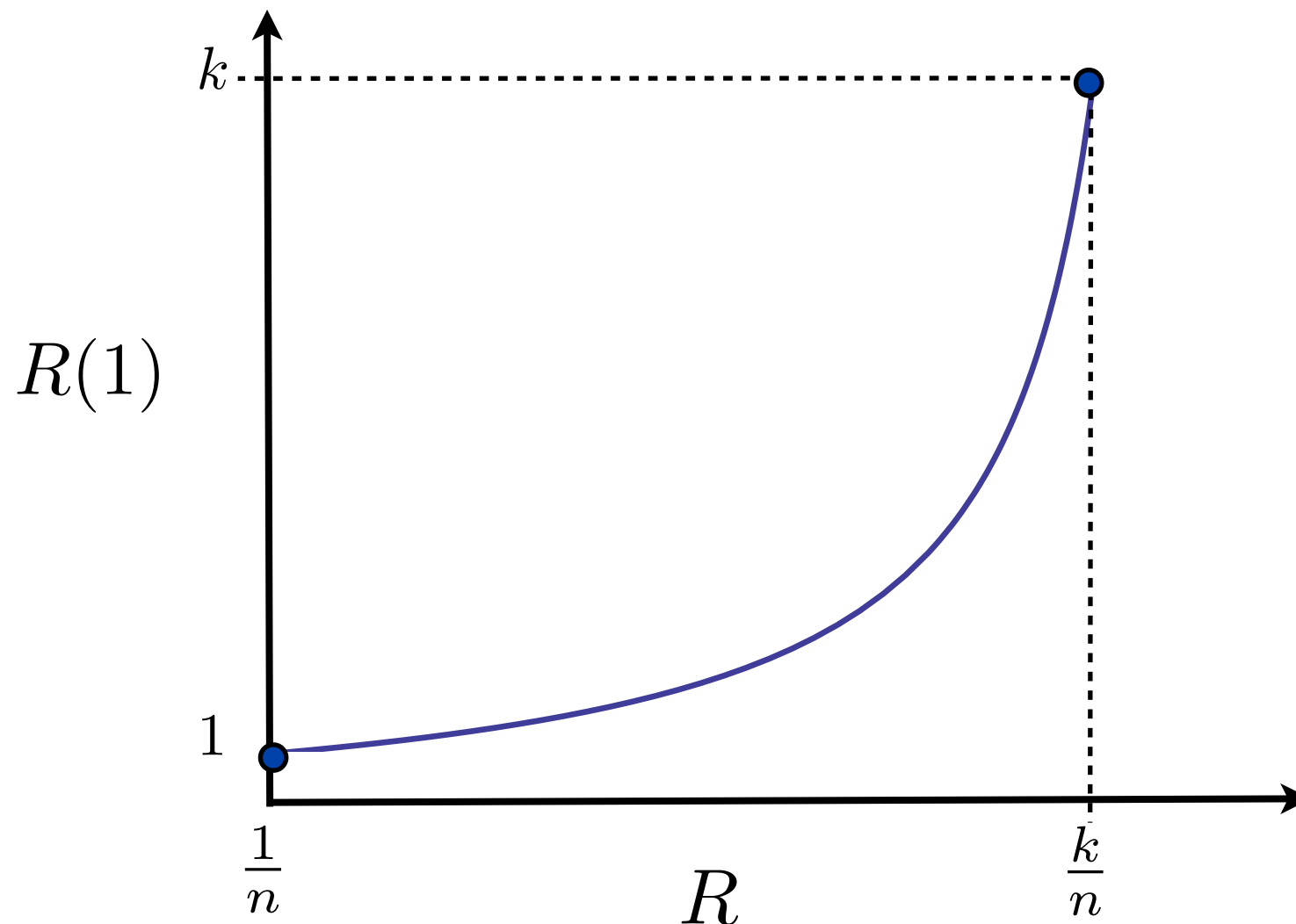
**Idea:** construct “pyramids” (“check-node splitting”, applied recursively within subgroups until the target  $R(1)$  is achieved)



# Modifications of MDS Codes: Pyramid Codes

**Remark:** for the repetition code,  $R(1)=1$ .

**Idea:** construct “pyramids” (“check-node splitting”, applied recursively within subgroups until the target  $R(1)$  is achieved)



# Modifications of MDS Codes: Pyramid Codes

---

**Remark:** for the repetition code,  $R(1)=1$ .

**Idea:** construct “pyramids” (“check-node splitting”, applied recursively within subgroups until the target  $R(1)$  is achieved)

\* Pyramid codes achieve the tradeoff between the **storage efficiency** and **repair access**, if the repair of information symbols is only taken into account [GHSY12]

**But:**

- Pyramid codes are non-MDS, except for two extreme points ( $R$  is not optimal for given  $n$  and  $d$ )
- decoding access  $\uparrow$
- encoding/decoding complexity  $\uparrow$

# Modifications of MDS Codes: Pyramid Codes

---

**Remark:** for the repetition code,  $R(1)=1$ .

**Idea:** construct “pyramids” (“check-node splitting”, applied recursively within subgroups until the target  $R(1)$  is achieved)

- \* Pyramid codes achieve the tradeoff between the **storage efficiency** and **repair access**, if the repair of information symbols is only taken into account [GHSY12]
- \* Close to the concept of **locally decodable codes** [Yek]
- \* **Generalized pyramid codes** are based on “overlapped pyramids”, where the overlapping should be done to satisfy the **recoverability condition** (guarantee to recover up to some number of erasures)

[Yek] S. Yekhanin. *Locally decodable codes*.

[GHSY12] P. Gopalan, Cheng Huang, H. Simitci, and S. Yekhanin. On the locality of codeword symbols. *Information Theory, IEEE Transactions on*, 58(11):6925–6934, 2012.

# Modifications of MDS Codes: Local Reconstruction Codes

---

Idea: add “local” parity for groups of information symbols

$$H = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & P_1 & P_2 & P_3 \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} & 0 & 1 & 0 \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} & 0 & 0 & 1 \end{pmatrix}$$

$$R(1) = 6$$

# Modifications of MDS Codes: Local Reconstruction Codes

Idea: add “local” parity for groups of information symbols

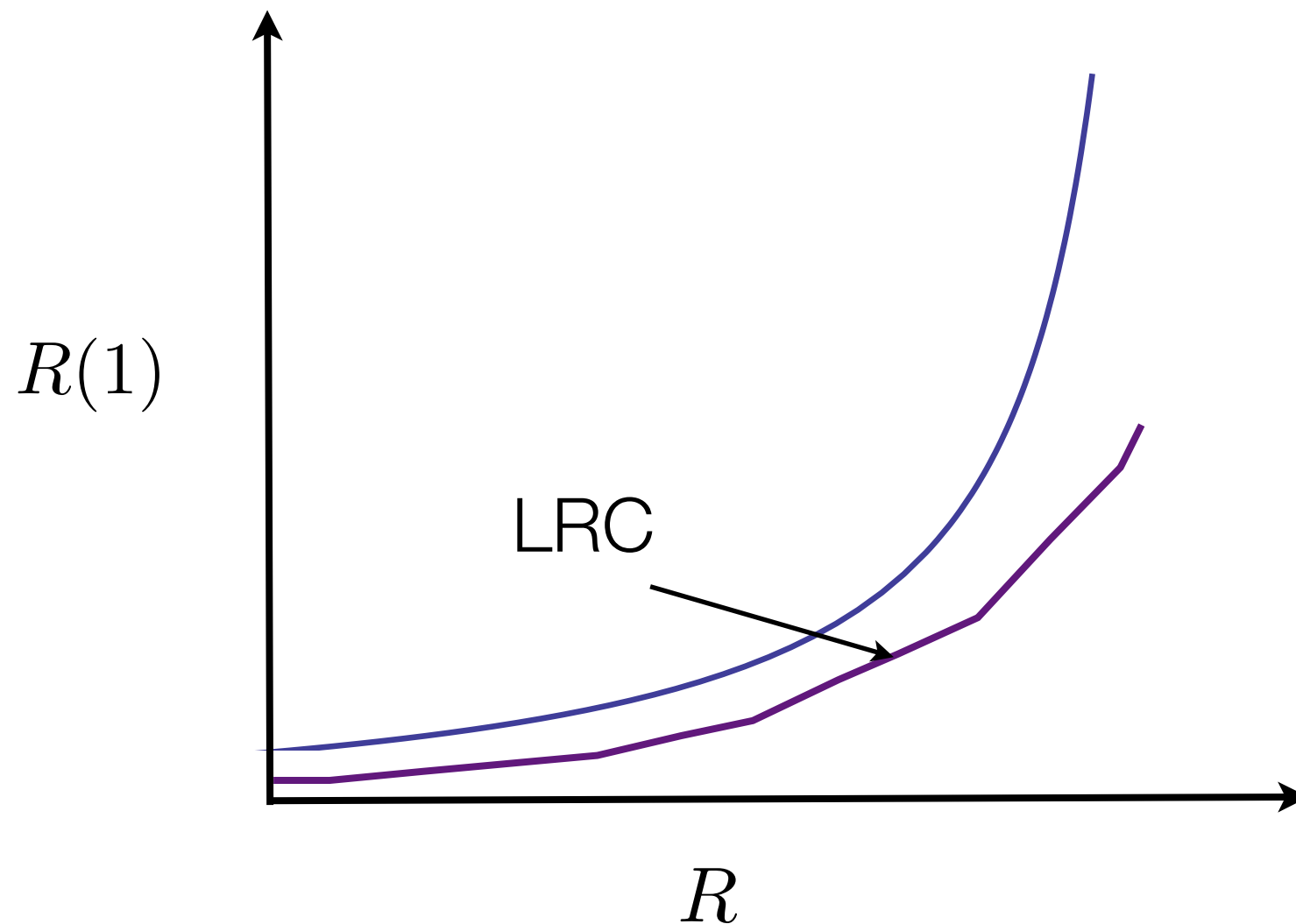
$$H = \begin{pmatrix} q_{11} & q_{12} & q_{13} & 0 & 0 & 0 & P_1 & P_2 & P_3 & P_4 & P_5 \\ 0 & 0 & 0 & q_{14} & q_{15} & q_{16} & 0 & 0 & 0 & 0 & 1 \\ p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & 1 & 0 & 0 & 0 & 0 \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} & 0 & 1 & 0 & 0 & 0 \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$R(1) = 3$$

# Modifications of MDS Codes: Local Reconstruction Codes

---

Particular example [HSX12, Hua13] for Windows Azure:



- [HSX<sup>+</sup>12] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin. Erasure coding in windows azure storage. *USENIX ATC'12*, 2012.
- [Hua13] C. Huang. Erasure coding for storage applications (part ii). presented at the USENIX FAST'13, 2013.

# Modifications of MDS Codes: Local Regeneration Codes

---

\* **LRC codes** are what is the best example by now in terms of  $R(1)$  for information symbols

$R(1) = O(k/t)$  with  $t$  - number of groups

\* Possibility to deal with the **already stored data**

\* **Decoding complexity** does not grow

But:

- LRC codes have low  $R$  ( $R$  is not optimal for given  $n$  and  $d$ )
- upload access ↑
- encoding complexity ↑

# What about Large DCNs?

— Examples of

**MDS codes:**  $[9,6]$  (Google GFS II),  $[14,10]$  (HDFS Facebook)

**LRC codes:**  $[16,12]$ ,  $[18,12]$

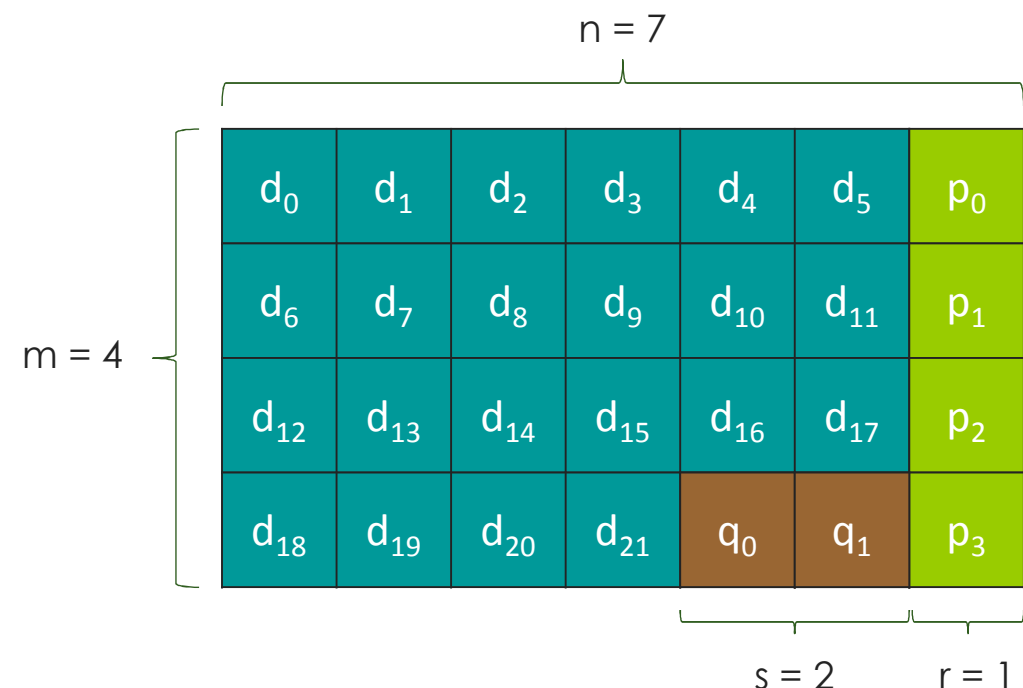
— What about larger  $n$ ?

- large encoding/decoding complexity and
- NP-hard problem of the code design

**Existing solutions:** concatenations of smaller codes (many variants)

**Example:** Partial MDS codes

Product codes: two MDS codes:  
local  $[m+r, m]$  and global  $[n+s, n]$   
(possibility to use local codes with different parameters) [Hua13]





# Network Codes for DSNs: Regenerating Codes

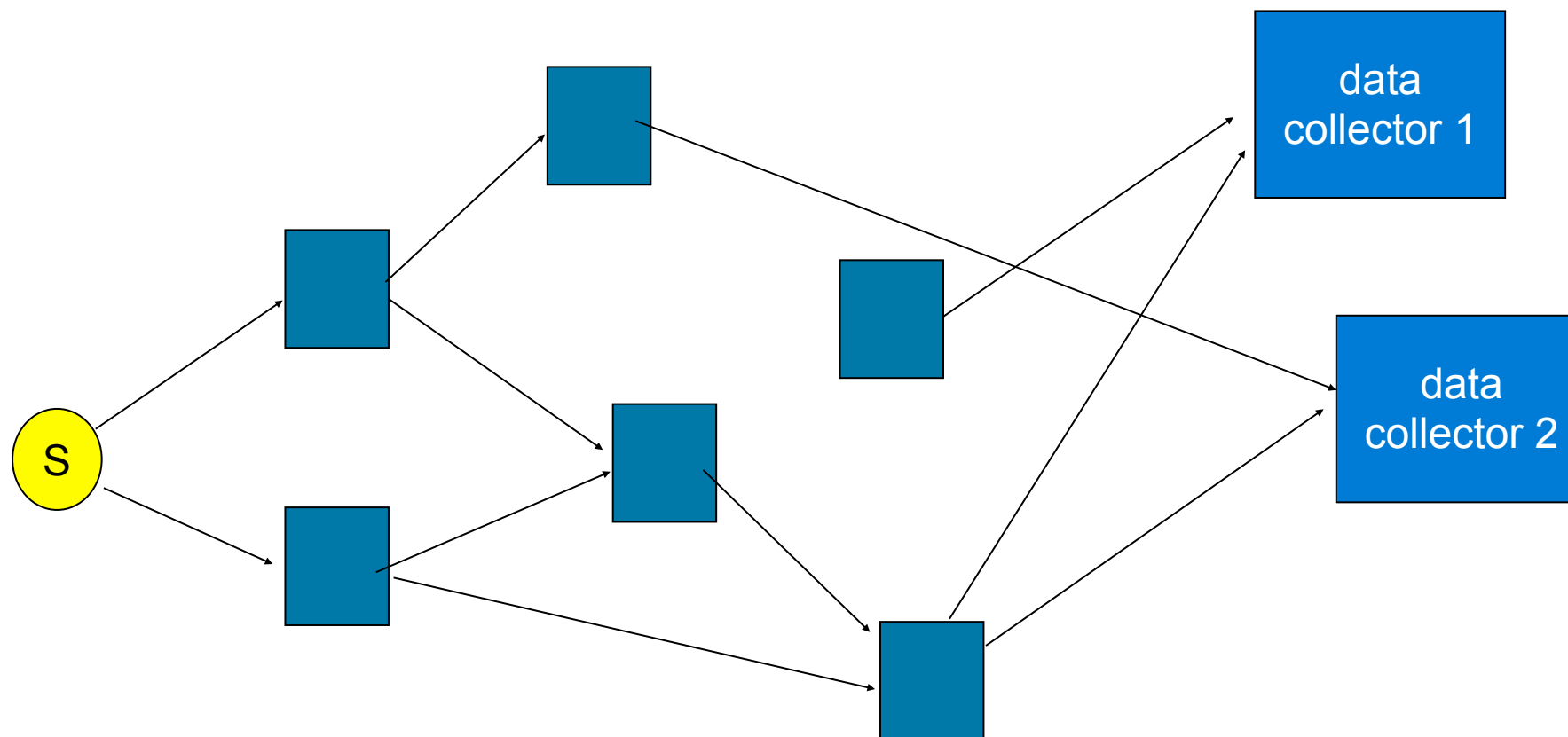
---

- \* Designed for a general class of DSNs
- \* Repair problem: optimization of the used network bandwidth
- \* More adapted to P2P wireless DSNs? ...

[DGW<sup>+</sup>10] A.G. Dimakis, P.B. Godfrey, Yunnan Wu, M.J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *Information Theory, IEEE Transactions on*, 56(9):4539 –4551, sept. 2010.

# Network Codes for DSNs: Regenerating Codes

**Idea:** 1) translate the upload/download problem in terms of network information flows

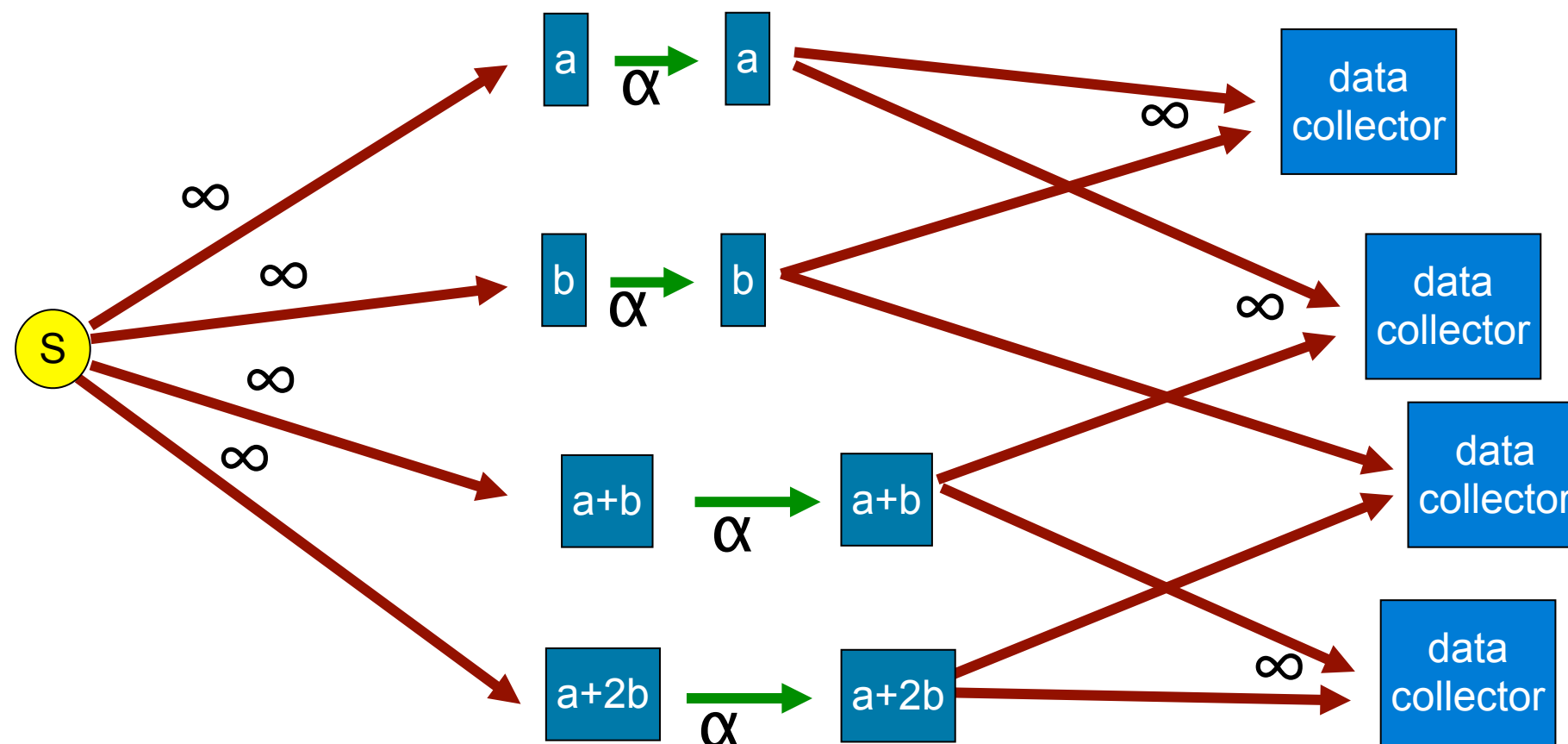


Hence, one can use the well-known cut-set bound in order to determine the amount information available at the collector

# Network Codes for DSNs: Regenerating Codes

**Idea:** 1) translate the upload/download problem in terms of network information flows  
2) use network coding to multicast

**Example** [DR13]:  $[4,2]$  MDS code seen as a network code



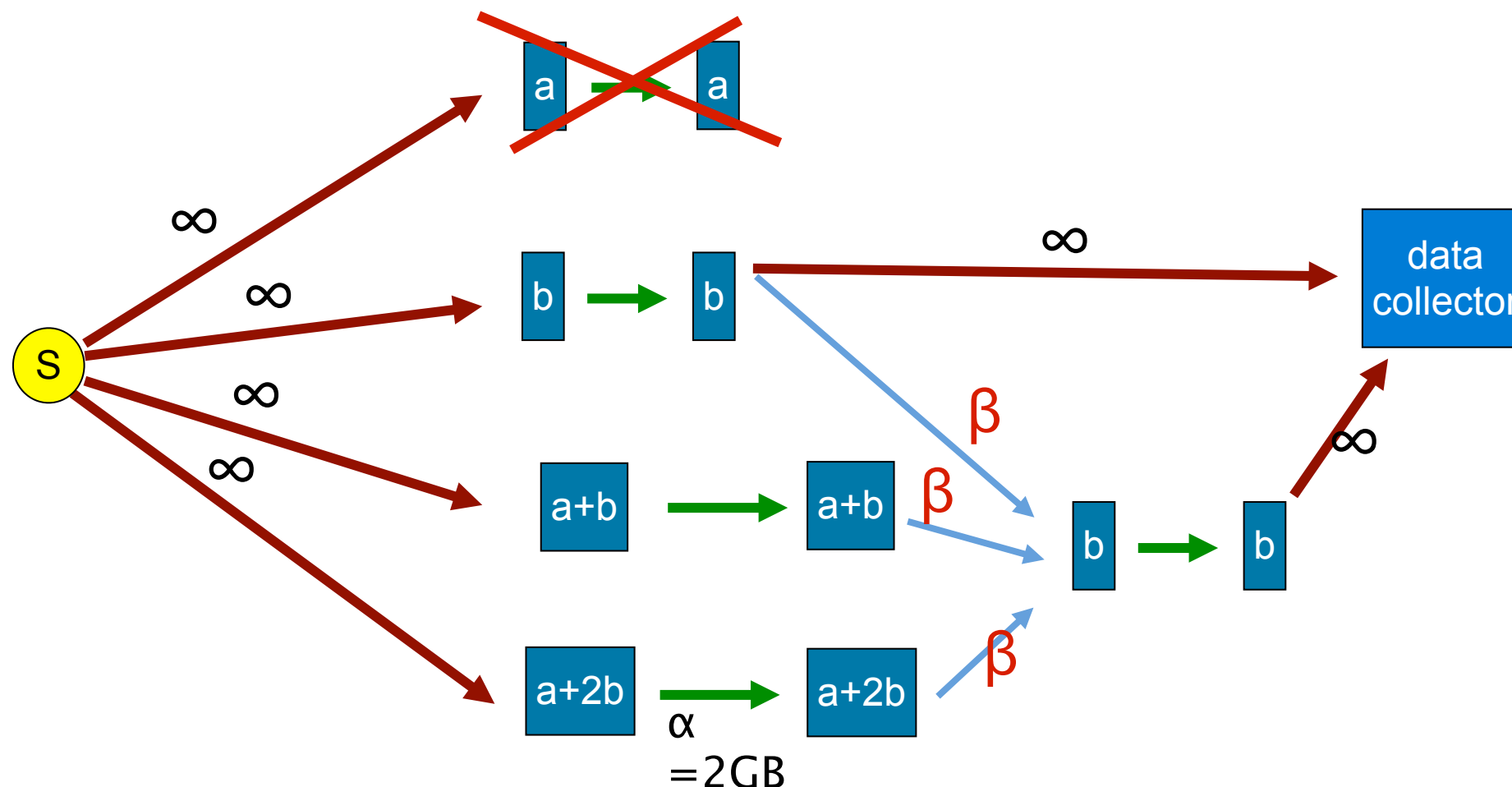
[DR13] A. Dimakis and K. Ramchandran. A (very long) tutorial on coding for distributed storage. presented at the IEEE ISIT'13, Istanbul, Turkey, 2013.

# Network Codes for DSNs: Regenerating Codes

**Idea:** 1) translate the upload/download problem in terms of network information flows

2) use network coding to multicast

3) formulate the repair optimisation problem to minimize the repair bandwidth  $B_R(t)$



# Network Codes for DSNs: Regenerating Codes

---

**Idea:** 1) translate the upload/download problem in terms of network information flows

2) use network coding to multicast

3) formulate the repair optimisation problem to minimize the repair bandwidth  $B_R(t)$

## *Definition*

*Exact repair* = repair operation where each erased symbol is reconstructed exactly.

*Functional repair* = repair operation where it is allowed to reconstruct a linear combination containing the erased symbol

# Network Codes for DSNs: Regenerating Codes

---

**Idea:** 1) translate the upload/download problem in terms of network information flows

2) use network coding to multicast

3) formulate the repair optimisation problem to minimize the repair bandwidth  $B_R(t)$

## *Definition*

*Exact repair* = repair operation where each erased symbol is reconstructed exactly.

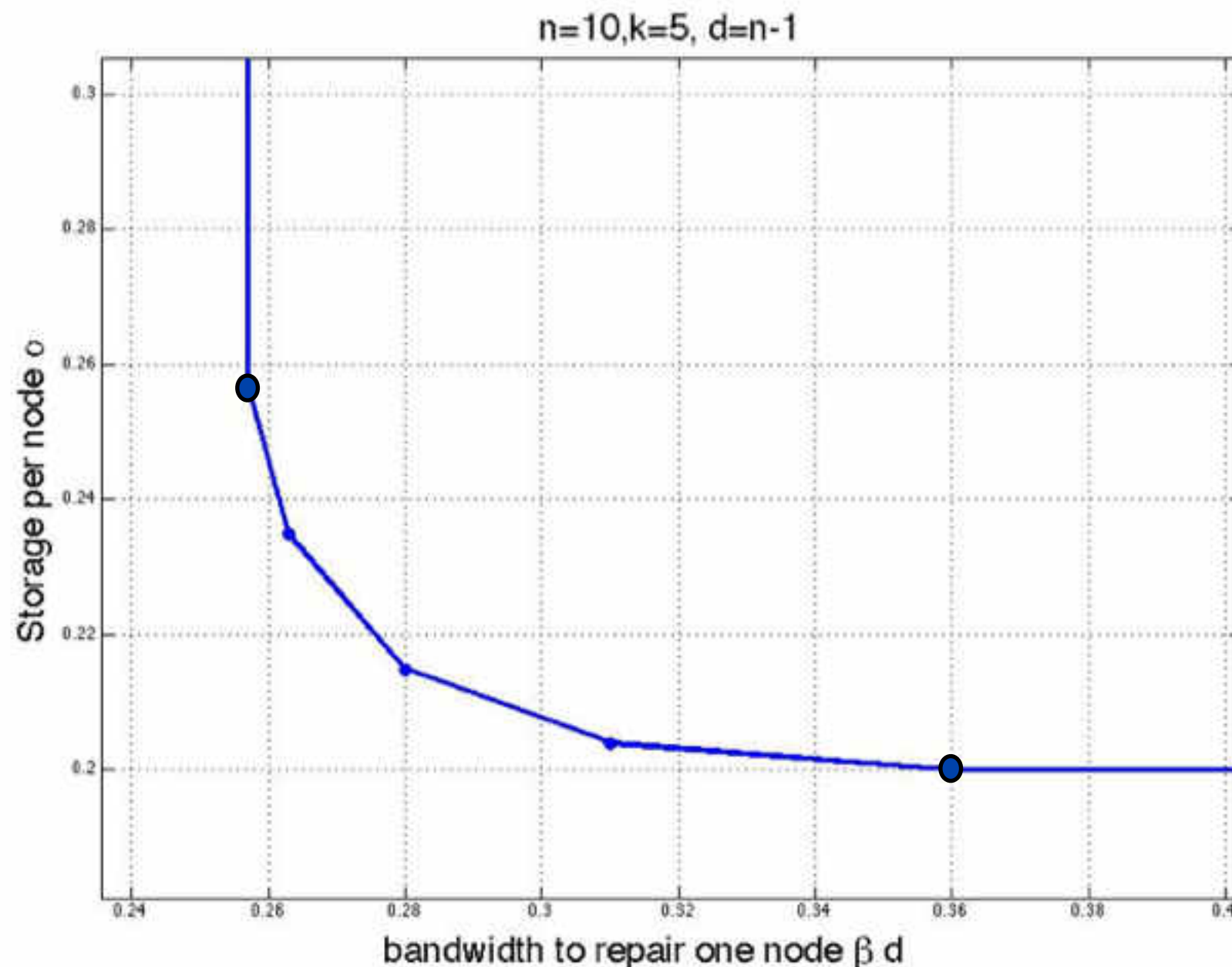
*Functional repair* = repair operation where it is allowed to reconstruct a linear combination containing the erased symbol

*Main tradeoff on the repair bandwidth vs. rate  $R$ :*

Functional repair bound = Cut-set bound

# Network Codes for DSNs: Regenerating Codes

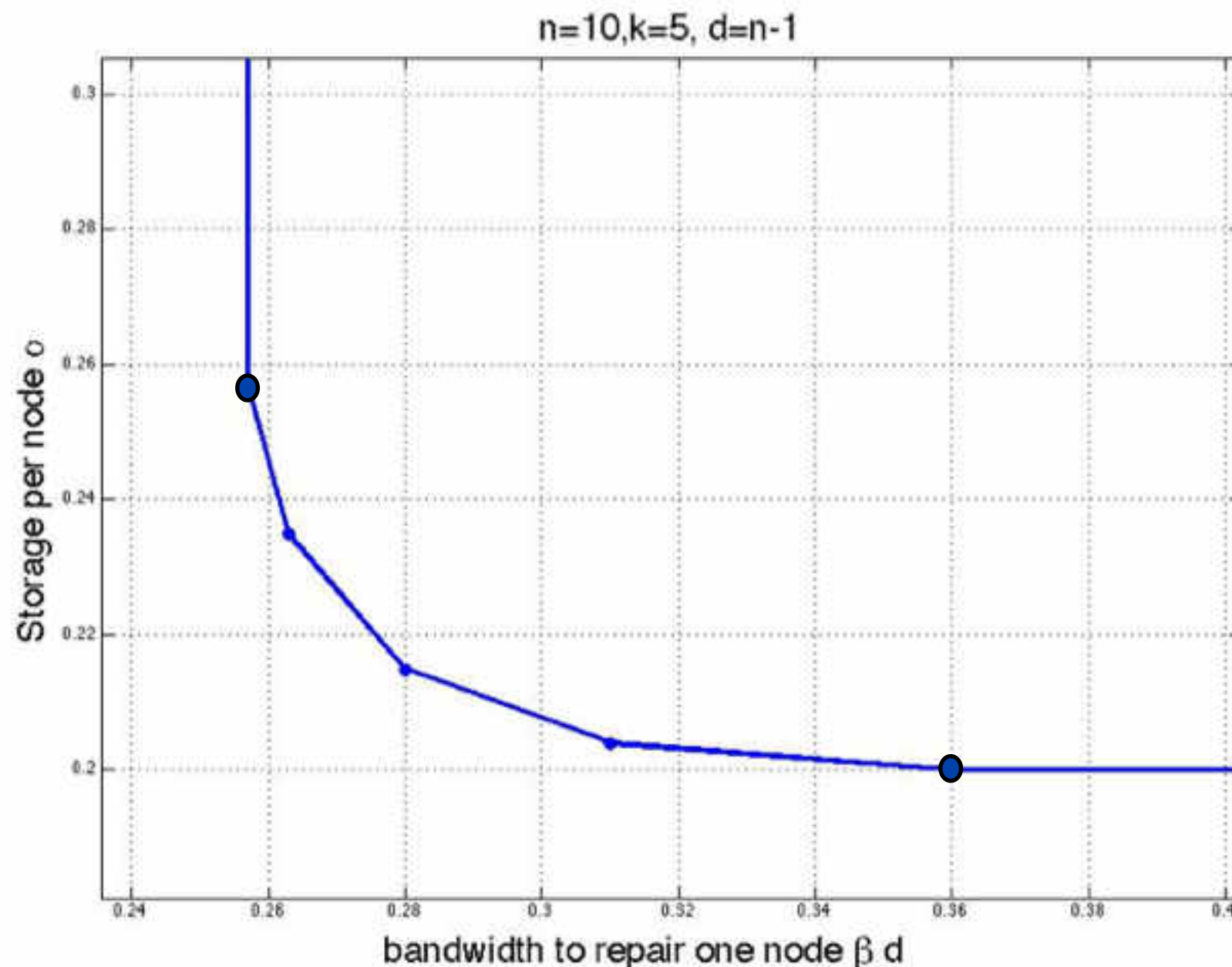
In literature, the functional repair region is usually plotted for some given  $R$ . The bound is tight for two extreme points.



# Network Codes for DSNs: Regenerating Codes

In literature, the functional repair region is usually plotted for some given  $R$ . The bound is tight for two extreme points.

Exact repair region?



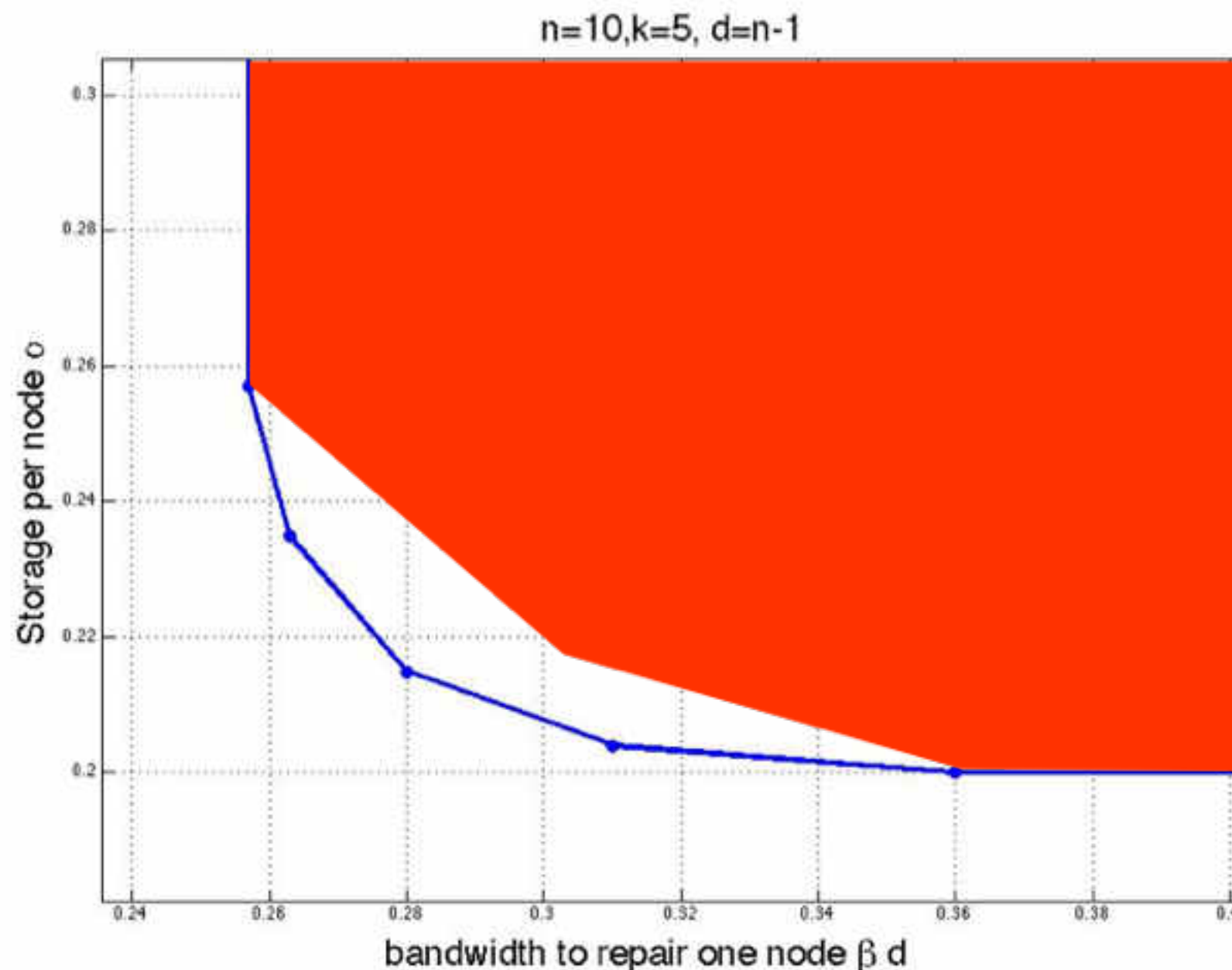


# Network Codes for DSNs: Regenerating Codes

In literature, the functional repair region is usually plotted for some given  $R$ .

Exact repair region? Strictly included into the functional repair region [Tia13]

[Tia13] C. Tian. Rate region of the (4,3,3) exact-repair regenerating code. In *Proceedings of ISIT'2013*, Istanbul, Turkey, July 2013.



# Network Codes for DSNs: Regenerating Codes

---

## Some code constructions:

- **Functional repair**

For one (or both) extreme points, based on product-matrices [RSK11], on local parities search [HLS13], on the theory of invariant subspaces [KK12], ...

- **Exact repair**

Based on the analysis of the network information flow [WZ13], by local parities search [PLD12], on geometric constructions [PNERR11],...

- [RSK11] K. V. Rashmi, N.B. Shah, and P.V. Kumar. Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction. *Information Theory, IEEE Transactions on*, 57(8):5227–5239, 2011.
- [HLS13] Yuchong Hu, Patrick P.C. Lee, and Kenneth W. Shum. Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems. In *INFOCOM, 2013 Proceedings IEEE*, pages 2355–2363, 2013.
- [KK12] G.M. Kamath and P.V. Kumar. Regenerating codes: A reformulated storage-bandwidth trade-off and a new construction. In *Communications (NCC), 2012 National Conference on*, pages 1–5, 2012.
- [WZ13] Anyu Wang and Zhifang Zhang. Exact cooperative regenerating codes with minimum-repair-bandwidth for distributed storage. In *INFOCOM, 2013 Proceedings IEEE*, pages 400–404, 2013.
- [PLD<sup>+</sup>12] D.S. Papailiopoulos, Jianqiang Luo, A.G. Dimakis, Cheng Huang, and Jin Li. Simple regenerating codes: Network coding for cloud storage. In *INFOCOM, 2012 Proceedings IEEE*, pages 2801–2805, 2012.
- [PNERR11] S. Pawar, N. Noorshams, S. El Rouayheb, and K. Ramchandran. Dress codes for the storage cloud: Simple randomized constructions. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 2338–2342, 2011.

# Network Codes for DSNs: Regenerating Codes

---

- \* Regenerating codes: well-developed **theory of network coding**, allowing to obtain bounds on performance of DSNs
- \* Good performance in terms of **repair bandwidth**
- \* Examples of **designed systems**: NCCloud, CORE
- \* Such a design may be useful for DSNs with low-capacity links

# Network Codes for DSNs: Regenerating Codes

---

But:

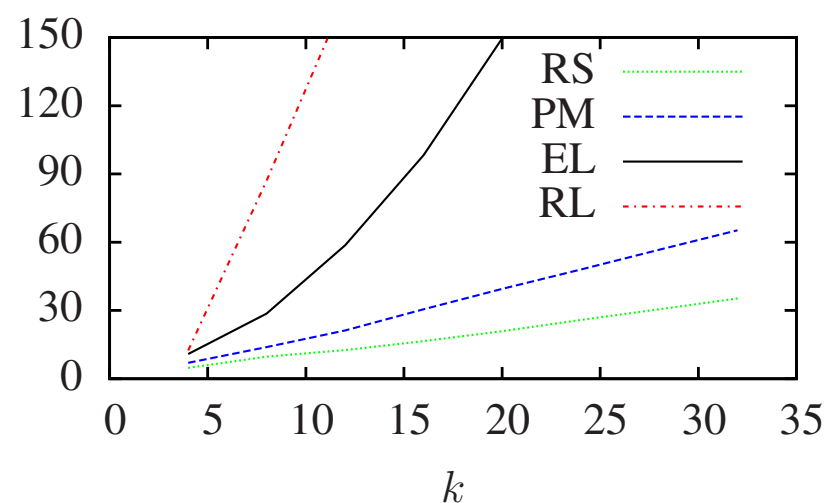
- Decoding access  $\uparrow$
- Code rates  $\downarrow$
- Encoding/decoding complexity  $\uparrow$
- For the functional repair, one should keep track of used code instances
- The optimisation problem cannot be formulated in terms of the repair access

# Network Codes for DSNs: Regenerating Codes

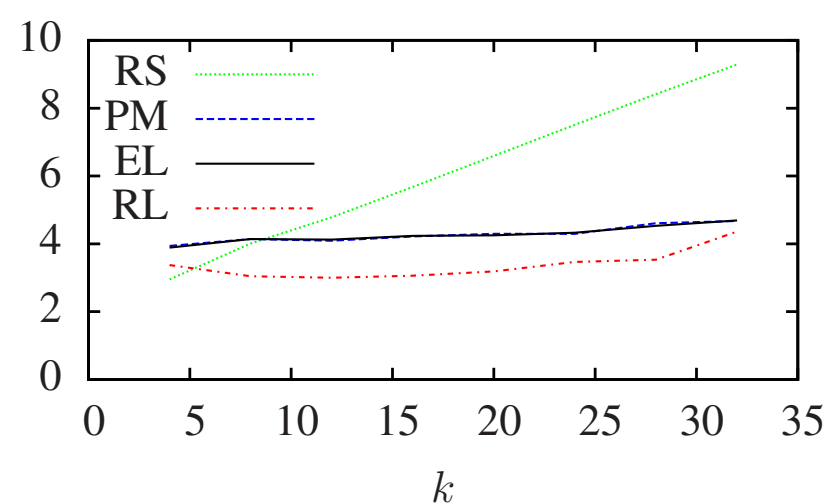
But:

- Decoding access  $\uparrow$
- Code rates  $\downarrow$
- Encoding/decoding complexity  $\uparrow$
- For the functional repair, one should keep track of used code instances
- The optimisation problem cannot be formulated in terms of the repair access

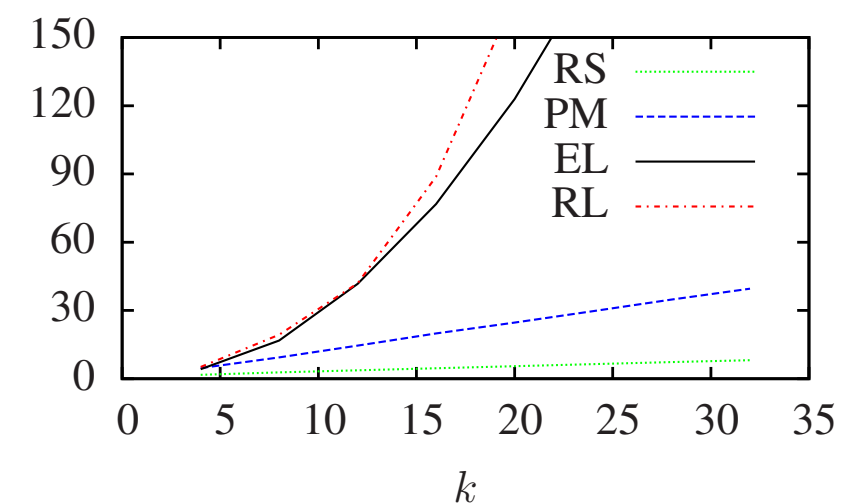
Illustration from [JKLS13]: comparison of various regenerating codes with RS codes



(a) Time for encoding a file in seconds



(b) Time for repairing a lost device in seconds



(c) Time for decoding a file in seconds

[JKLS<sup>+</sup>13] Steve Jiekak, Anne-Marie Kermarrec, Nicolas Le Scouarnec, Gilles Straub, and Alexandre Van Kempen. Regenerating codes: a system perspective. *SIGOPS Oper. Syst. Rev.*, 47(2):23–32, July 2013.

# Sparse-Graph Coding for DSNs

---

**Remark:** in contrast to the codes, presented before, sparse-graph codes have **encoding/decoding complexity  $\sim O(n)$** .

Also, the iterative decoding algorithm can correct erasures beyond the erasure-correcting capability

# Sparse-Graph Coding for DSNs

---

**Remark:** in contrast to the codes, presented before, sparse-graph codes have **encoding/decoding complexity**  $\sim O(n)$ .

Also, the iterative decoding algorithm can correct erasures beyond the erasure-correcting capability

**Particularity:** the parity matrix  $H$  has a small number of non-zero elements per row/column, of order  $o(n)$ .

# Sparse-Graph Coding for DSNs

---

**Remark:** in contrast to the codes, presented before, sparse-graph codes have **encoding/decoding complexity**  $\sim O(n)$ .

Also, the iterative decoding algorithm can correct erasures beyond the erasure-correcting capability

**Particularity:** the parity matrix  $H$  has a small number of non-zero elements per row/column, of order  $o(n)$ .

*Q: What is the repair access  $R(1)$ ?*



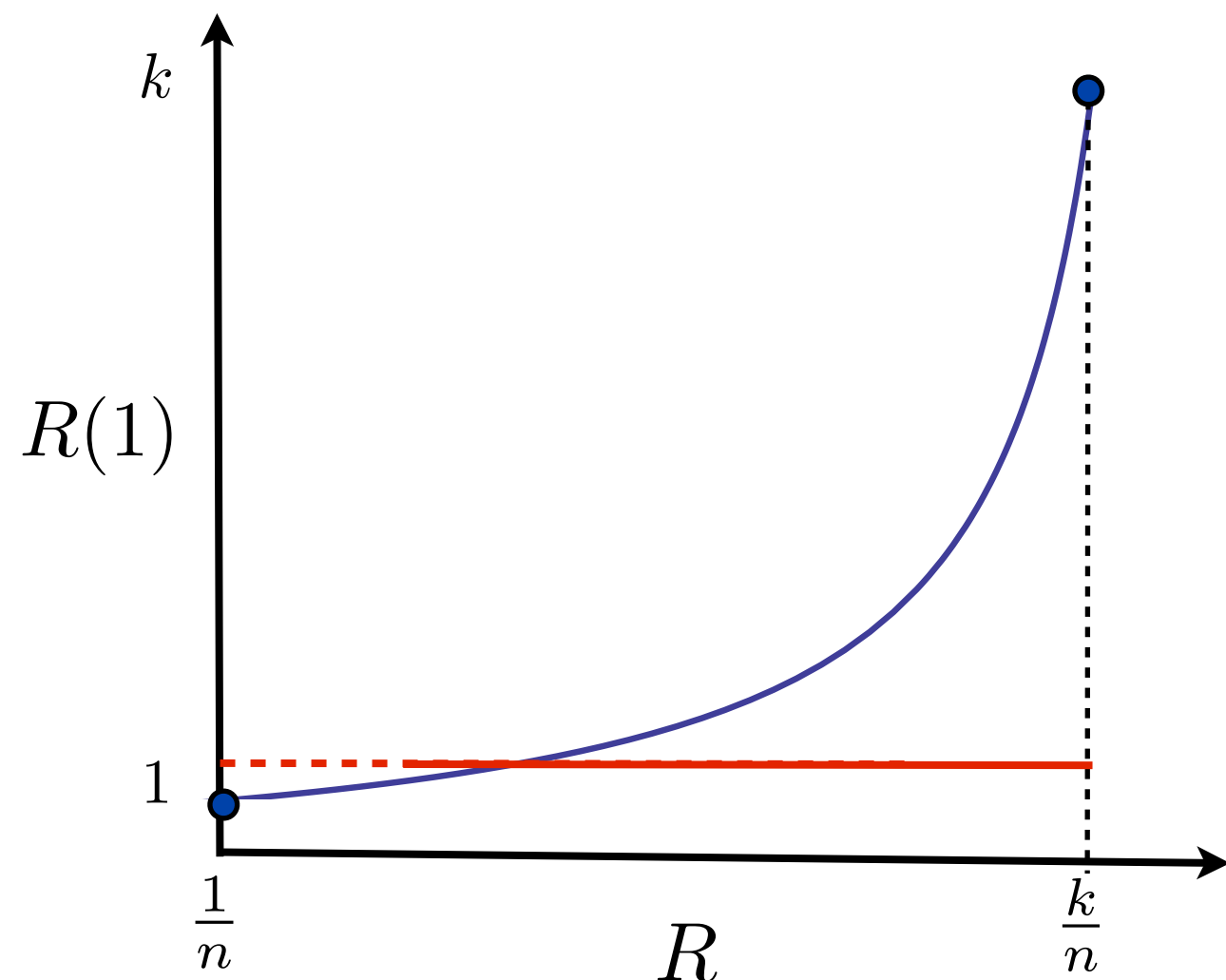
# Sparse-Graph Coding for DSNs

**Remark:** in contrast to the codes, presented before, sparse-graph codes have **encoding/decoding complexity  $\sim O(n)$** .

Also, the iterative decoding algorithm can correct erasures beyond the erasure-correcting capability

**Particularity:** the parity matrix  $H$  has a small number of non-zero elements per row/column, of order  $o(n)$ .

**Q: What is the repair access  $R(1)$ ?**  
 **$R(1)=c-1$  if the rows have weight  $c$ .**



# Sparse-Graph Coding for DSNs

**Remark:** in contrast to the codes, presented before, sparse-graph codes have **encoding/decoding complexity**  $\sim O(n)$ .

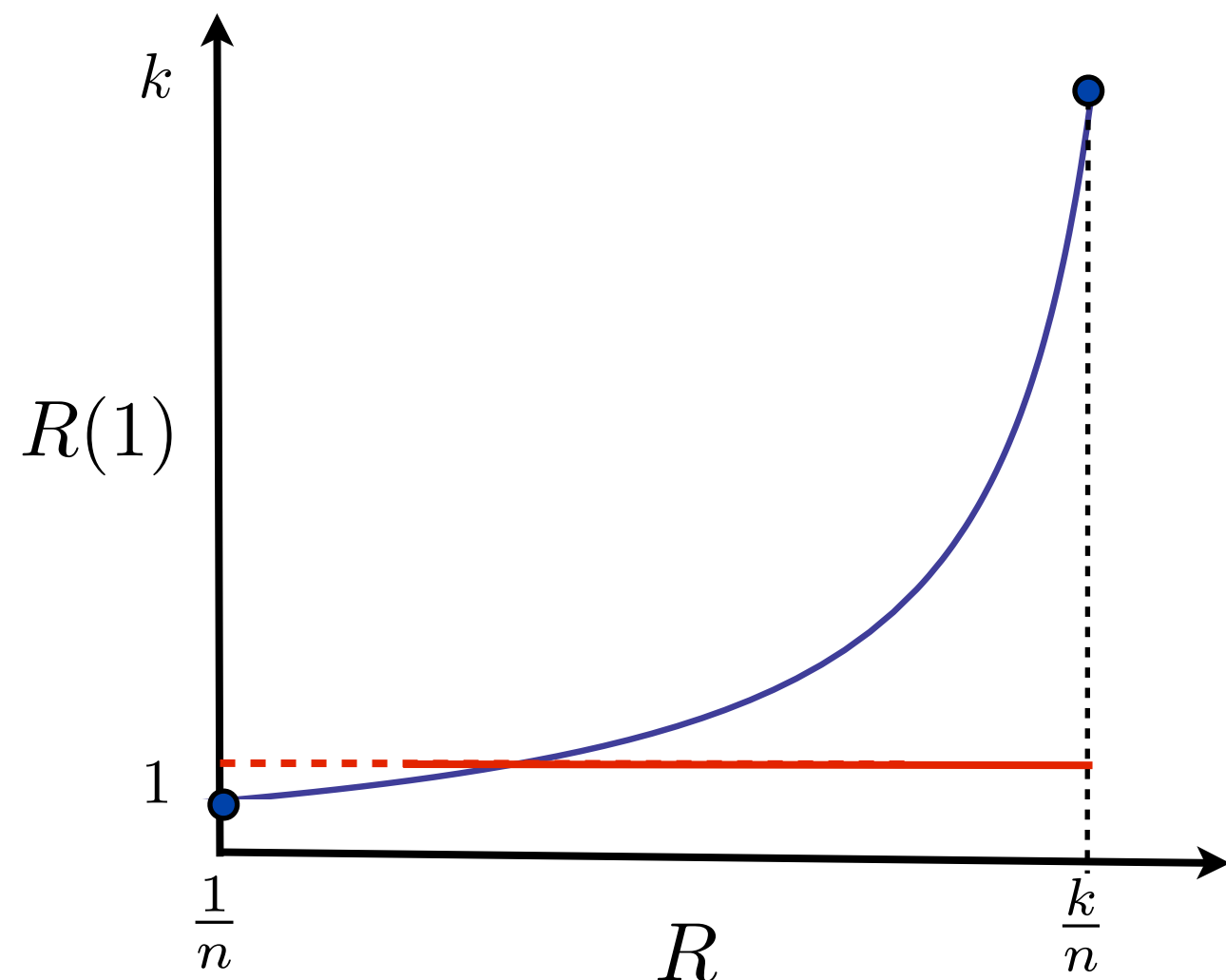
Also, the iterative decoding algorithm can correct erasures beyond the erasure-correcting capability

**Particularity:** the parity matrix  $H$  has a small number of non-zero elements per row/column, of order  $o(n)$ .

**Q: What is the repair access  $R(1)$ ?**

**$R(1)=c-1$  if the rows have weight  $c$ .**

Moreover, sparse-graph codes are **scalable** in  $n$ .



# Sparse-Graph Coding for DSNs

---

But... disadvantages are also serious:

- Sparse-graph codes are efficient starting from some **sufficiently large  $n$**  (several hundreds)
- To obtain a good performance, some **additional constraints on minimum distance** should be verified
- A sparse-graph code is not MDS and **erasure-correction capability is not guaranteed** in general, but only in some very particular cases. This makes the code **allocation-dependent**.
- **Systematic sparse-graph codes are bad**. Hence, non-systematic constructions should be used  $\Rightarrow$  non-zero decoding complexity

# Sparse-Graph Coding for DSNs

---

## Some sparse-graph code constructions:

- **LDPC codes**: first test of simple codes [Pla03], LDPC codes tested in CERN [GKS07], SpreadStore prototype based on projective geometry codes [HJC10]
- **Tornado**: prototype for archival storage [WT06]
- **LT codes**: [XC05]

[Pla03] James S. Plank. On the practical use of ldpc erasure codes for distributed storage applications. Technical report, 2003.

[GKS07] Benjamin Gaidioz, Birger Kobnitz, and Nuno Santos. Exploring high performance distributed file storage using ldpc codes. *Parallel Comput.*, 33(4-5):264–274, May 2007.

[HJC<sup>+</sup>10] Subramanyam G. Harihara, Balaji Janakiram, M. Girish Chandra, Aravind Kota Gopalakrishna, Swanand Kadhe, P. Balamuralidhar, and B. S. Adiga. Spreadstore: A ldpc erasure code scheme for distributed storage system. In *DSDE*, pages 154–158. IEEE Computer Society, 2010.

[WT06] M. Woitaszek and H.M. Tufo. Fault tolerance of tornado codes for archival storage. *High-Performance Distributed Computing, International Symposium on*, 0:83–92, 2006.

[XC05] Huaxia Xia and Andrew A. Chien. Robustore: Robust performance for distributed storage systems. Technical report, 2005.

# Sparse-Graph Coding for DSNs

## Our prototype:

- based on structured LDPC codes
- erasure-correction guarantee
- quasi-systematic structure
- joint allocation protocol

| Solution     | Number of disjoint networks | Maximum guaranteed protection | Cost |
|--------------|-----------------------------|-------------------------------|------|
| RAID 1       | 5                           | 4                             | 31   |
| RAID 5       | 4                           | 1                             | 4    |
| RAID 6       | 3                           | 2                             | 6    |
| Our solution | 1                           | 4                             | 4    |

TABLE IV

COMPARISON OF DIFFERENT SOLUTIONS IN THE CASE OF 36 DISKS

# Sparse-Graph Coding for DSNs

## Our prototype:

- based on structured LDPC codes
- erasure-correction guarantee
- quasi-systematic structure
- joint allocation protocol

| Operation         | Time (ms)  | Throughput (MBytes/s) |
|-------------------|------------|-----------------------|
| Encoding          | 0.346      | 33.74                 |
| Repair of 1 disk  | 215/ 0.011 | 1111                  |
| Repair of 3 disks | 281/ 0.022 | 546.4                 |

TABLE I

NECESSARY TIME AND THROUGHPUT OF ENCODING, DECODING OPERATIONS OVER IN THE NETWORK OF 12 DISKS AND ALLOWING UP TO 3 SIMULTANEOUS FAILURES

# Coding for DSNs: Important Points

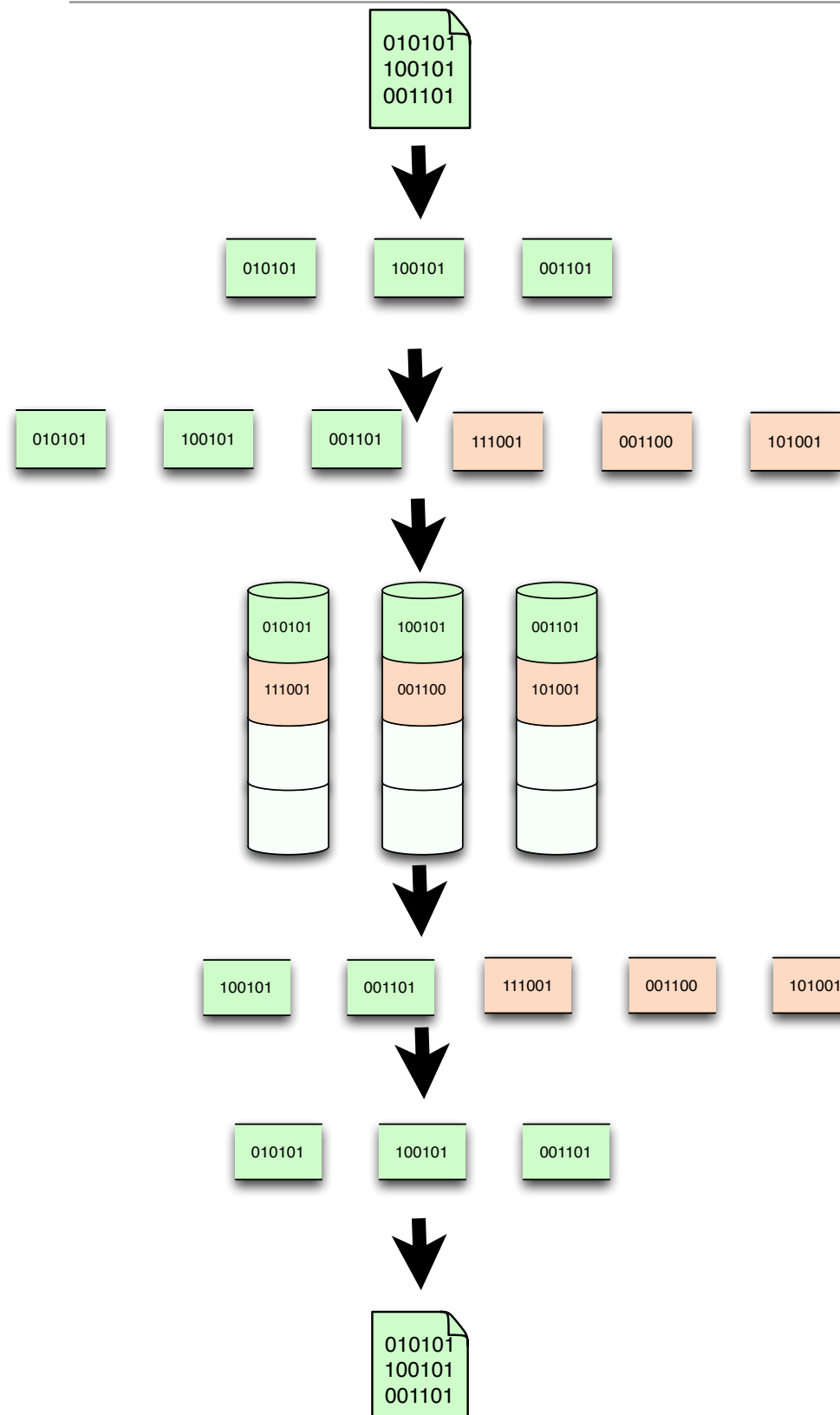
---

- \* **Common basis of comparison** is needed (at least for similar applications)
- \* In many cases, **fundamental tradeoffs** should still be derived
- \* General **code constructions**, both algebraic and probabilistic are welcome!

## Part 3: Allocation Problem in DSNs




# Our Previous Example: BEC Model



# encoded segments = # disks =  $n$

1 segment  $\rightarrow$  1 separate disk

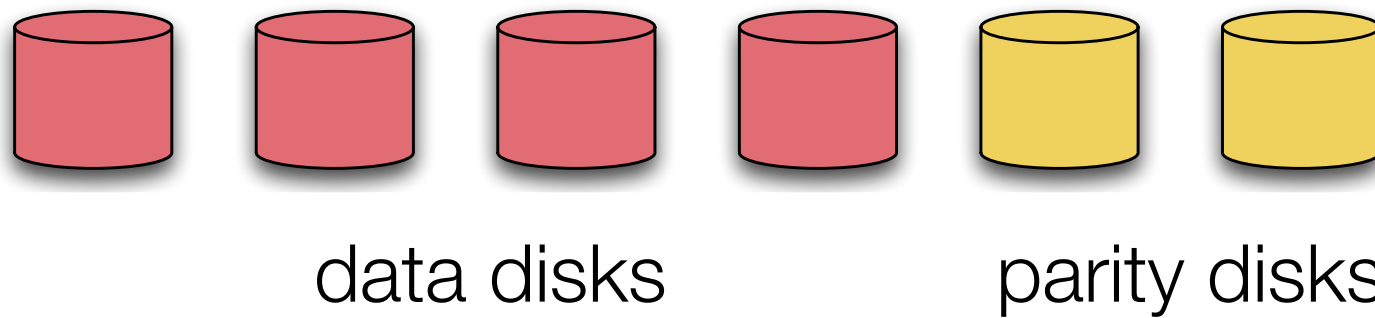
independent loss model, failure probability =  $p$

 gives us the channel model, which is symbol erasure channel (binary erasure channel) with probability  $p$

# New Example 1

---

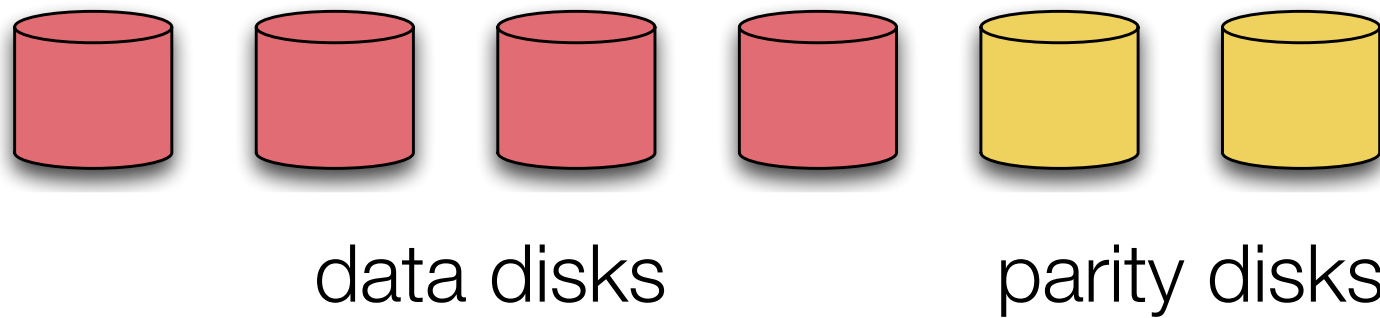
Therefore, if one uses a MDS code, one has the scheme below?



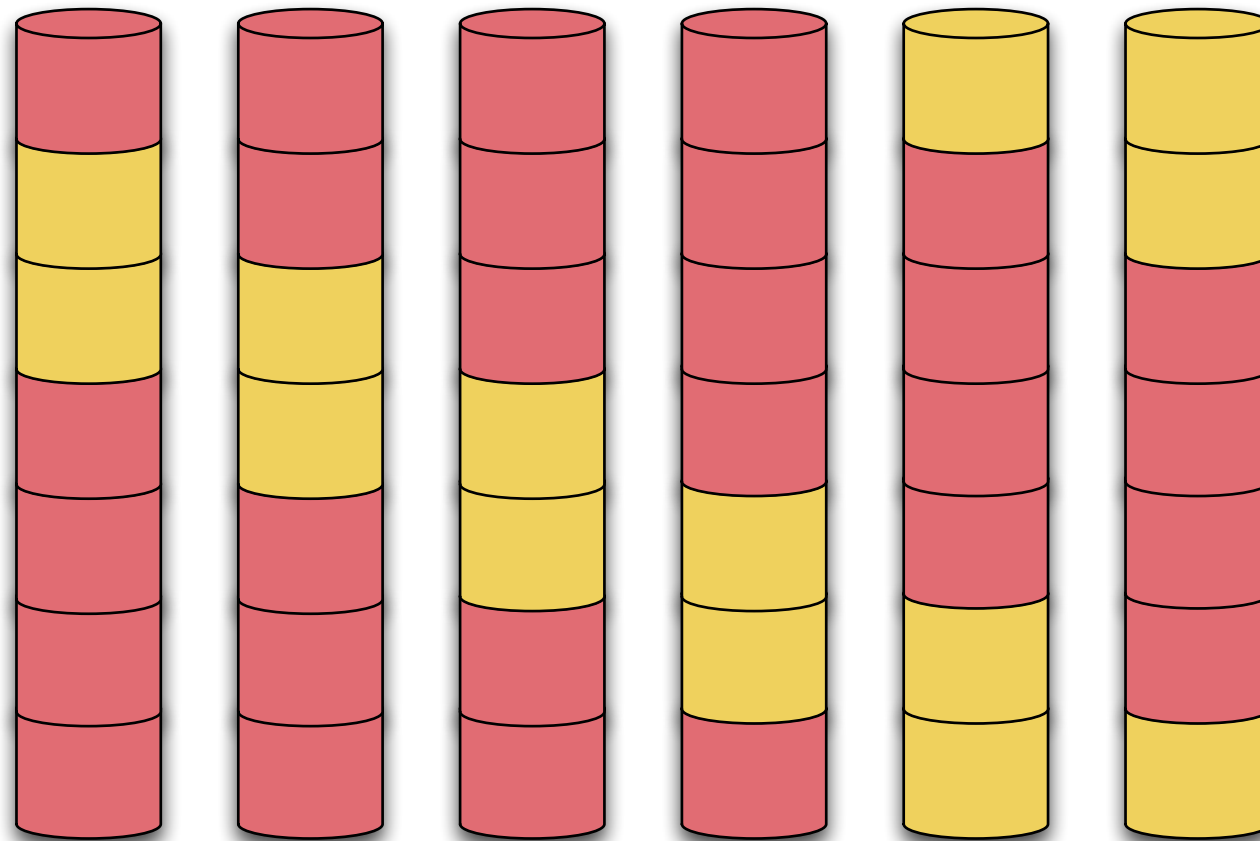
# New Example 1

---

Therefore, if one uses a MDS code, one has the scheme below?



Practical RAID DCN:

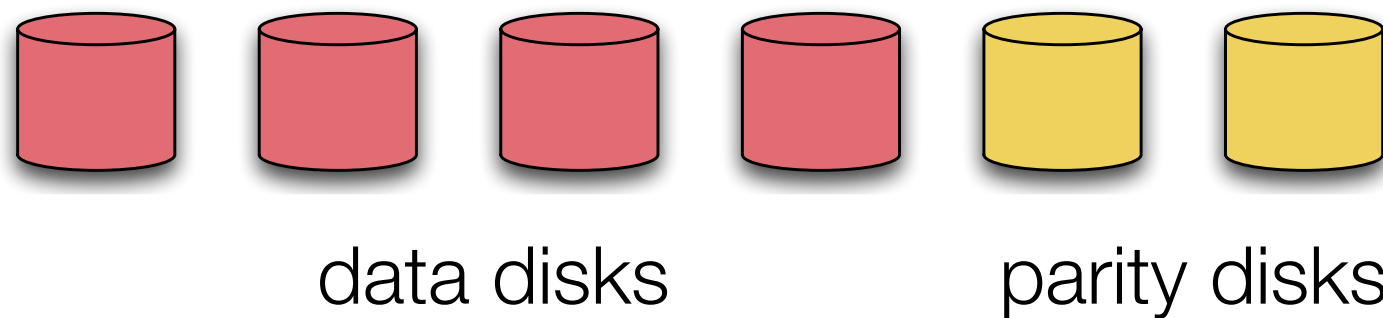


*Q: What is the reason?*

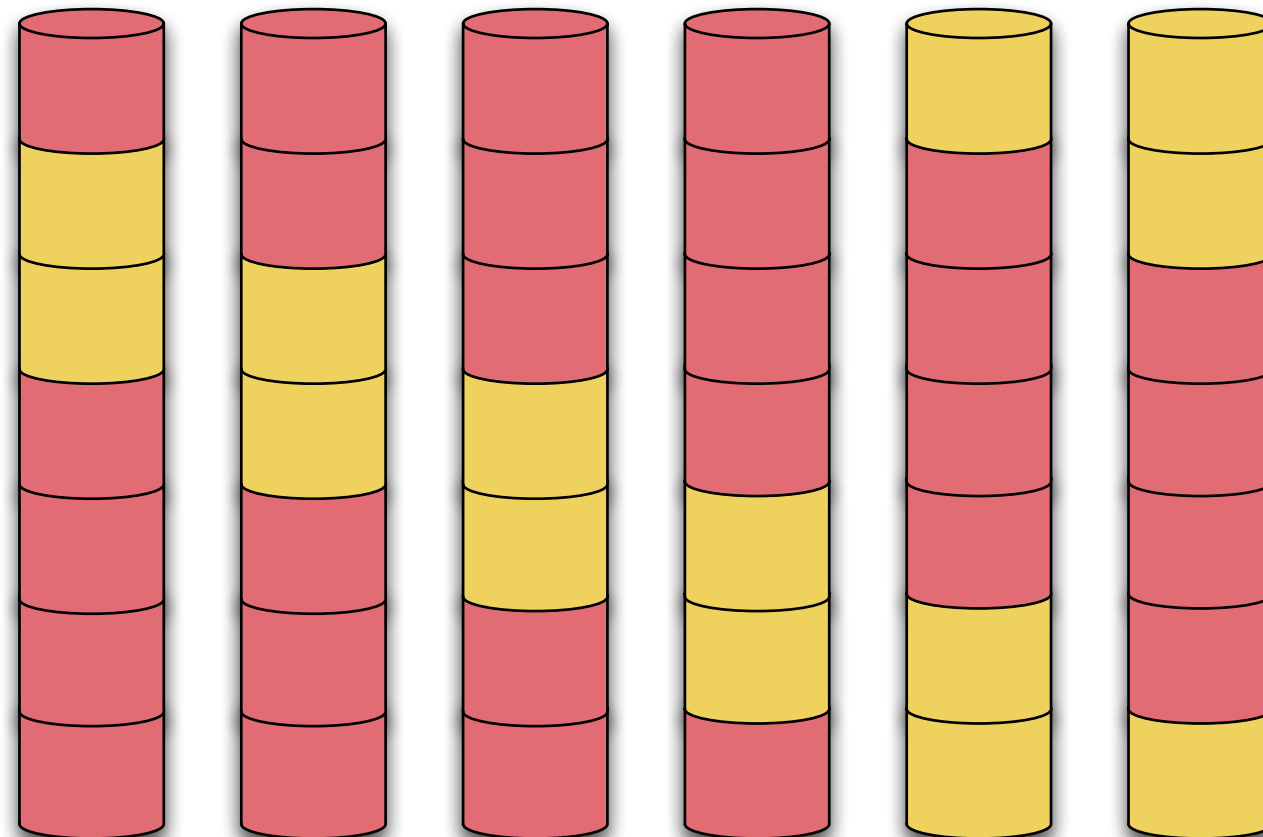
# New Example 1

---

Therefore, if one uses a MDS code, one has the scheme below?



Practical RAID DCN:



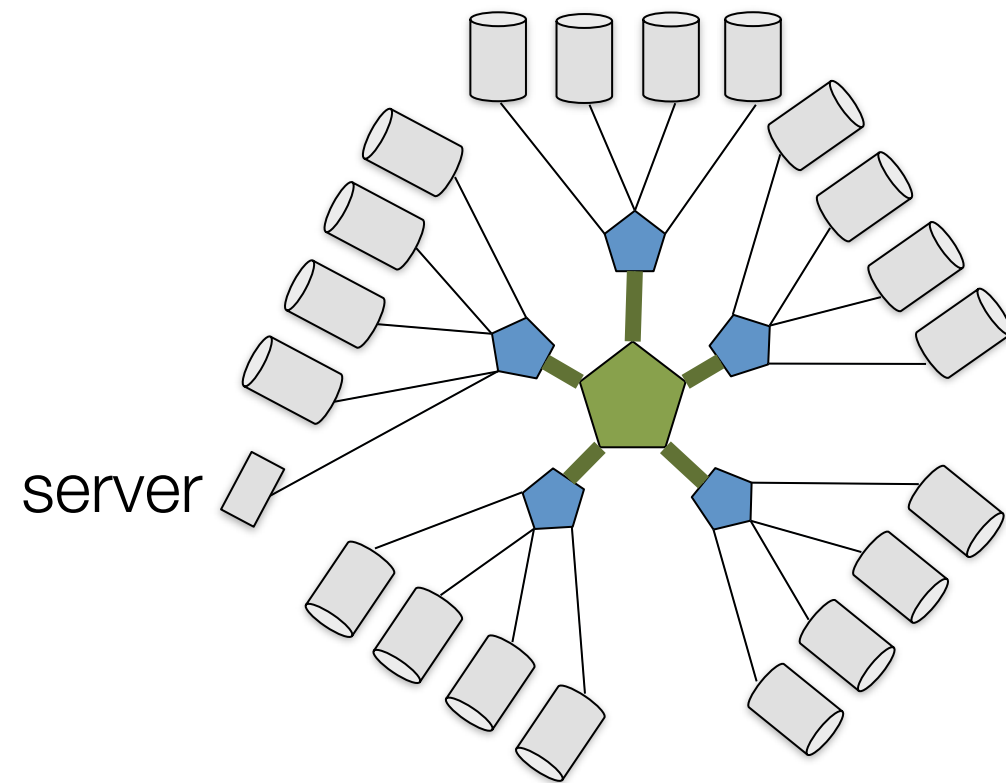
*Q: What is the reason?*

*1. rebalancing  
upload and repair  
loads over the disks*

*2. dealing with  
multiple files/users*

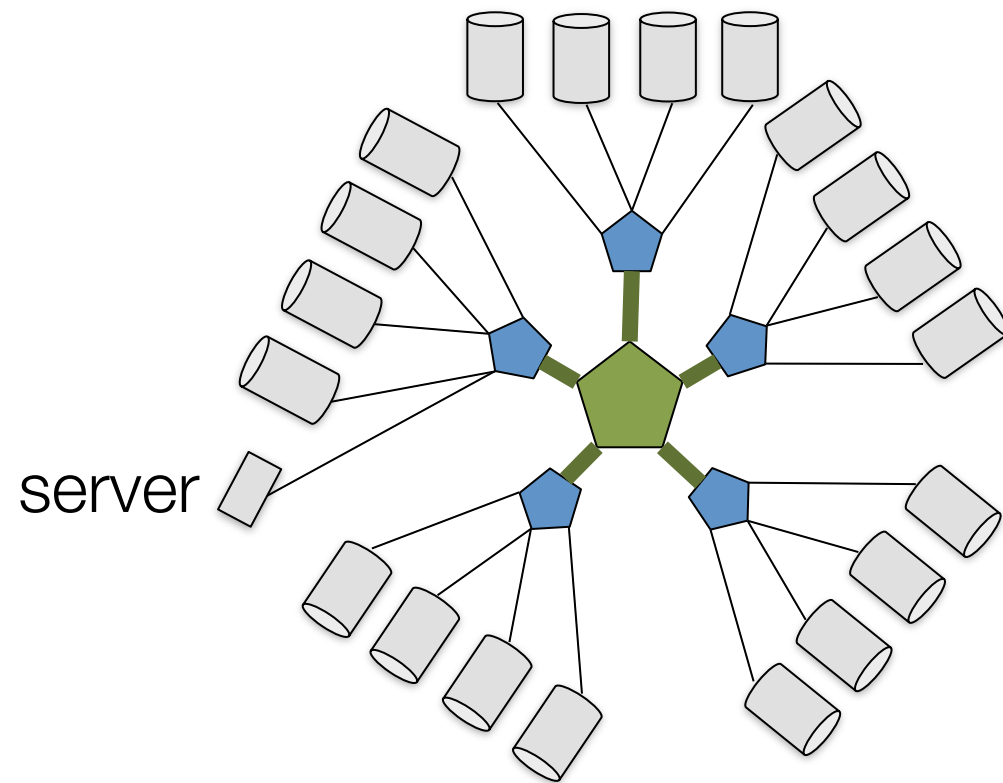
# New Example 2

---



# New Example 2

---



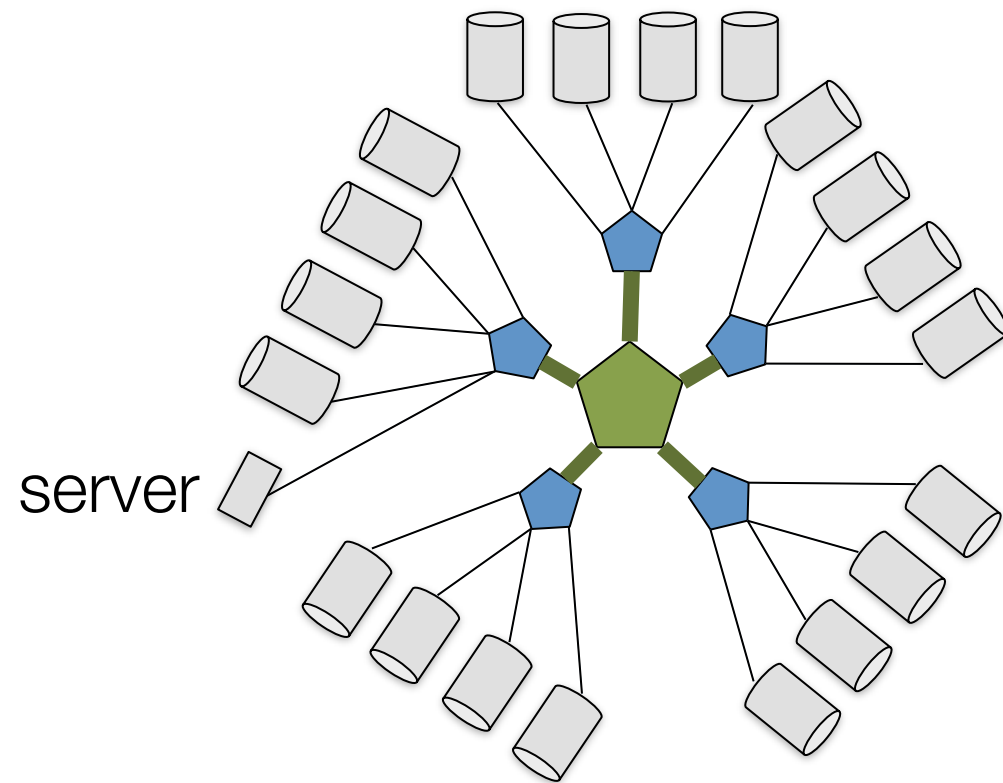
*Q: The allocation, is it network-dependent?*

*No if:*

*Yes if:*

# New Example 2

---



*Q: The allocation, is it network-dependent?*

*No if: all the disks fail independently from each other  
and  
the network bandwidth is large enough*

*Yes if: otherwise*

# More Questions to Ask

---

Is the allocation:

- code-dependent? (see the part on coding)
- channel- or bandwidth-dependent?

...

**Consequence:** a change in the failure or access model change the model of the equivalent “communication channel”.

Hence, a new allocation protocol is needed (and possibly, an allocation-aware code design)



# Allocation Problems

---

Optimal allocation of data segments (code symbols) in a DSN:

- to maximize reliability
- to provide better quality of service
- to balance stored data in the network
- to balance network load from each of the nodes

# Allocation Problems

---

## Optimal allocation of data segments (code symbols) in a DSN:

- to maximize reliability
- to provide better quality of service
- to balance stored data in the network
- to balance network load from each of the nodes

## Possible considerations to take into account in order to build the equivalent channel model:

- network topology
- single user/multiple users
- correlated disk failures
- heterogenous links
- codelength  $n \neq \text{\#disks}$

# Allocation Problems

---

Optimal allocation of data segments (code symbols) in a DSN:

- to maximize reliability
- to provide better quality of service
- to balance stored data in the network
- to balance network load from each of the nodes

Possible considerations to take into account in order to build the equivalent channel model:

- network topology
- single user/multiple users
- correlated disk failures
- heterogenous links
- codelength  $n \neq \# \text{disks}$

Moreover, one has to deal with the re-allocation if there is a change:

- in network topology
- in user access models

# Equivalent Channel Models and a Particular Allocation Example

---

Considering allocation problems or joint code-allocation design for DSNs is new in the coding/information theory community.

# Equivalent Channel Models and a Particular Allocation Example

---

Considering allocation problems or joint code-allocation design for DSNs is new in the coding/information theory community.

Two examples:

- **Markov reliability model** (“erasure channel with memory”)  
[FLP10], [NYGS06]
- **Block erasure channel** [JA13]

- [FLP<sup>+</sup>10] Daniel Ford, Francois Labelle, Florentina Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. Availability in globally distributed storage systems. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.
- [NYGS06] Suman Nath, Haifeng Yu, Phillip B. Gibbons, and Srinivasan Seshan. Subtleties in tolerating correlated failures in wide-area storage systems. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, NSDI’06, pages 17–17, Berkeley, CA, USA, 2006. USENIX Association.
- [JA13] A. Jule and I. Andriyanova. An efficient family of sparse-graph codes for use in data centers. 2013.

# Equivalent Channel Models and a Particular Allocation Example

---

Considering allocation problems or joint code-allocation design for DSNs is new in the coding/information theory community.

Two examples:

- **Markov reliability model** (“erasure channel with memory”)  
[FLP10], [NYGS06]
- **Block erasure channel** [JA13]

[FLP<sup>+</sup>10] Daniel Ford, Francois Labelle, Florentina Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. Availability in globally distributed storage systems. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[NYGS06] Suman Nath, Haifeng Yu, Phillip B. Gibbons, and Srinivasan Seshan. Subtleties in tolerating correlated failures in wide-area storage systems. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, NSDI’06, pages 17–17, Berkeley, CA, USA, 2006. USENIX Association.

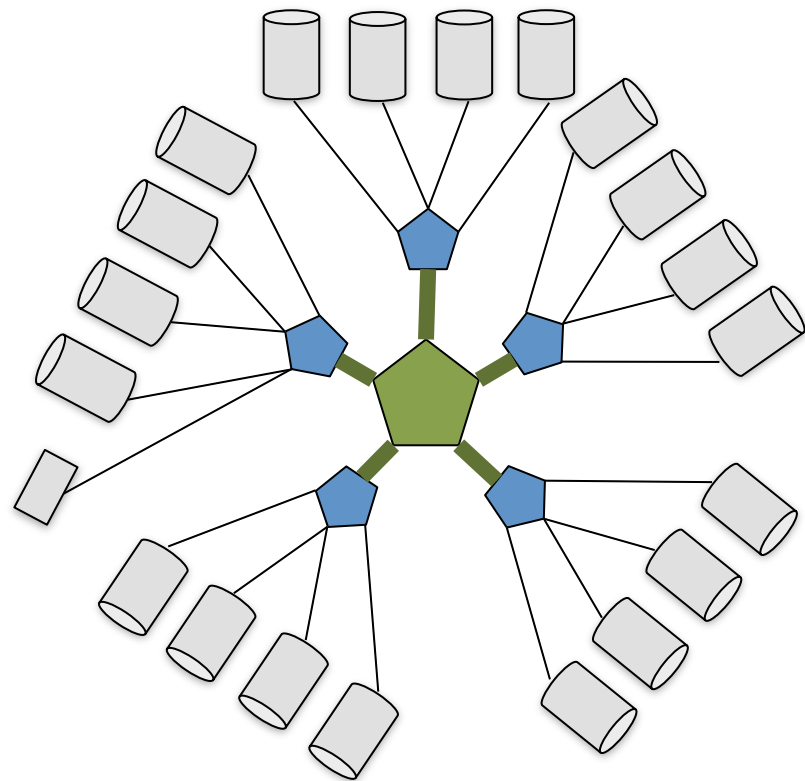
[JA13] A. Jule and I. Andriyanova. An efficient family of sparse-graph codes for use in data centers. 2013.

One more interesting work (P2P DSNs with MDS coding): [LDH12]

[LDH12] D. Leong, A.G. Dimakis, and Tracey Ho. Distributed storage allocations. *Information Theory, IEEE Transactions on*, 58(7):4733–4752, 2012.

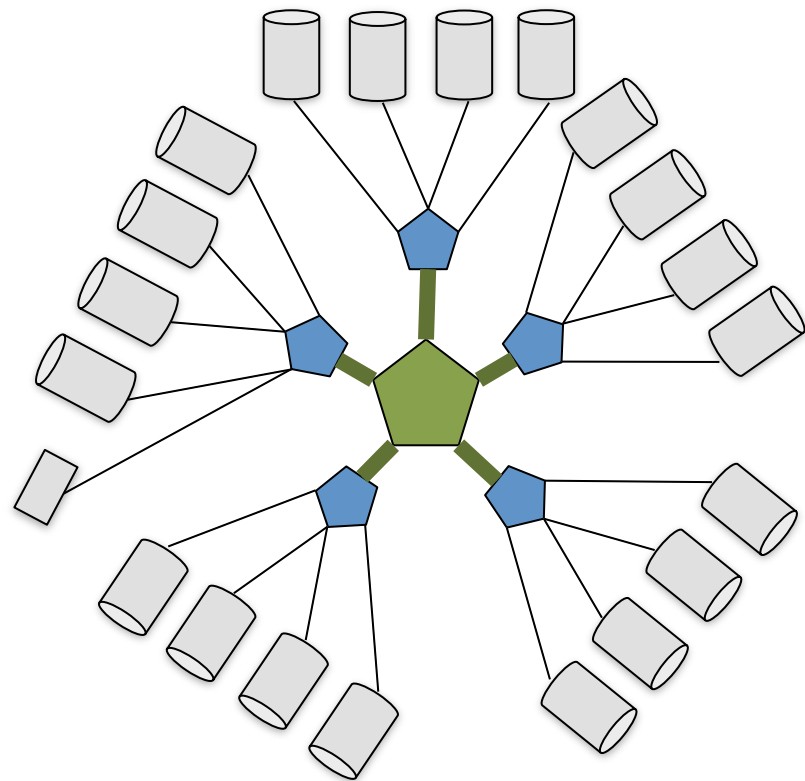
# Erasure Channel with Memory

---



**Modelling disk failures in a DCN:**  
- can a single disk failure imply the **failure of others**?

# Erasure Channel with Memory

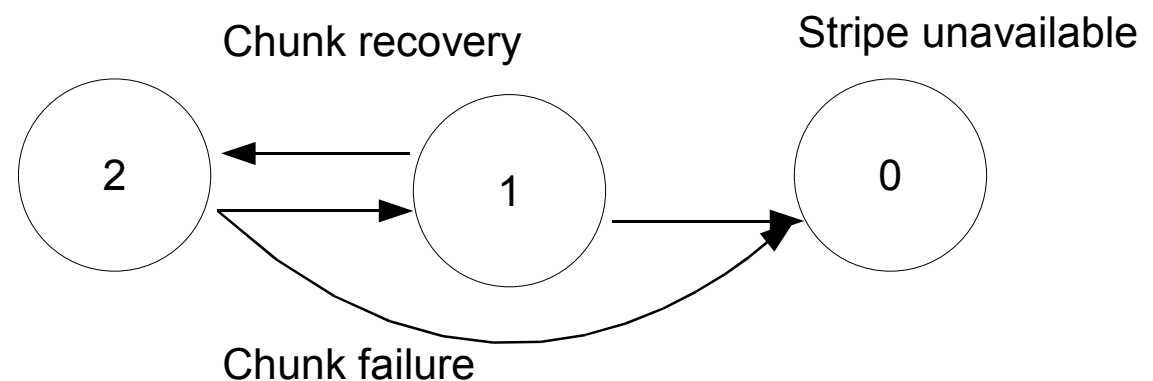


## Modelling disk failures in a DCN:

- can a single disk failure imply the **failure of others**?

[FLP10]: models based on measurements made over Google DCNs:

- a disk (chunk) failure is probable to cause the failure of the whole cluster (stripe)

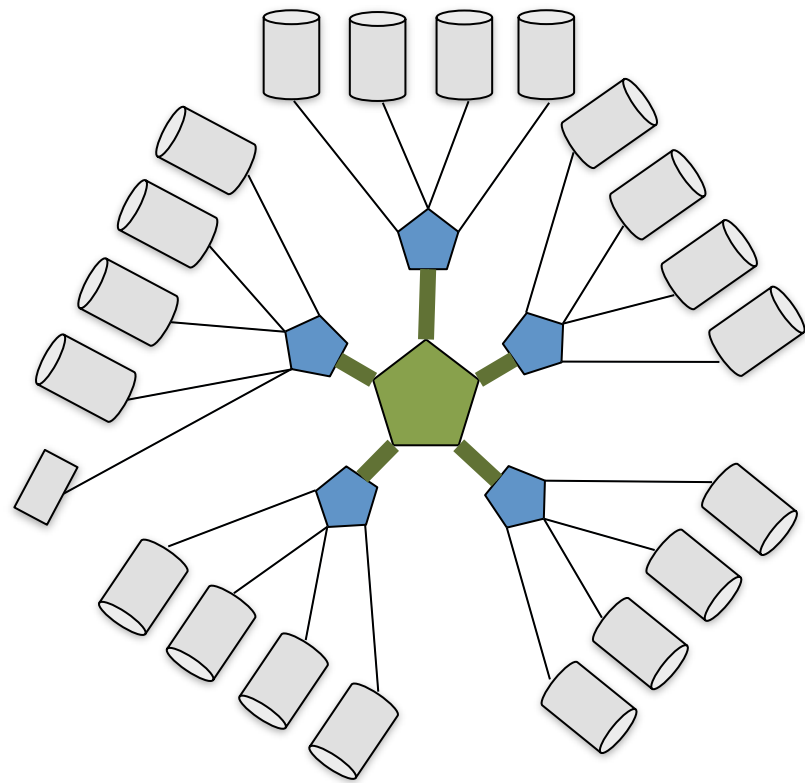


**Note:** such a model can be used to design a convenient allocation protocol or an erasure code (e.g. Partial MDS)



# Erasure Channel with Memory

---



## Modelling disk failures in a DCN:

- can a single disk failure imply the **failure of others**?

[NYGS06]: bi-exponential failure model

$$\text{Prob}(i \text{ failures}) = \alpha f_1(i; p_1) + (1 - \alpha) f_2(i; p_2)$$

with parameters  $\alpha$ ,  $p_1$ ,  $p_2$   
and  $f(i; p) \propto p^i$

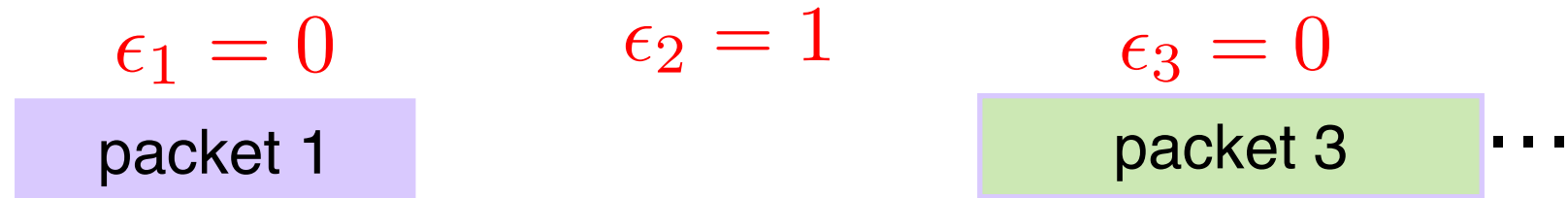
# Block Erasure Channel

---

[JA13]: we propose to consider the block erasure channel model to deal with the following scenarios:

- cluster erasures in DCNs
- codelength  $>$  #disks

Our model:



$n$  coded symbols are divided into  $t$  disjoint packets (not necessarily of the same size), each packet is erased with some probability  $p$

$$\text{Prob}(\epsilon_i = 1) = p$$

# Block Erasure Channel

---

## Our concern:

Given the channel model, to find an **efficient code-allocation scheme**, in terms of repair, storage efficiency and complexity.

# Block Erasure Channel

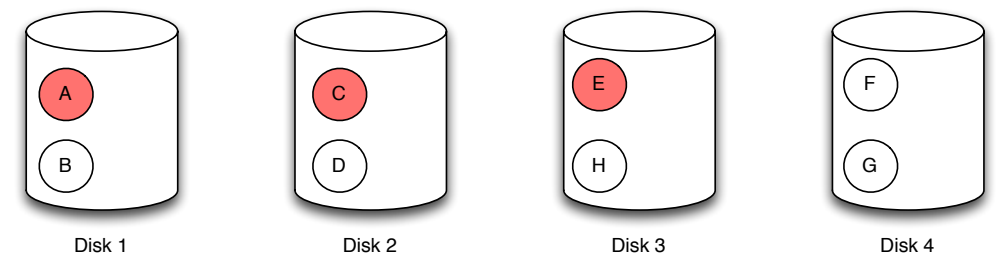
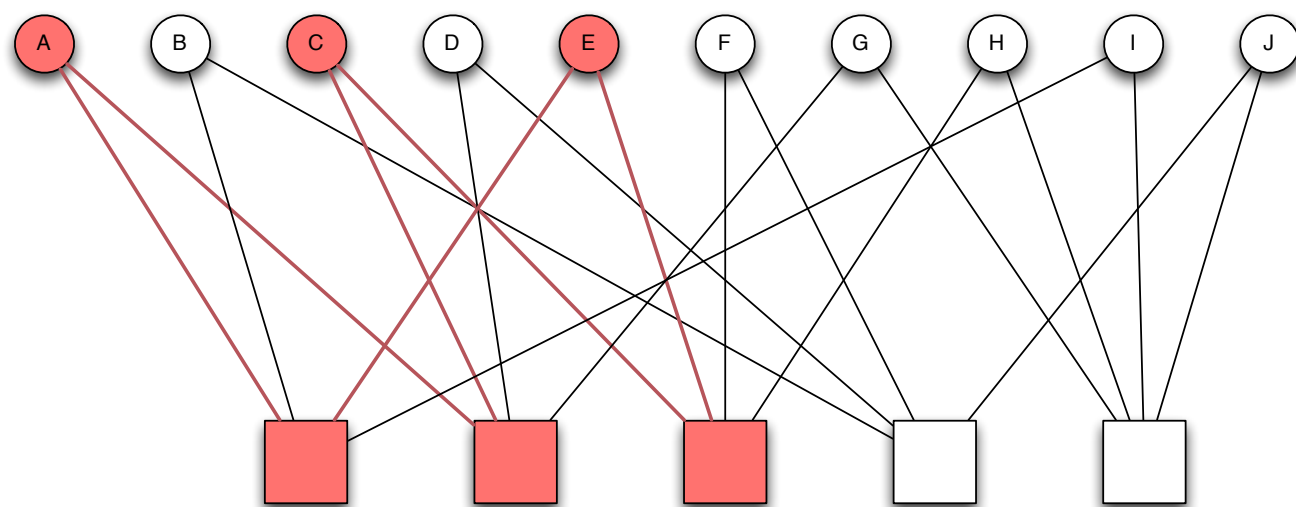
## Our concern:

Given the channel model, to find an **efficient code-allocation scheme**, in terms of repair, storage efficiency and complexity.

## Example of a block code (e.g. LDPC):

the allocation should maximally spread the symbols from small uncorrectable sets (stopping sets);

the code should be designed to have a sufficiently large size of these sets.



**Note:** a similar approach has been proposed to design full-diversity codes over block-erasure channels.

# Block Erasure Channel

---

## Our concern:

Given the channel model, to find an **efficient code-allocation scheme**, in terms of repair, storage efficiency and complexity.

## Example of a block code (e.g. LDPC):

the allocation should maximally spread the symbols from small uncorrectable sets (stopping sets);

the code should be designed to have a sufficiently large size of these sets.

## Example of a code with memory (e.g. convolutional code):

there is a tradeoff between the encoding/decoding complexity, performance and allocation

- maximum spread allocation  $\Rightarrow$  best performance  $\Rightarrow$  highest complexity
- minimum spread allocation  $\Rightarrow$  worst performance  $\Rightarrow$  lowest complexity

# Distributed Storage Allocations with MDS Codes

---

[LDH12]: a particular P2P DSN model with a fixed coding scheme (MDS) and given access model

## DSN model:

- (a) collector has a random (uniform) access to each node
- (b) collector accesses to a random, fixed-size subset of nodes
- (c) source does a probabilistic allocation

## Goal:

To find the **optimal data allocation** over all nodes that maximized the download success probability

## Main result:

Minimum spread allocation - for codes of high rates

Maximum spread allocation - for codes of low rates

# Important Points on the Allocation

---

- \* Some **equivalent channel models** have been proposed, but more is needed.  
Some of models are very close to those existing in the communications theory
- \* An **allocation-based constraint** can be incorporated into the code design
- \* In (almost) all examples of existing DSNs, **the allocation is an issue**, so it should be taken into account  
... especially if one wants to design new efficient storage systems (see the example of flexible storage)

## Part 4: New Problems in Distributed Storage (... for new-generation DSNs? )



# Two Problems

---

1. Storage-flexible DCN
2. Distributed Storage with Data Update

# 1. Storage-Flexible Systems

---

Existing data centers have fixed storage (code) parameters.

Stored data may be subject to changes, e.g:

- \* the stored file became more 'hot' → more accesses to deal with
- \* the stored file became less 'hot' → into the archive
- \* data centers grow/merge
- \* ...

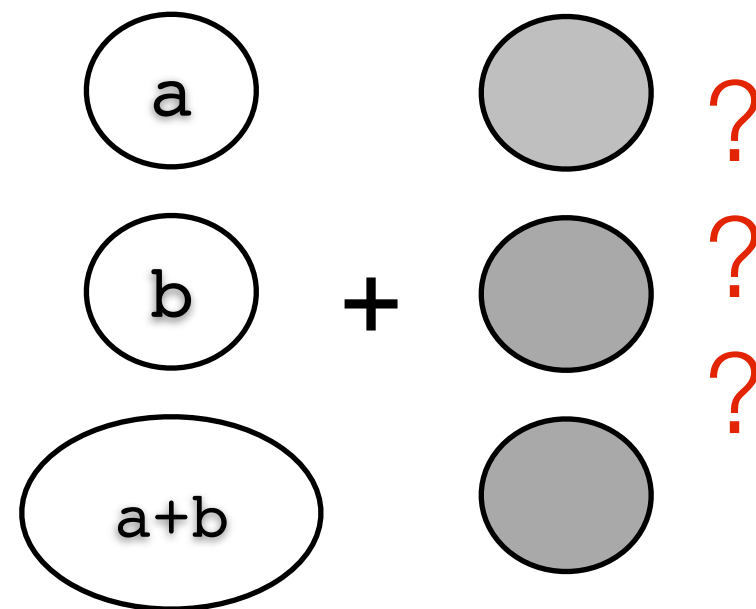
How to design a storage-flexible system? Its cost?

# Storage-Flexible Systems: are those changes indeed needed?

---

Assume the example of the extension of the data center network (DCN):

$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions

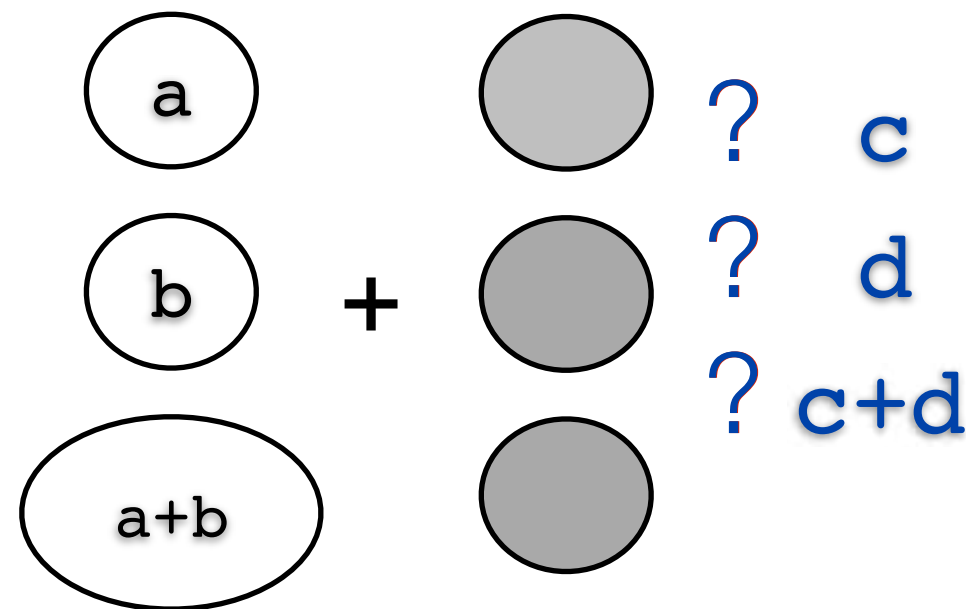


# Storage-Flexible Systems: are those changes indeed needed?

---

Assume the example of the extension of the data center network (DCN):

$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions



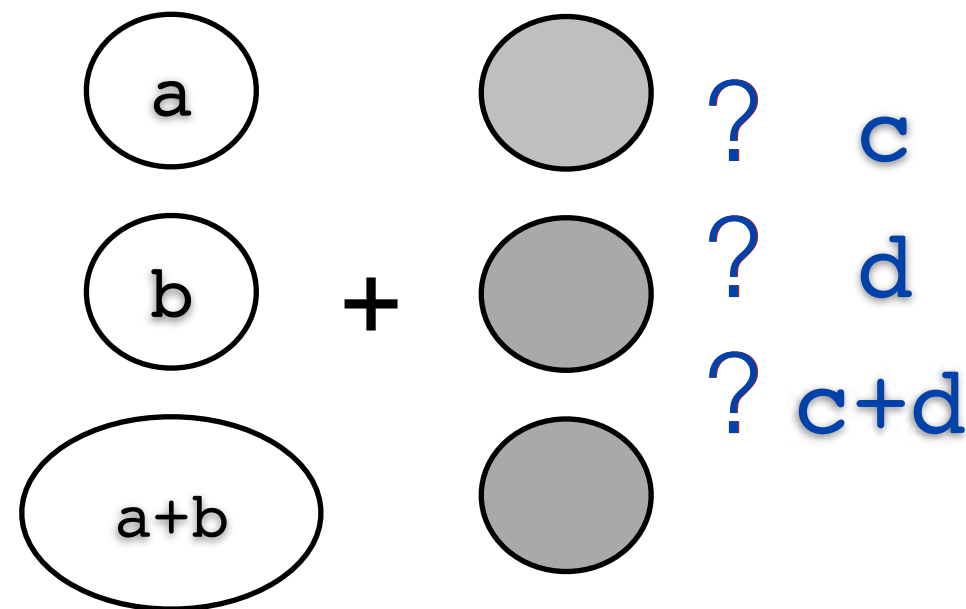
Let's use the same code over 6 disks. Where is the problem?

# Storage-Flexible Systems: are those changes indeed needed?

---

Assume the example of the extension of the data center network (DCN):

$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions



Let's use the same code over 6 disks. Where is the problem?

$[6, 4, 2]$  code is not optimal. One can obtain either  $[6, 4, 3]$  or  $[6, 5, 2]$ .

# Example of the DCN Extension cont'd

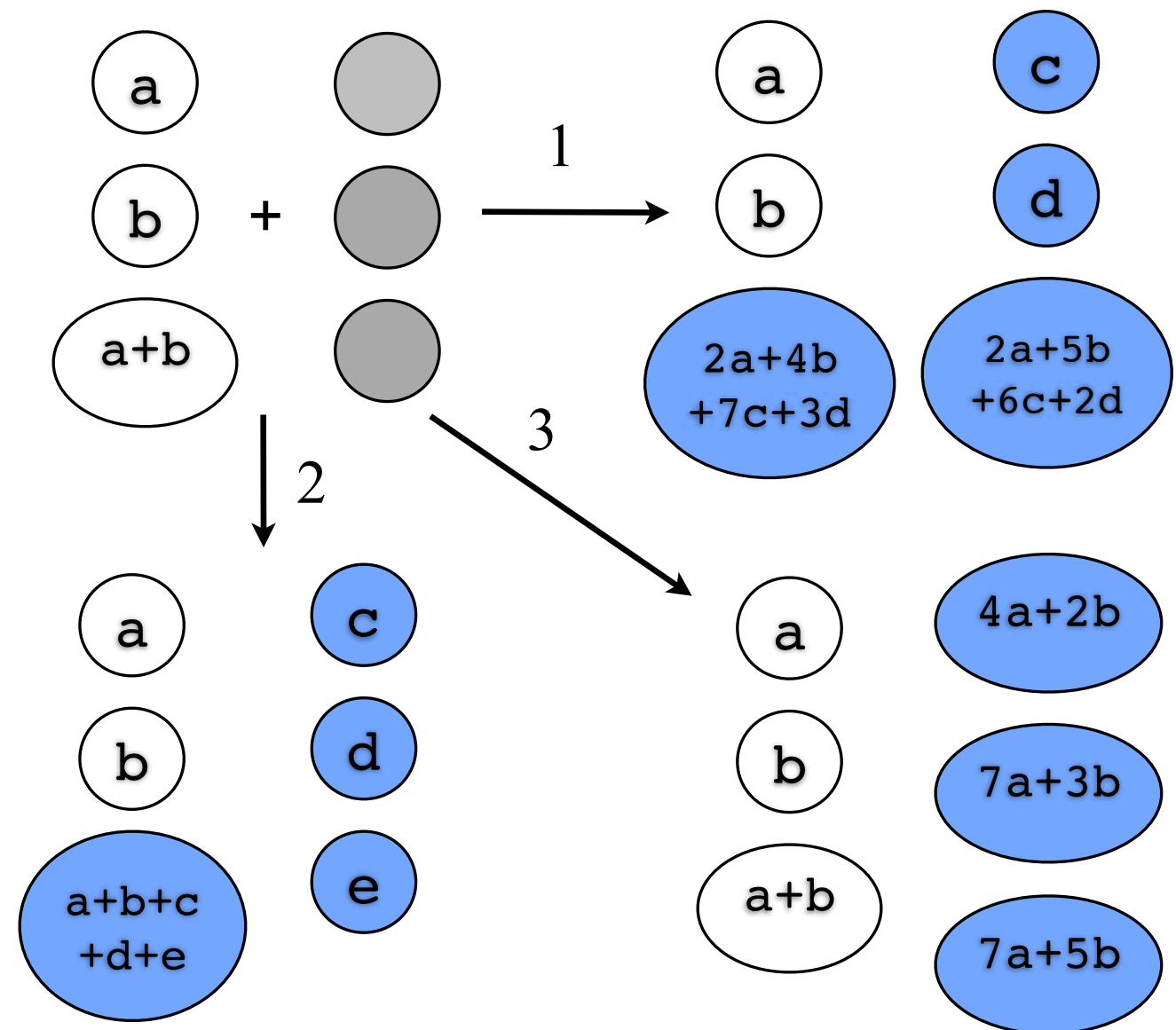
$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions

Possible extensions:

1) keep the rate constant

2) increase the storage volume

3) improve the reliability



# Example of the DCN Extension cont'd

$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions

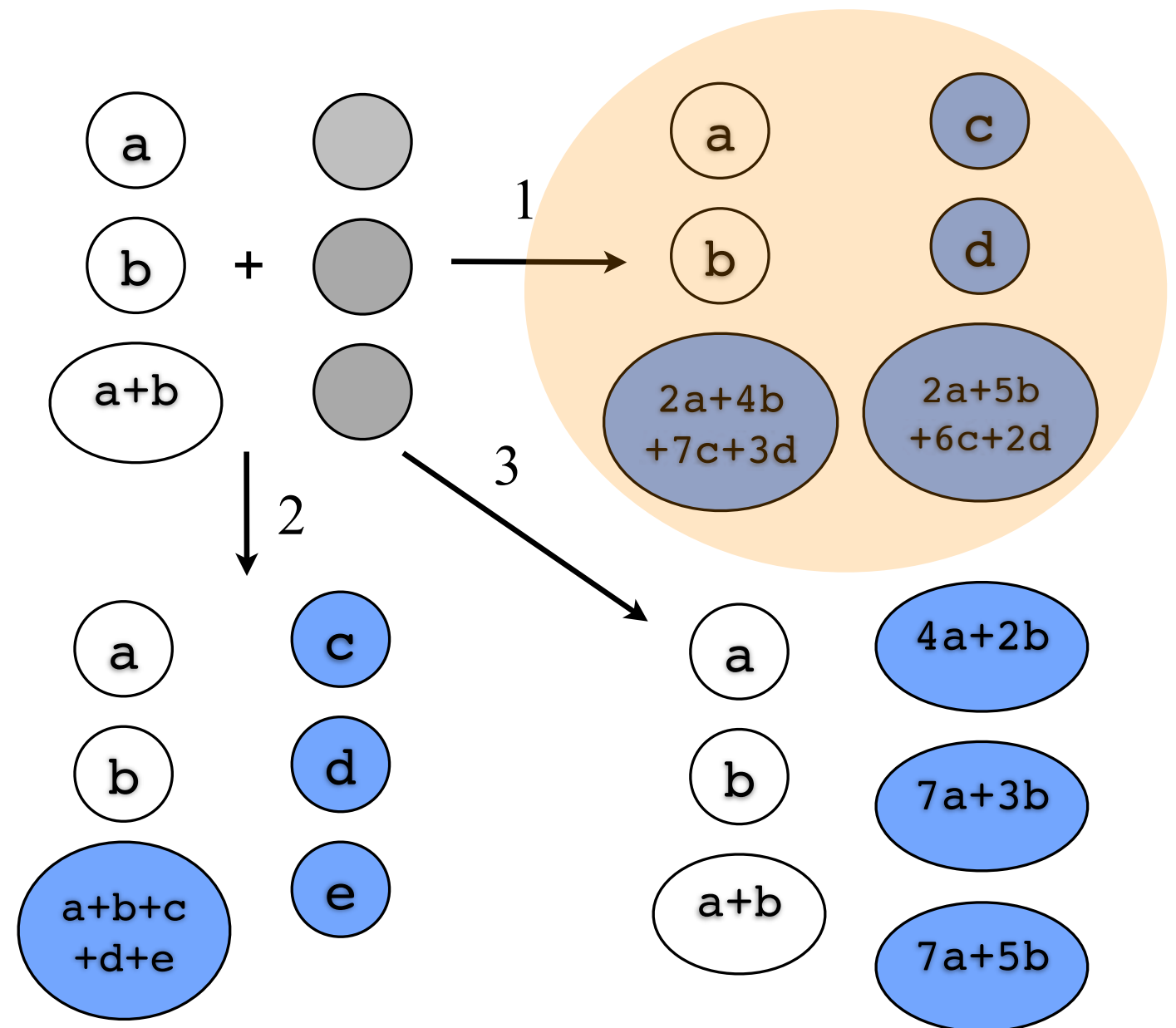
Possible extensions:

1) keep the rate constant

$[6, 4, 3]$

2) increase the storage volume

3) improve the reliability



# Example of the DCN Extension cont'd

$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions

Possible extensions:

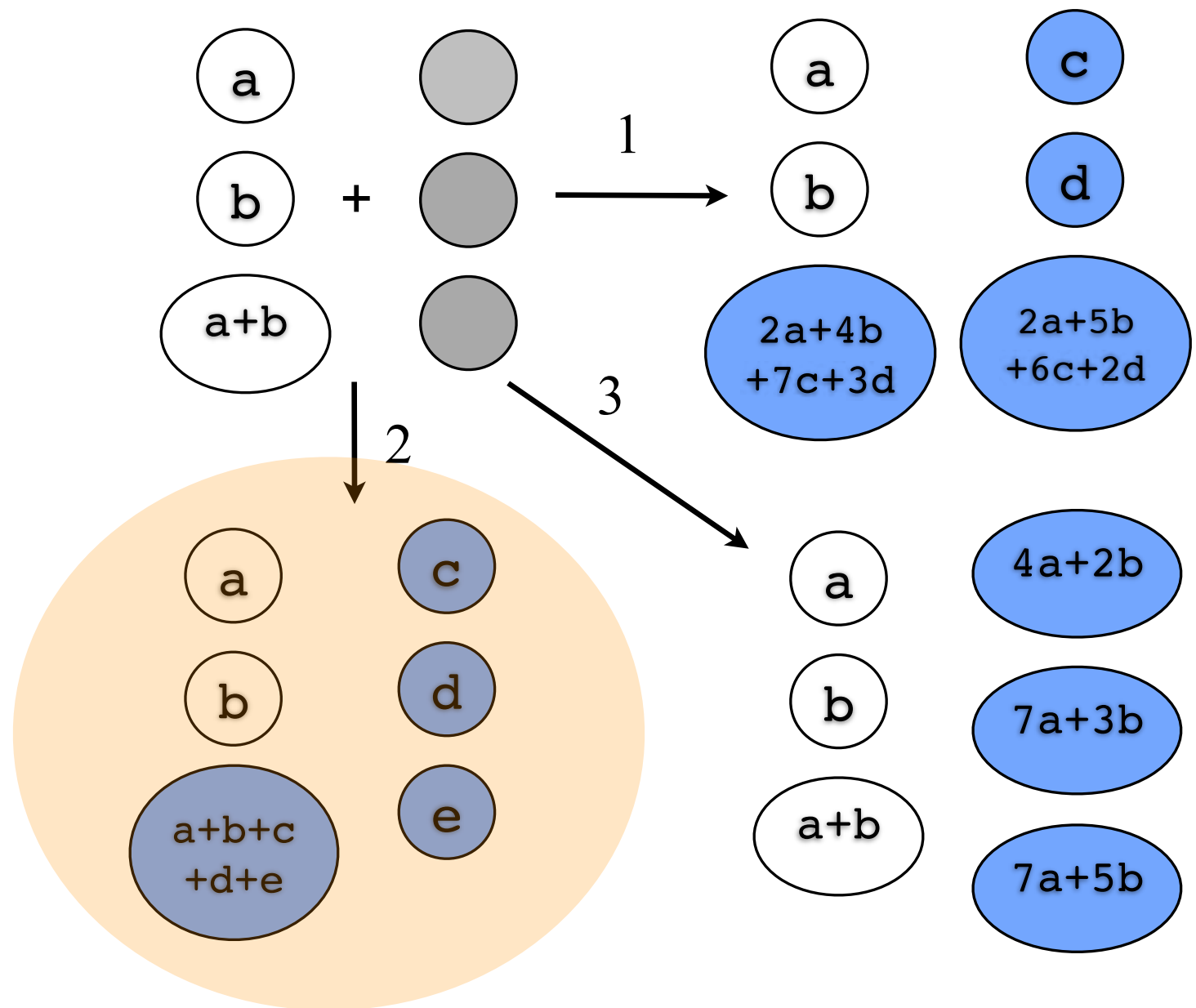
1) keep the rate constant

$[6, 4, 3]$

2) increase the storage volume

$[6, 5, 2]$

3) improve the reliability





# Example of the DCN Extension cont'd

$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions

Possible extensions:

1) keep the rate constant

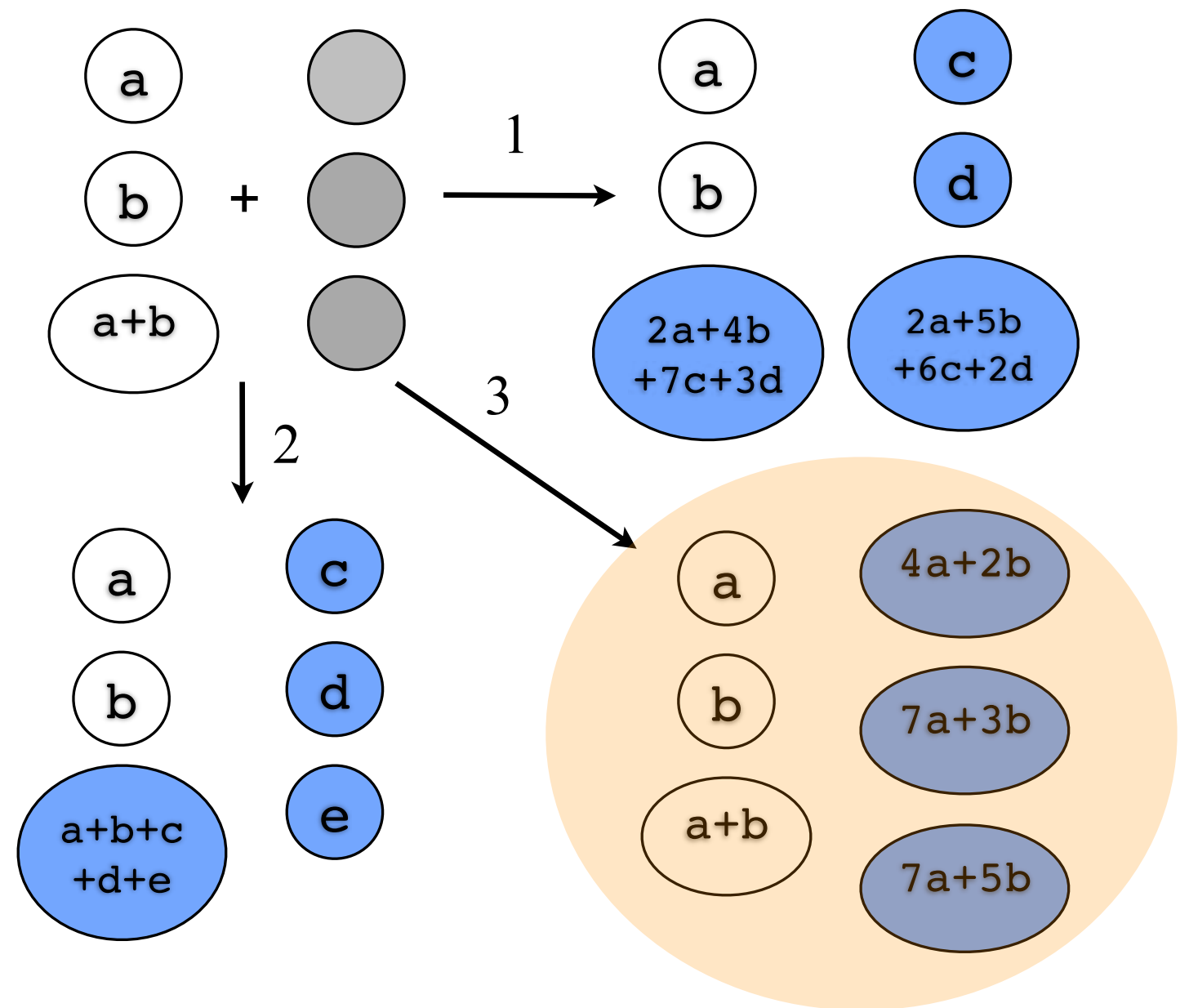
$[6, 4, 3]$

2) increase the storage volume

$[6, 5, 2]$

3) improve the reliability

$[6, 2, 5]$



# Example of the DCN Extension cont'd

$[3, 2, 2]$  MDS code over  $GF(8)$  + 3 new positions

Possible extensions:

1) keep the rate constant

$[6, 4, 3]$

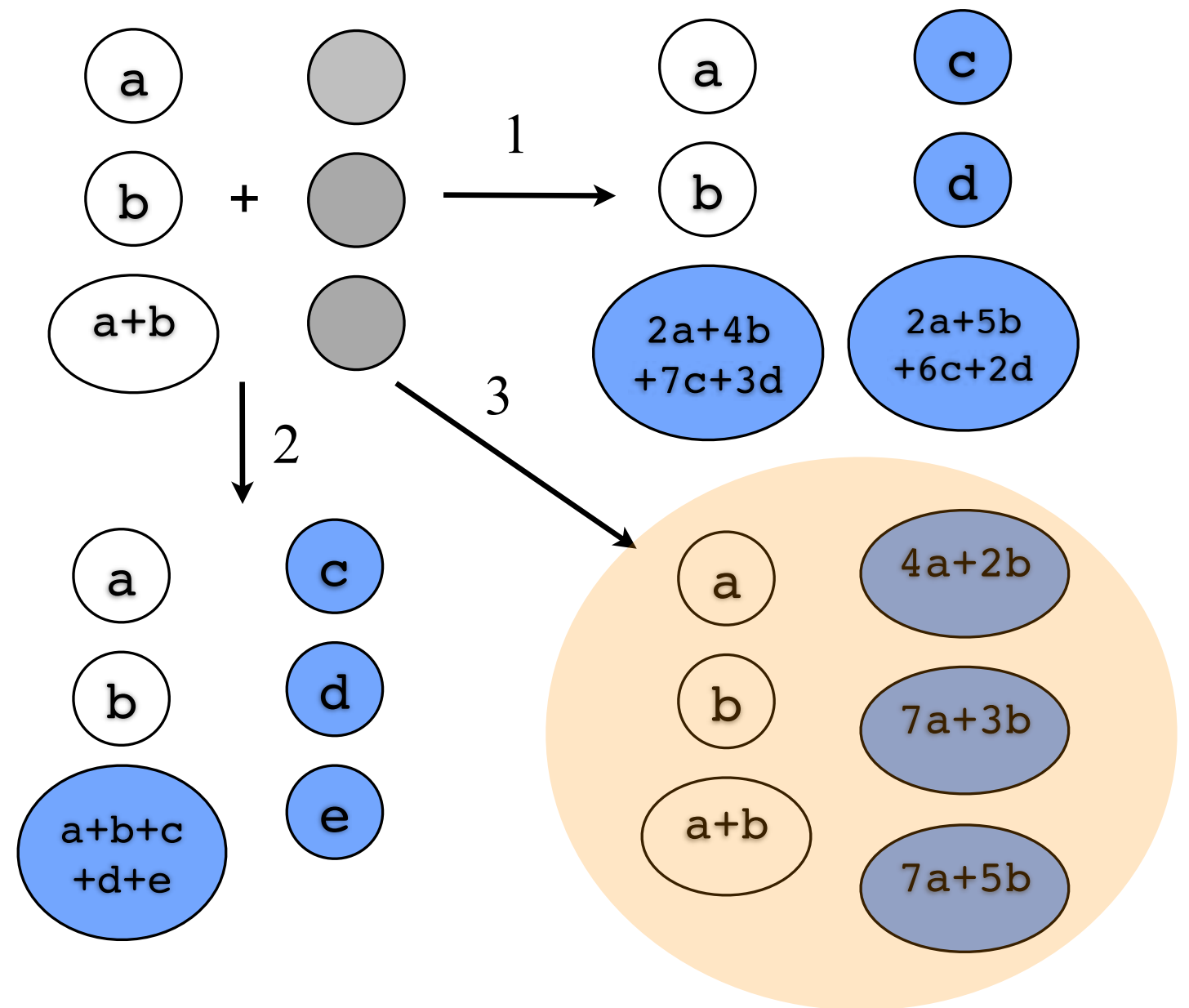
2) increase the storage volume

$[6, 5, 2]$

3) improve the reliability

$[6, 2, 5]$

*Thus, one recovers from 1 to 4 failed disks, and has from 2 to 5 information disks in the DCN.*



# DCN Extension with MDS Codes: General Framework

---

General result from [MS]:

*There exist  $s$  information and  $r$  redundancy columns to add to the parity matrix  $H$ , so that one obtains a  $[n + s + r, k + s, d + r]$  MDS code from a  $[n, k, d]$  MDS code by lengthening. Here one assumes that  $n + s + r \leq q$  (cardinality).*

# DCN Extension with MDS Codes: General Framework

---

General result from [MS]:

*There exist  $s$  information and  $r$  redundancy columns to add to the parity matrix  $H$ , so that one obtains a  $[n + s + r, k + s, d + r]$  MDS code from a  $[n, k, d]$  MDS code by lengthening. Here one assumes that  $n + s + r \leq q$  (cardinality).*

Our data storage example:  $[n_1, k_1, d_1]$  systematic MDS code +  $n_2$  new positions

$$G_1 = (I_{k_1} \quad P)$$

Case 1 :  $[n_1 + n_2, k_1 + R_1 n_2, d_1 + n_2(1 - R_1)]$   
 (no change of rate)

$$G = \left( \begin{array}{cc|c|c} I_{k_1} & \mathbf{0} & P & \\ \mathbf{0} & I_{Rn_2} & P_2 & Q \end{array} \right)$$

# DCN Extension with MDS Codes: General Framework

---

Our data storage example:  $[n_1, k_1, d_1]$  systematic MDS code +  $n_2$  new positions

$$G_1 = (I_{k_1} \quad P)$$

Case 2  
(better storage) :

$$G = \begin{pmatrix} I_{k_1} & \mathbf{0} & P \\ \mathbf{0} & I_{n_2} & P_2 \end{pmatrix}$$

Case 3  
(better reliability) :

$$G = \begin{pmatrix} I_{k_1} & P & P_2 \end{pmatrix}$$

# DCN Extension with MDS Codes :

## Practical Example

---

In (most) existing distributed storage systems, the DCN extension is known as **RAID scaling**.

The DCN size is changed without changing the code parameters:

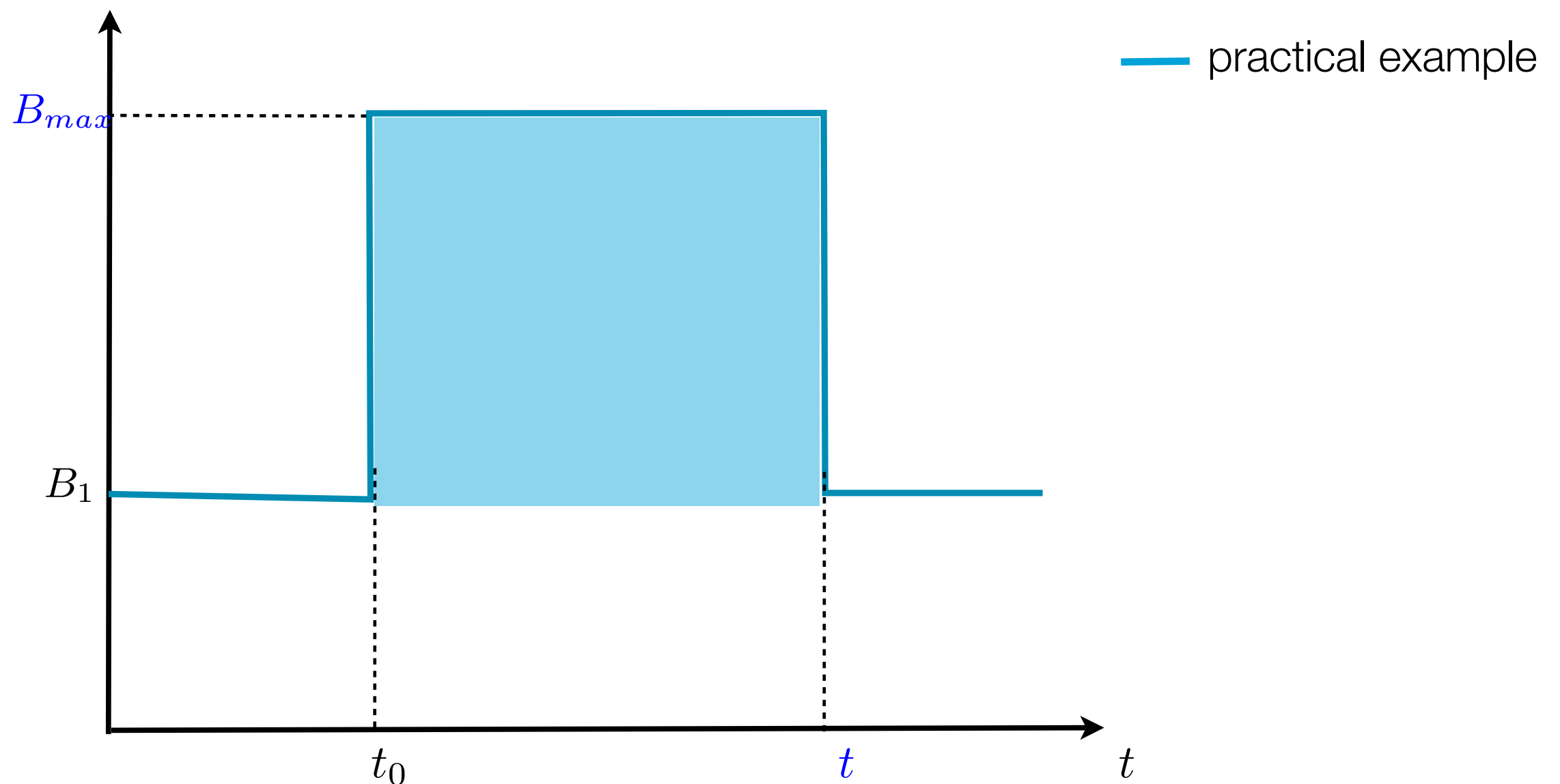
- 1) parity symbols are calculated using old parameters  $n$  and  $k$
- 2) a block placement algorithm moves blocks (symbols) around in order to ensure load balancing and reliability

**Advantage:** — no need to re-code the data already stored

**Drawbacks:** — important network load due of load rebalancing  
— loss of reliability/storage space compared to optimal

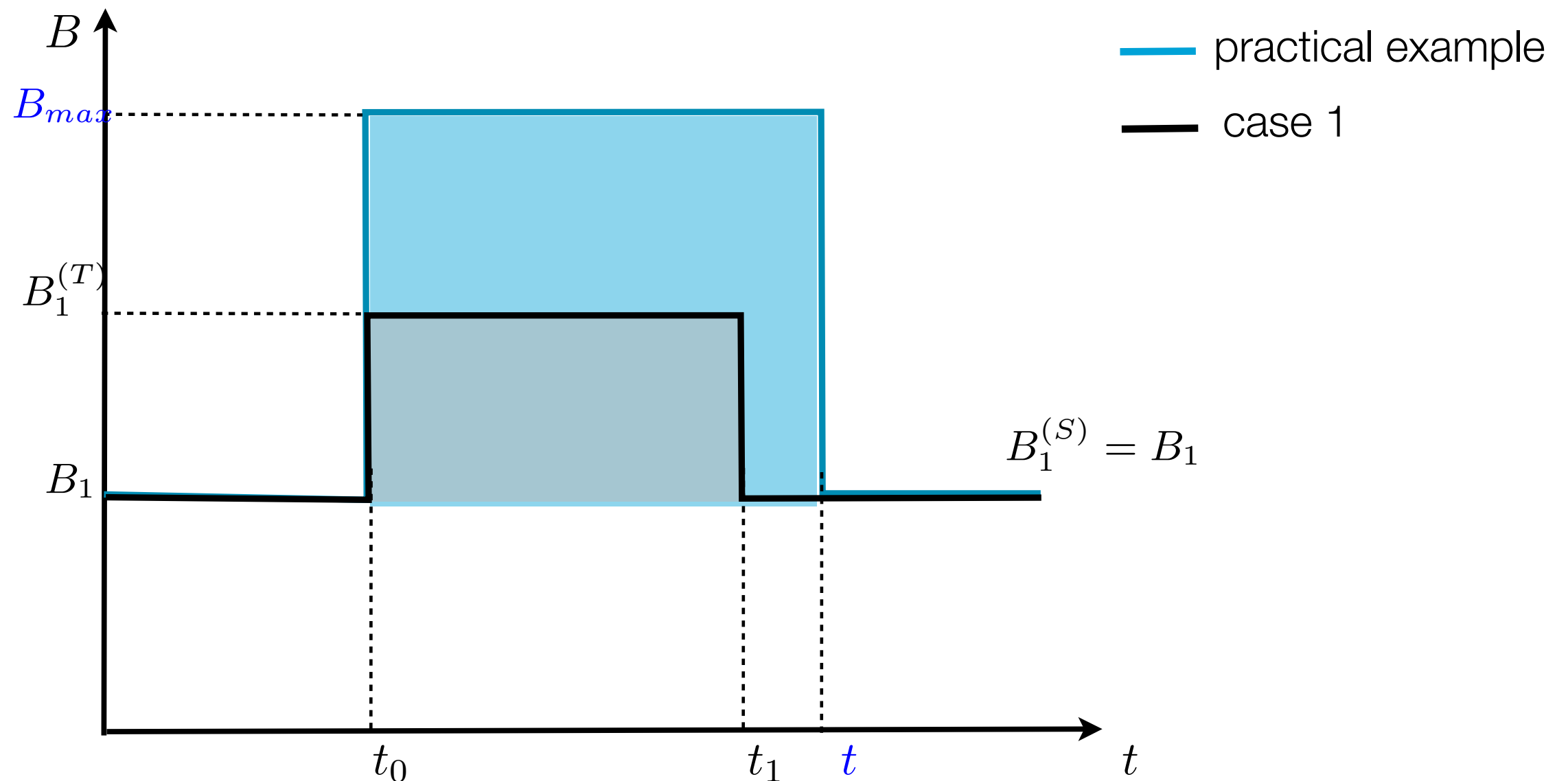
# DCN Extension with MDS Codes: Network Load due to Extension

Let the network maximum bandwidth be  $B_{max}$   
Assume the uploading process of constant rate  $B_1$



# DCN Extension with MDS Codes: Network Load due to Extension

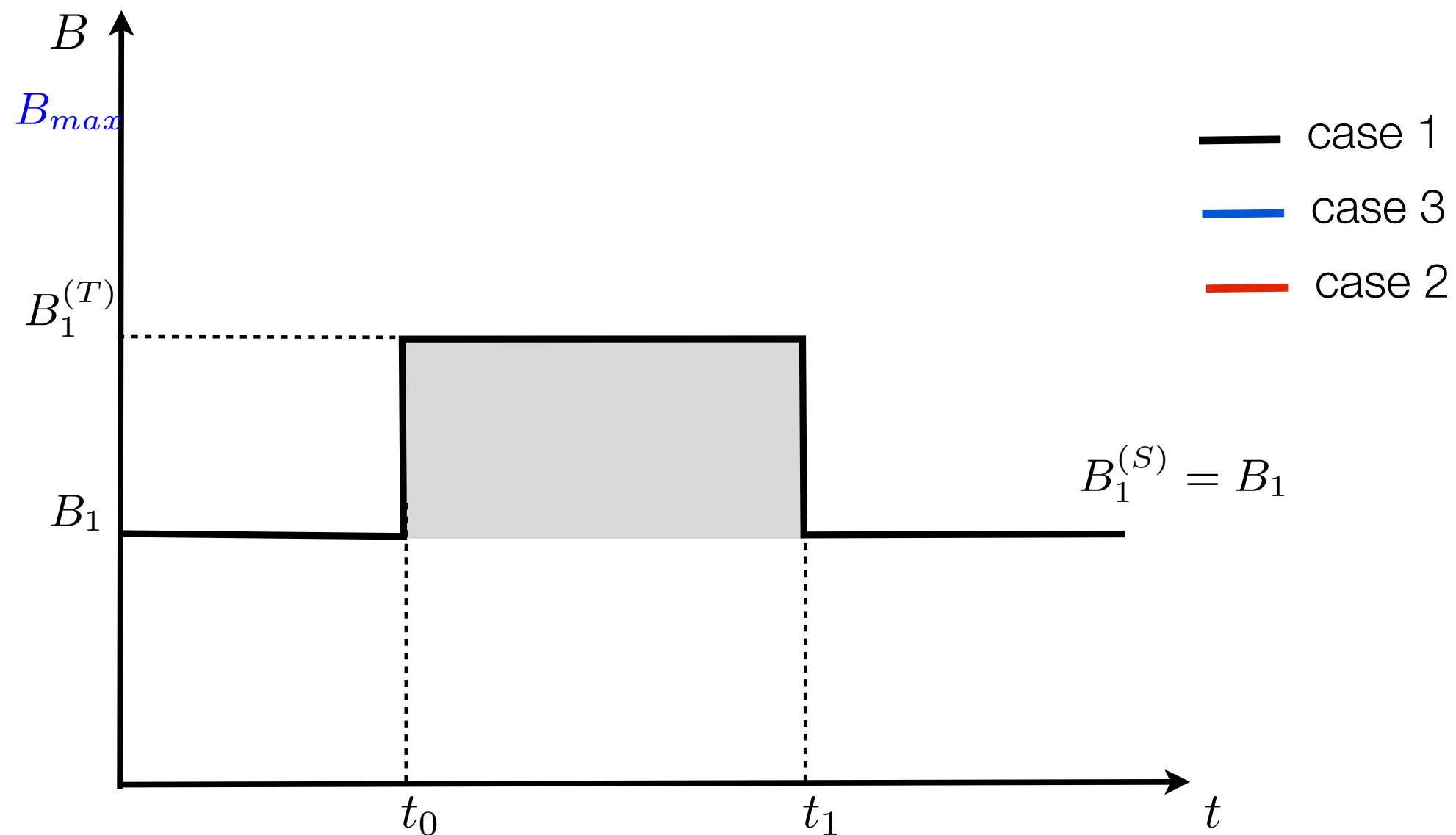
Let the network maximum bandwidth be  $B_{max}$   
Assume the uploading process of constant rate  $B_1$





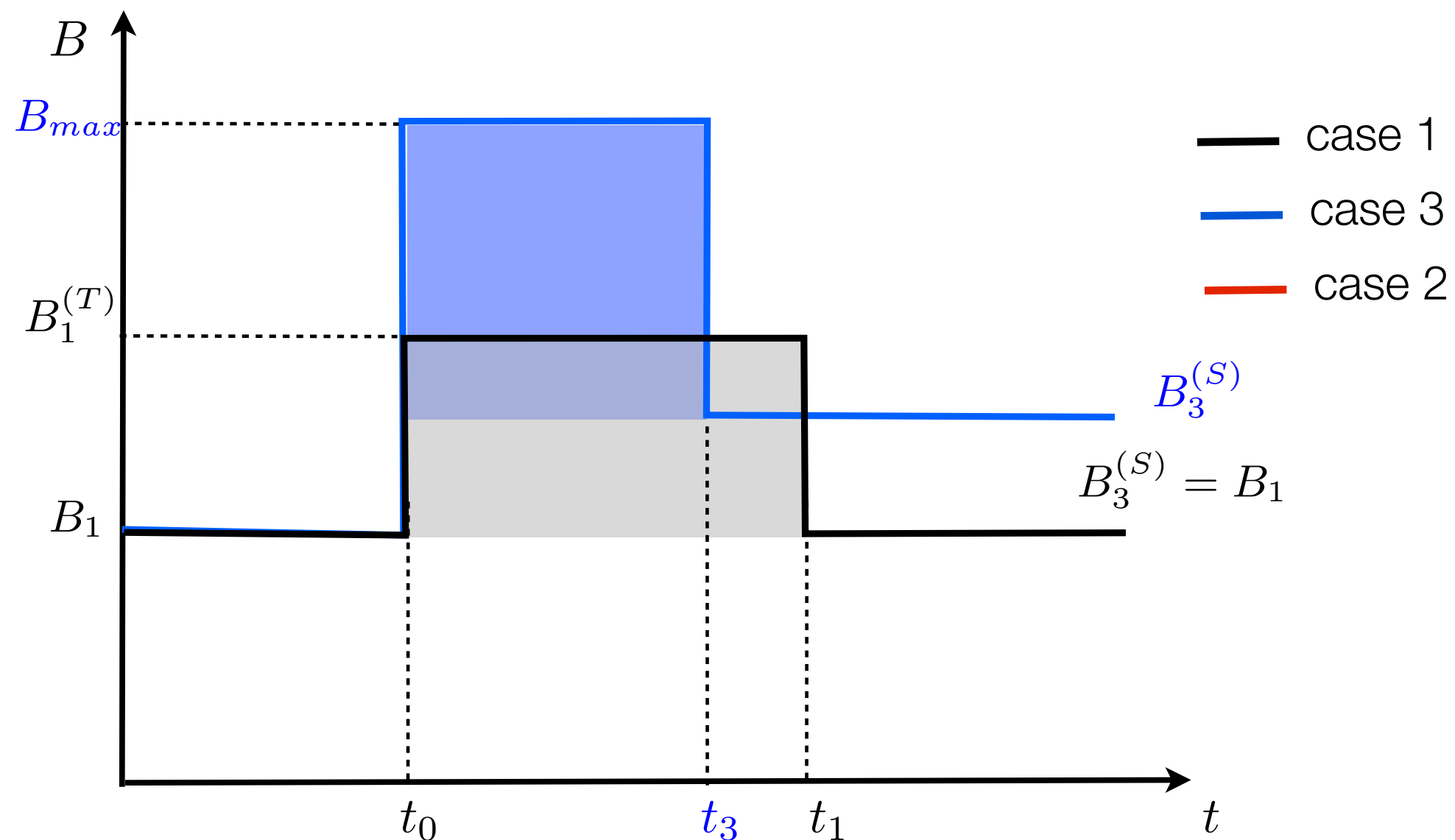
# DCN Extension with MDS Codes: Network Load due to Extension

Let the network maximum bandwidth be  $B_{max}$   
Assume the uploading process of constant rate  $B_1$



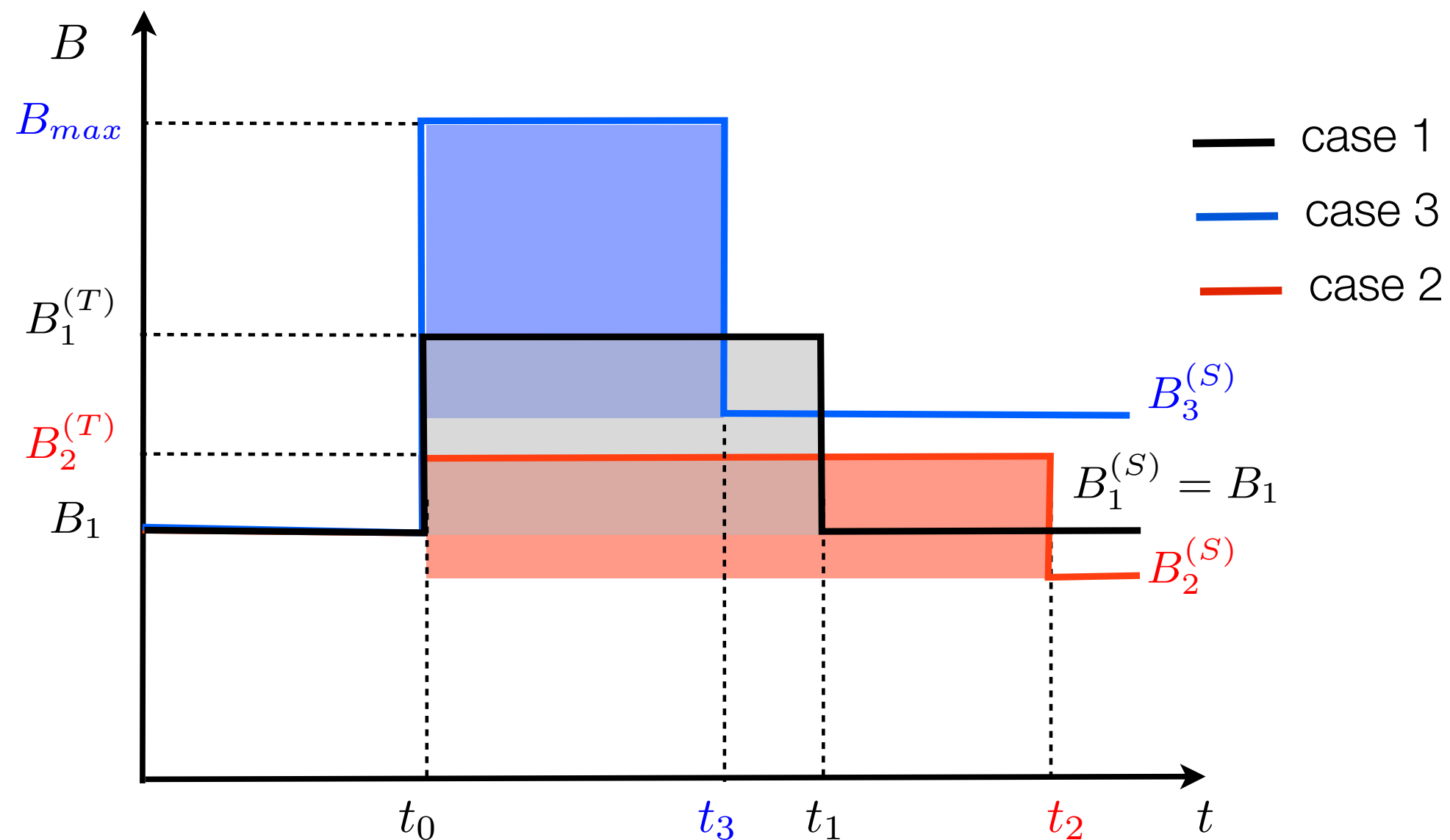
# DCN Extension with MDS Codes: Network Load due to Extension

Let the network maximum bandwidth be  $B_{max}$   
Assume the uploading process of constant rate  $B_1$



# DCN Extension with MDS Codes: Network Load due to Extension

Let the network maximum bandwidth be  $B_{max}$   
Assume the uploading process of constant rate  $B_1$



# DCN Extension with MDS Codes: Network Load due to Extension

## Theorem [AJS13]

Let the data be uploaded to the DCN with the network load of constant rate  $B_1 = \frac{T}{R_1}$ .

Assume the use of  $[n_1, k_1, d_1]$  systematic MDS code.

Let  $t_0$  denote the time instant when the DCN has been extended by  $n_2$  disks.

Then,

$$\mathcal{B}_1 = \begin{cases} B_1, & t < t_0; \\ B_1^{(T)}, & t_0 \leq t < t_1; \\ B_1, & t_1 \leq t; \end{cases}$$

$$\mathcal{B}_2 = \begin{cases} B_1, & t < t_0; \\ B_2^{(T)}, & t_0 \leq t < t_2; \\ B_2^{(S)}, & t_2 \leq t; \end{cases}$$

$$\mathcal{B}_d = \begin{cases} B_1, & t < t_0; \\ B_{max}, & t_0 \leq t < t_3; \\ B_3^{(S)}, & t_3 \leq t; \end{cases}$$

where

$$B_1^{(T)} = B_1 + \frac{1 - R_1}{R_1} \frac{n_1}{n_2} > B_1$$

$$B_2^{(T)} = T \frac{n_1 - k_1 + n_2}{n_2} = B_1 R_1 + (1 - R_1) \frac{n_1}{n_2}$$

$$B_2^{(S)} = T \frac{n}{k_1 + n_2} < B_1$$

$$B_3^{(S)} = T \frac{n}{k_1} > B_1$$

$$t_1 = t_0 + \Delta t_1 = t_0 + \frac{R_1 n_2 m_1 l}{T}$$

$$t_2 = t_0 + \Delta t_2 = t_0 + \frac{n_2 m_1 l}{T}$$

$$t_3 = t_0 + \Delta t_3 = t_0 + \frac{k_1 n_2 \cdot m_1 l}{(k_1 + n_2)(B_{max} - B_3^{(S)})}$$

# DCN Extension with MDS Codes: Network Load due to Extension

---

Downloading and repair stationary processes:

| Parameters              | $C_1$                | $C_3$                       | $C_2$                 |
|-------------------------|----------------------|-----------------------------|-----------------------|
| Rate $R$                | $R_1$                | $\frac{k_1 + n_2}{n} > R_1$ | $\frac{k_1}{n} < R_1$ |
| Minimum distance $d$    | $d_1 + n_1(1 - R_1)$ | $d_1$                       | $d_1 + n_2$           |
| Downloading load $D(k)$ | $k$                  | $k$                         | $k$                   |
| Repair load $R(1)$      | $k_1 + R_1 n_2$      | $k_1 + n_2$                 | $k_1$                 |

# DCN Extension with MDS Codes :

## Remarks

---

Change of  $n$  and  $k$  impact on:

### **Storage parameters**

- \* Failure protection:  $\downarrow$  when  $R \uparrow$
- \* Storage efficiency:  $\uparrow$  when  $R \uparrow$

### **Network parameters**

- \* Stationary network loads:  $\downarrow$  when  $R \uparrow$
- \* Transition network loads:
  - the smallest for the optimal code solution
  - rebalancing load  $\downarrow$  when  $R \uparrow$
  - transition time  $\uparrow$  when  $R \uparrow$
  - in general, the total load  $\downarrow$  when  $R \uparrow$

# Storage-Flexible Systems: Beyond MDS?

---

Is it possible to get better network parameters?

# Storage-Flexible Systems: Beyond MDS?

---

Is it possible to get better network parameters?

**Yes.**

Use codes with a good **locality** → less operations/network accesses related to parity symbols

Example:

staircase quasi-cyclic non-binary LDPC codes [AJS13]

piggyback construction ...

**Advantage:** — improvement of network loads for lower code rates  
(cases 1 and 3)

**Drawback:** — careful design of the code (non-MDS)  
— possible loss of coding-allocation independence, hence design of an allocation scheme



# Storage-Flexible Systems: Important Points

---

- \* Load rebalancing process should be also considered (in addition to upload, download, repair)
- \* In practical DCNs, the impact of the allocation protocol is huge (even for MDS codes!)
- \* Network loads are code-dependent  
new code structure = new calculation?
- \* Efficient code schemes are needed  
non-MDS but with some local MDS properties?

## 2. Distributed Storage with Data Update

---

Existing DSNs do not allow the update of the data already stored.

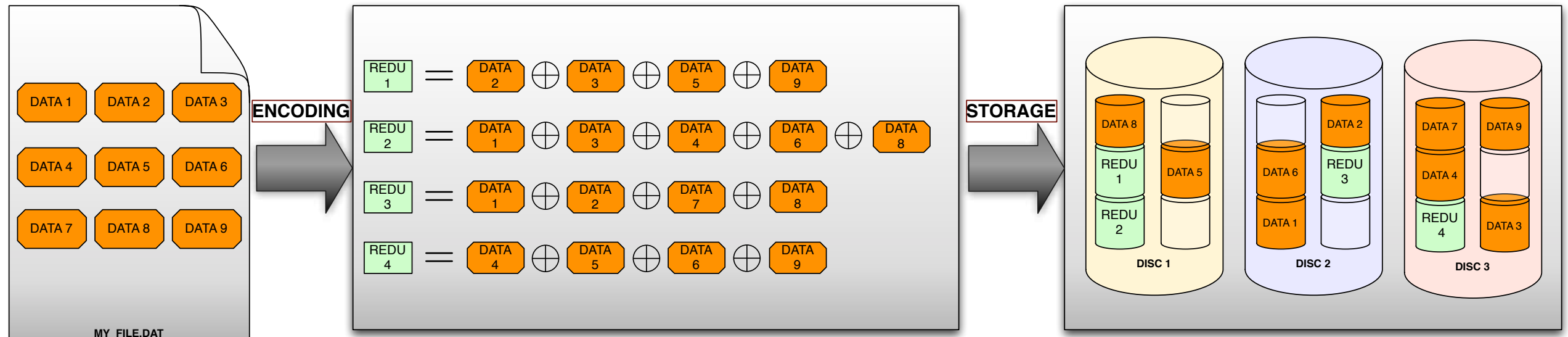
### Dynamical DS system:

- \* the **rewrites** of the stored data are allowed
- \* one wants to minimize the update repair: tradeoff between update repair and other DSN parameters

Is it possible to design an update-efficient DSN?

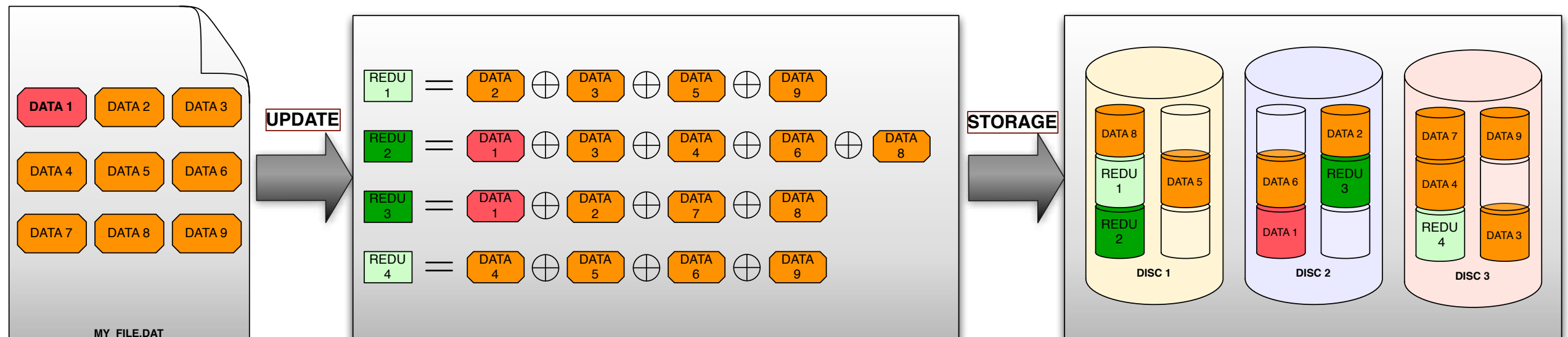
# Illustration on the Data Update

## 1 - DISTRIBUTED STORAGE



## 2 - UPDATE OF DATA 1

\* **REDU** : REDUNDANCY



# Update Complexity

---

An update-efficient DSN should be based on an update-efficient code.

## Definitions:

The **average update complexity**  $\gamma_{avg}$  is the average number of symbol updates (rewrites), made when one information symbol is changed.

The **maximum update complexity**  $\gamma_{max}$  is the maximum number of symbol updates, made when one information symbol is changed.

[ASV10] N.P. Anthapadmanabhan, E. Soljanin, and S. Vishwanath. Update-efficient codes for erasure correction. In *Proceedings of Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 2010.

# Update Complexity

---

An update-efficient DSN should be based on an update-efficient code.

## Definitions:

The **average update complexity**  $\gamma_{avg}$  is the average number of symbol updates (rewrites), made when one information symbol is changed.

The **maximum update complexity**  $\gamma_{max}$  is the maximum number of symbol updates, made when one information symbol is changed.

[ASV10] N.P. Anthapadmanabhan, E. Soljanin, and S. Vishwanath. Update-efficient codes for erasure correction. In *Proceedings of Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 2010.

## *Definition*

*A code is update-efficient if its update complexity is  $o(n)$ .*

## Main tradeoff:

update complexity versus fault tolerance (minimum distance)

# Update Complexity vs Minimum Distance

---

| Case                      | Examples of codes         | Lower bound on $\gamma$ | Upper bound on $\gamma$ | Update efficient ? |
|---------------------------|---------------------------|-------------------------|-------------------------|--------------------|
| $d_{min} = O(n)$          | Reed-Solomon, (3, 6) LDPC | $O(n)$                  | $O(n)$                  | <b>no</b>          |
| $d_{min} = O(n^\alpha)^*$ | Repeat-Accumulate         | $O(n^\alpha)$           | $O(n)$                  | <b>?</b>           |
| $d_{min} = O(\log n)$     | Turbocodes, (2, 4) LDPC   | $O(\log n)$             | $O(n)$                  | <b>?</b>           |
| $d_{min} = O(1)$          | LDGM codes                | $O(1)$                  | $O(1)$                  | <b>yes</b>         |

\*  $\alpha < 1$

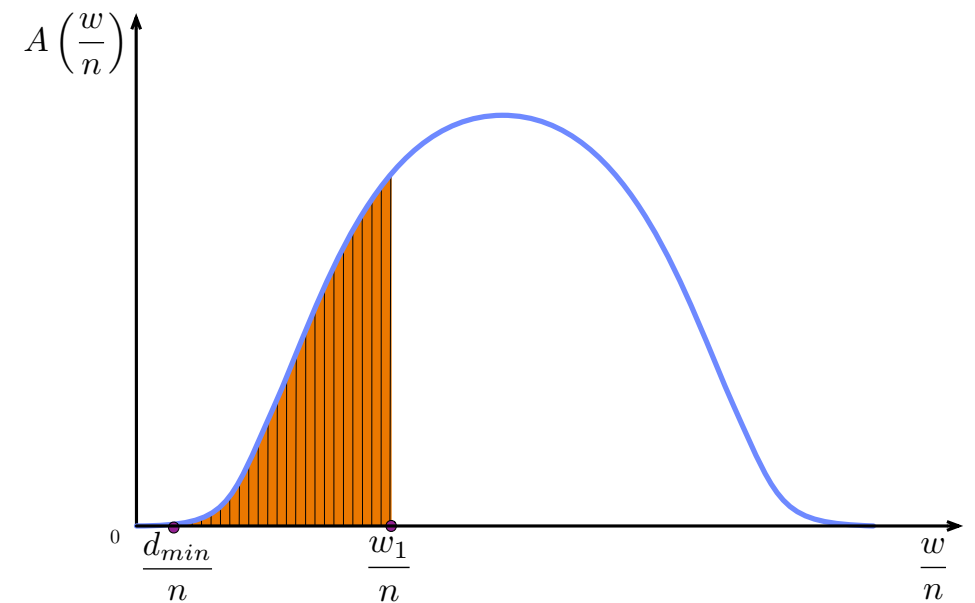
# Update Complexity vs Minimum Distance

Average weight distribution  $\{A(i)\}$  is used to prove:

## Theorem 1: [Tight lower bound] :

If there exists  $w_1$  such that  $\sum_{i=d_{min}}^{w_1} A(i) \leq k$ , then

$$\begin{aligned} \gamma_{max} &\geq w_1 \\ \gamma_{avg} &\geq \frac{1}{k} \sum_{i=d_{min}}^{w_1} iA(i) \end{aligned}$$



$\Rightarrow$  **Codes with  $d_{min} = O(n^\alpha)$  are not update efficient**

...and more generally, **a code is update efficient if  $d=O(\log n)$**   
(and under some supplementary conditions)

# Update-Efficient DS Systems: Important Points

---

- \* **Update-efficiency** is incompatible with a good fault tolerance (of order  $O(n)$ )
- \* Clearly, the storage of **frequently modified data** will take more storage space than the storage of archival data
- \* Example of good storage-efficient codes: Reed-Solomon, example of good update-efficient codes: LDGM, a **tradeoff** between the two: codes with logarithmic minimum distance, e.g. some LDPC



# Final Discussion

---

- \* **Storage-repair tradeoff**

- for modified **MDS codes** via the generalisation of the Singleton bound (access tradeoff)
- can be studied for some **P2P DSNs** by using the information-theoretic approach (bandwidth tradeoff)

- \* **Repair access** tradeoff is only known for some particular codes.

- **LRC codes** are a good example
- **sparse-graph codes**: efficient but very careful design

- \* **Modelling failures and collector accesses** in DSNs

- design of allocation patterns/codes robust to **burst erasures**
- **optimal asymptotic allocations for P2P DSNs**

- \* **Storage-flexible** DCNs

- **load rebalancing** parameter

- \* **Update-efficiency**

- **tradeoff** vs. fault tolerance

# Final Discussion

---

\* New **efficient code-allocation schemes** are needed for present and future DSNs

- good performance and of low complexity
- flexible in parameters  $k$ ,  $d$  and  $n$
- for use to store large amounts of data
- for some cases, compatible with already deployed codes

\* **Fundamental performance tradeoffs** should be better understood:

- unfortunately, they are usually code-, network- or data dependent:  
is it possible to formulate a general framework? several separate models?
- good approximations of difficult cases
- use of already existing tools in combinatorics, information theory, communications, coding