Distributed Storage Systems

Reliable Storage Continued:
Regenerating Codes & Local Repairability

Agenda

Reliable storage

Today's topics

- Exact repair and regenerating codes
- Local repair

Goals

This week's Learning Goals

- Understand basics of regenerating codes
- Understand local repairability

Class Structure

	Lecture	Lab
Week 1	Course introduction, networking basics, socket programming	Python sockets
Week 2	RPC, NFS, Practical RPC	Flask, JsonRPC, REST API
Week 3	AFS, reliable storage introduction	ZeroMQ, ProtoBuf
Week 4	Hard drives, RAID levels	RPi stack intro, RPi RAID with ZMQ
Week 5	Finite fields, Reed-Solomon Codes	Kodo intro, RS and RLNC with Kodo
Week 6	Repair problem, RS vs Regenerating codes	RPi simple distributed storage with Kodo RS
Week 7	**Regenerating codes, XORBAS**	**RPi Regenerate lost fragments with RS**
Week 8	Hadoop	RPi RLNC, recovery with recode
Week 9	Storage Virtualization, Network Attached Storage, Storage Area Networks	RPi basic HDFS (namenode+datanode, read and write pipeline)
Week 10	Object Storage	RPi basic S3 API
Week 11	Compression, Delta Encoding	Mini project consultation
Week 12	Data Deduplication	RPi Dedup
Week 13	Fog storage	Mini project consultation
Week 14	Security for Storage Systems and Recap	Mini project consultation

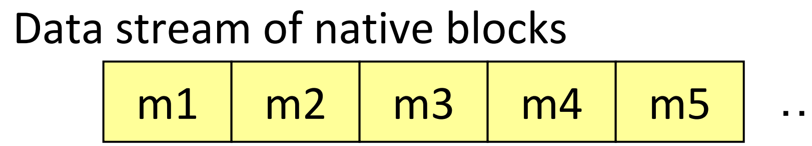
Regenerating Codes & Exact Repair

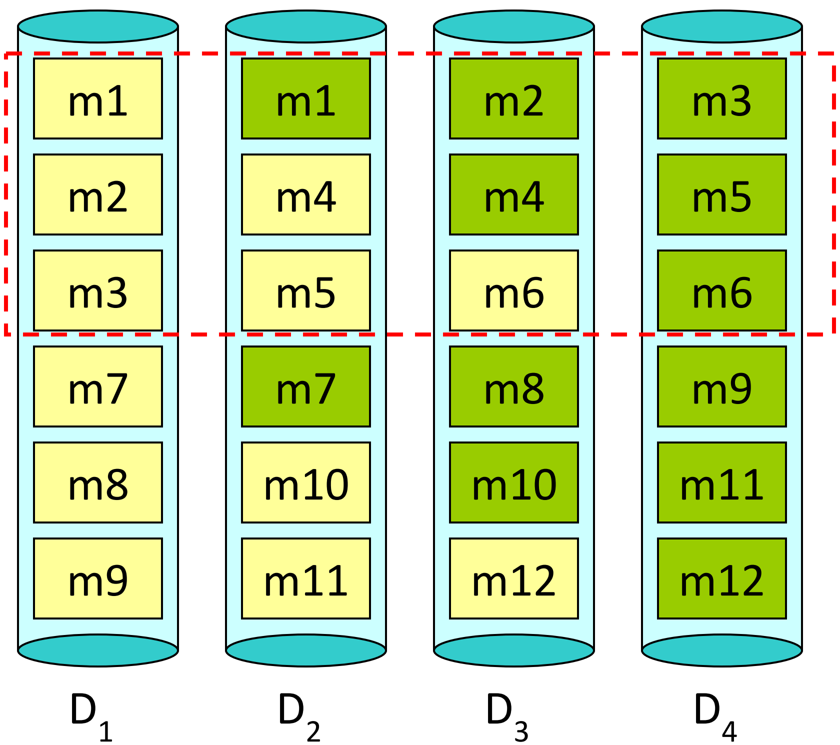
- E-MBR (Minimum bandwidth regenerating) codes with exact repair
  - Minimize repair bandwidth while maintaining MDS property
- Idea:
  - Assume d = n – 1 (repair data from n – 1 survival disks during a single-node failure)
  - Make a duplicate copy for each native/code block
  - To repair a failed disk, download exactly one block per segment from each survival disk

Rashmi et al., "Explicit construction of optimal exact regenerating codes for distributed storage". Allerton 2009.

Regenerating Codes

- E-MBR(n, k=n-1, d=n-1)

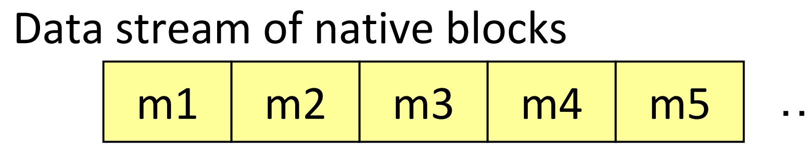


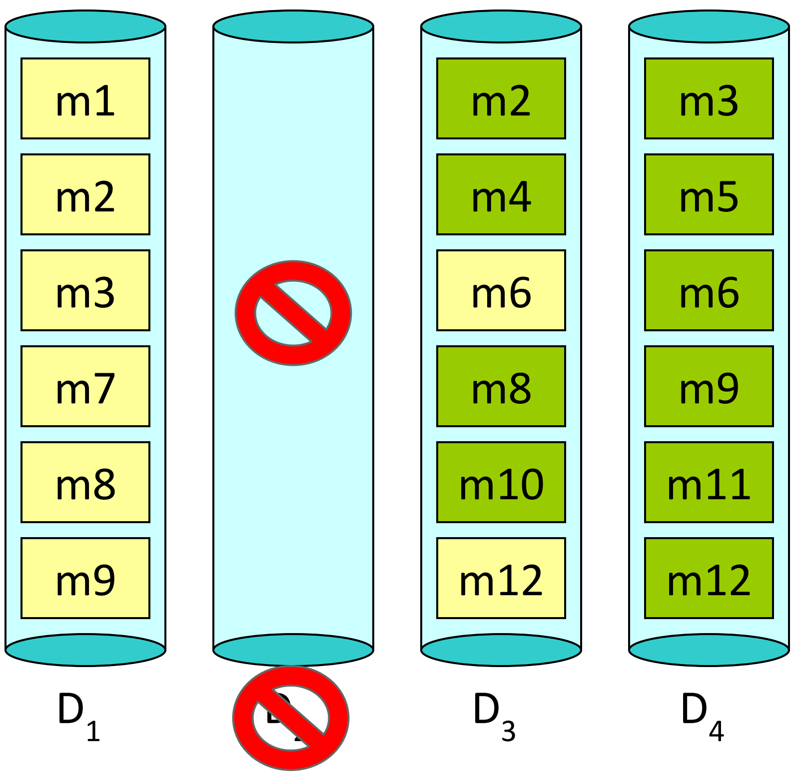


- ➢ Duplicate each native block. Both native and duplicate blocks are in different disks
- ➢ Parameters:
  - n = can be any ≥ 2
  - k = n - 1
  - d = n - 1
  - m = n(n-1)/2
  - c = 0 (i.e., no code block)

Regenerating Codes

- E-MBR(n, k=n-1, d=n-1)





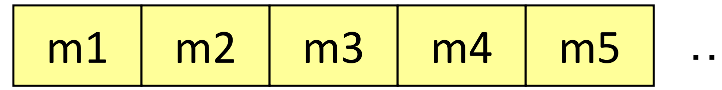
➢ Duplicate each native block. Both native and duplicate blocks are in different disks

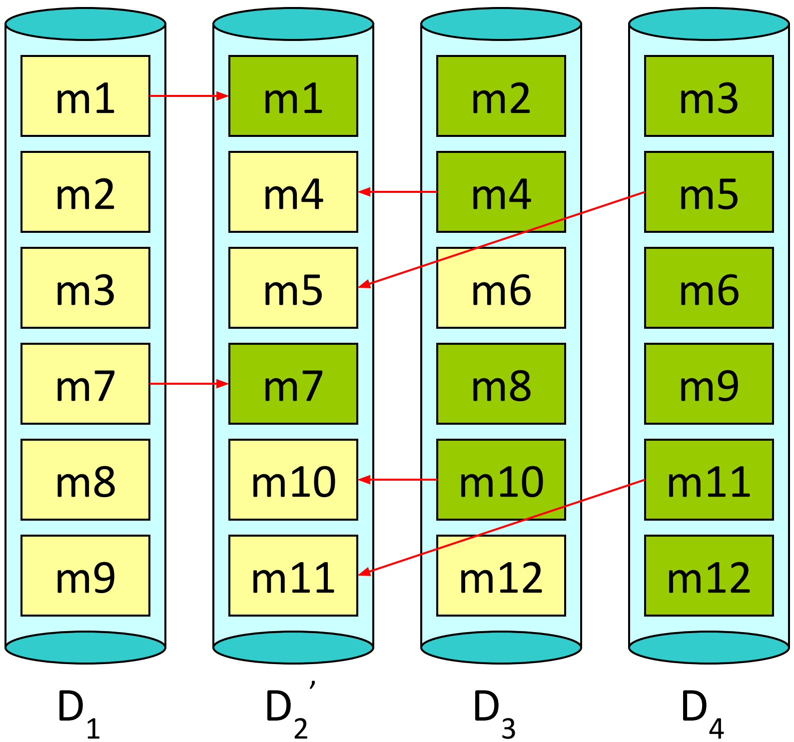
➢ Parameters:
  - n = can be any ≥ 2
  - k = n - 1
  - d = n - 1
  - m = n(n-1)/2
  - c = 0 (i.e., no code block)

Regenerating Codes

- E-MBR(n, k=n-1, d=n-1)

Data stream of native blocks





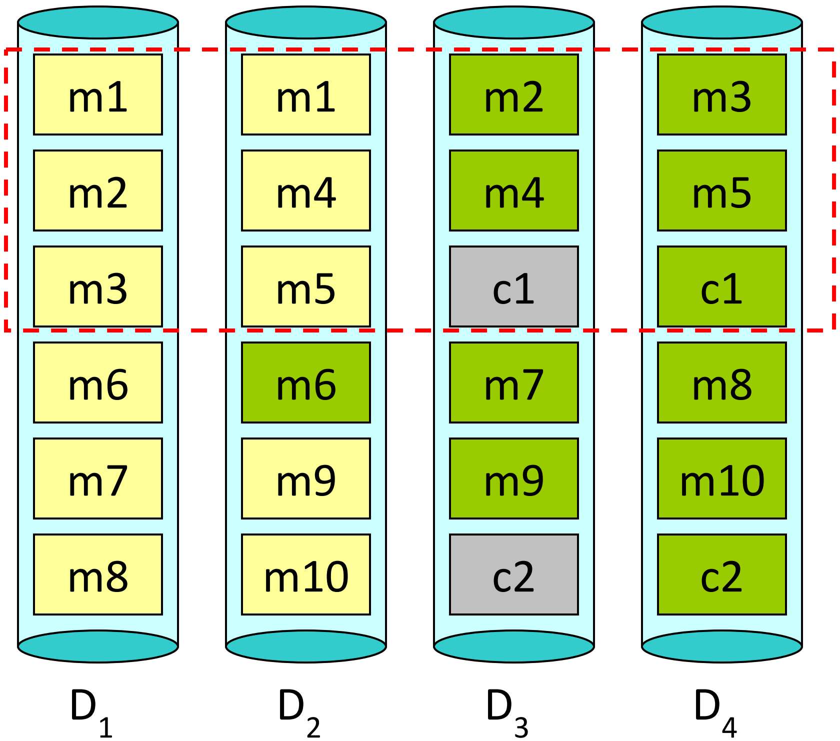
➢ Duplicate each native block. Both native and duplicate blocks are in different disks

➢ Parameters:
  - n = can be any ≥ 2
  - k = n - 1
  - **d = n - 1**
  - m = n(n-1)/2
  - c = 0 (i.e., no code block)

Regenerating Codes

Data stream of native blocks

- E-MBR(n, k=n-2, d=n-1)

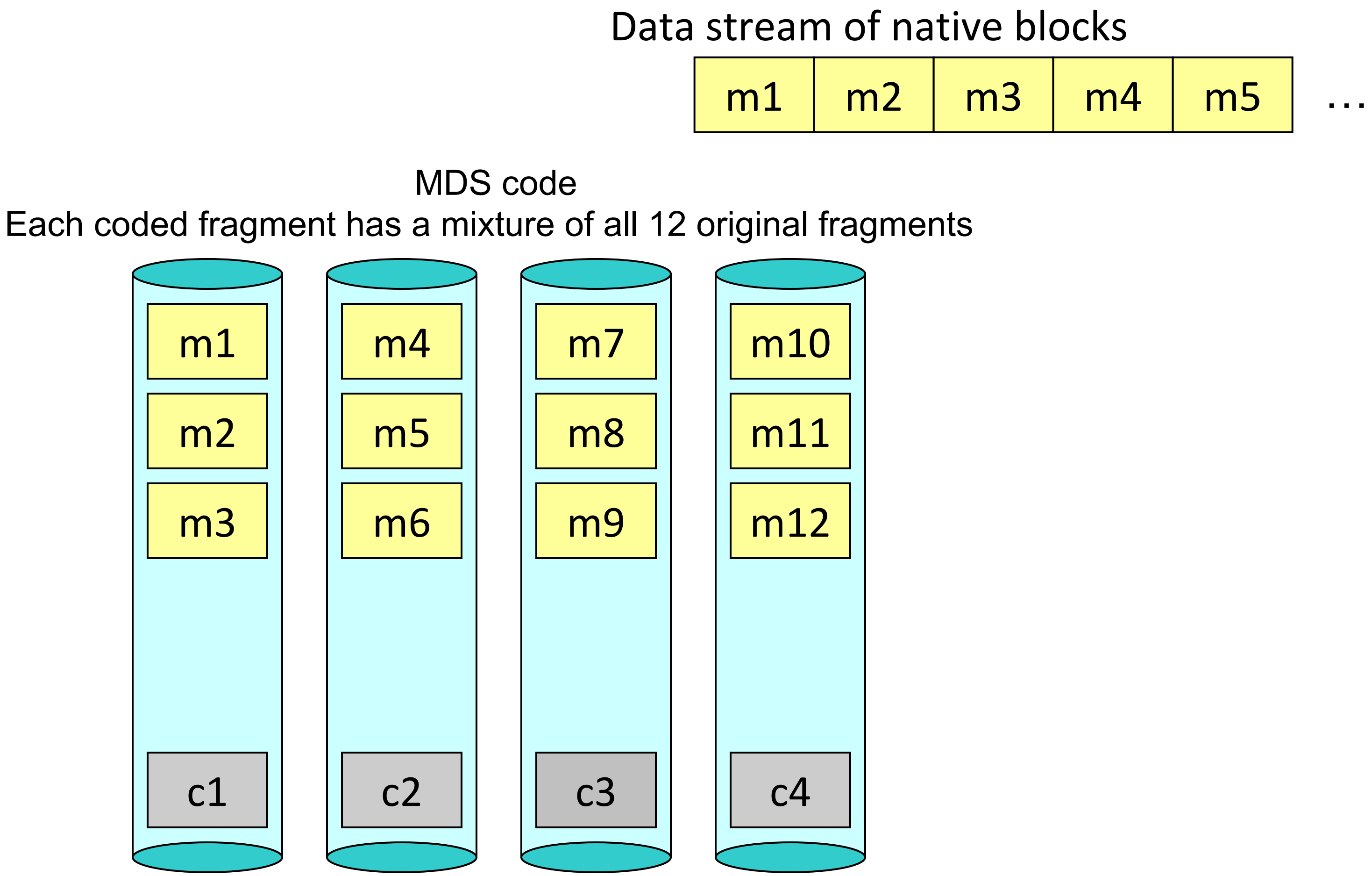


- ➢ Code block:
  - c1 = m1 + m2 + m3 + m4 + m5
- ➢ Parameters:
  - n = can be any ≥ 3
  - k = n - 2
  - d = n - 1
  - m = (n-2)(n+1)/2
  - c = 1

Regenerating Codes

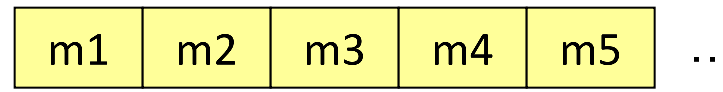
- E-MBR(n, k, d=n-1)
  - For general n, k
  - Each native/code block still has a duplicate copy
  - Code blocks are formed by Reed-Solomon codes
- Parameters:
  - n = can be any $\geq k+1$
  - k = can be any
  - $d = n - 1$
  - $m = k(2n-k-1)/2$
  - $c = (n - k)(n - k - 1)/2$

Regenerating Codes



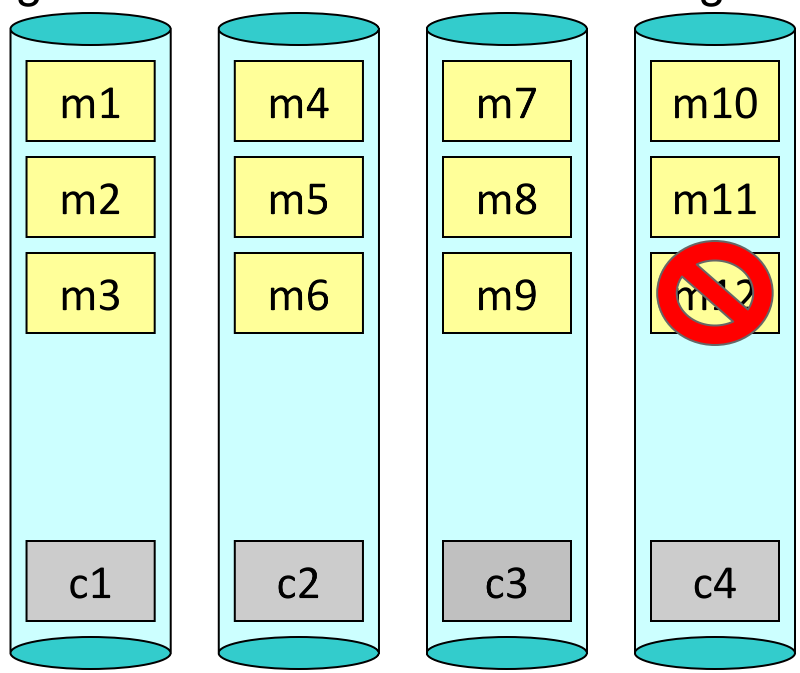
Regenerating Codes

Data stream of native blocks



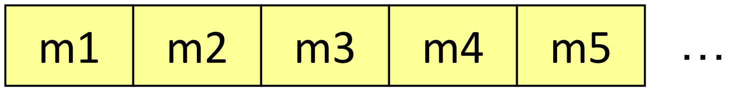
MDS code

Each coded fragment has a mixture of all 12 original fragments



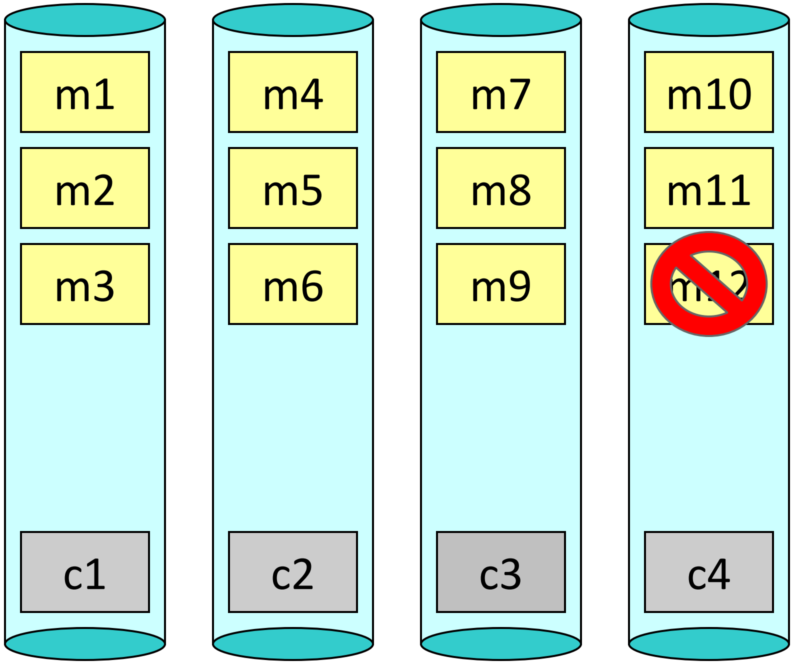
How to recover?

Regenerating Codes



MDS code

Each coded fragment has a mixture of all 12 original fragments

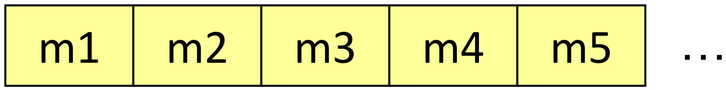


“Interference Cancellation”

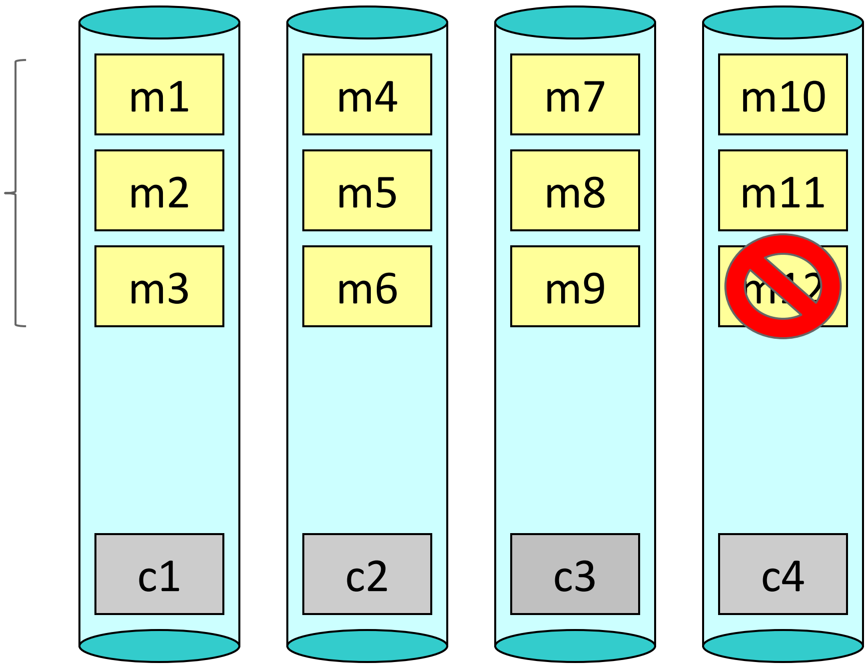
If we know c4 = m1 + 2 x m2 + 3 x m3 + … + 11 x m11 + 12 x m12

Regenerating Codes

Data stream of native blocks



Generate & send:
m1 + 2 x m2 + 3 x m3



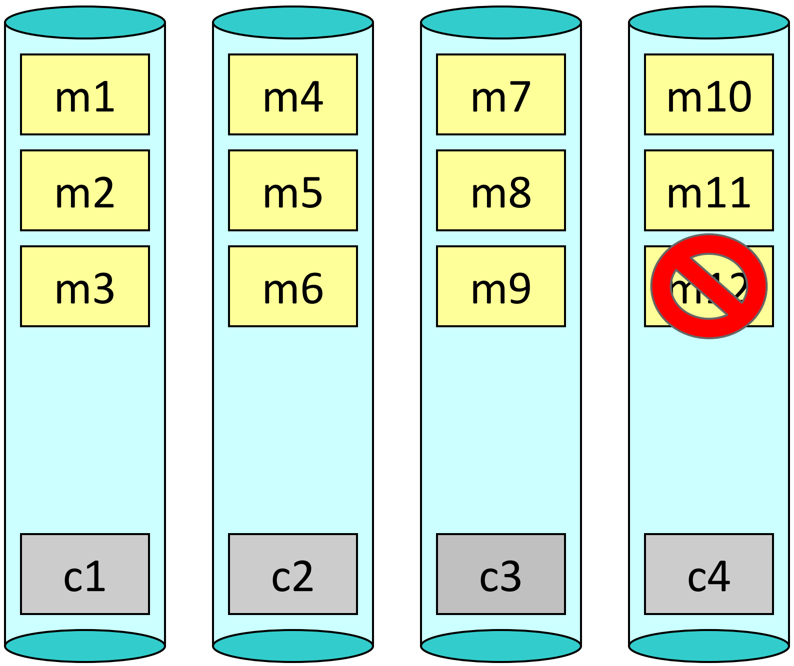
"Interference Cancellation"
If we know c4 = m1 + 2 x m2 + 3 x m3 + … + 11 x m11 + 12 x m12

Regenerating Codes

Data stream of native blocks

m1	m2	m3	m4	m5	…

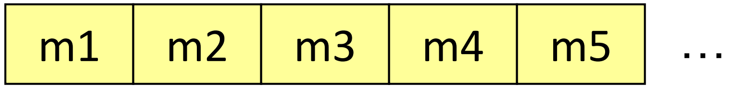
Generate & send:
4 x m4 + 5 x m5 + 6 x m6



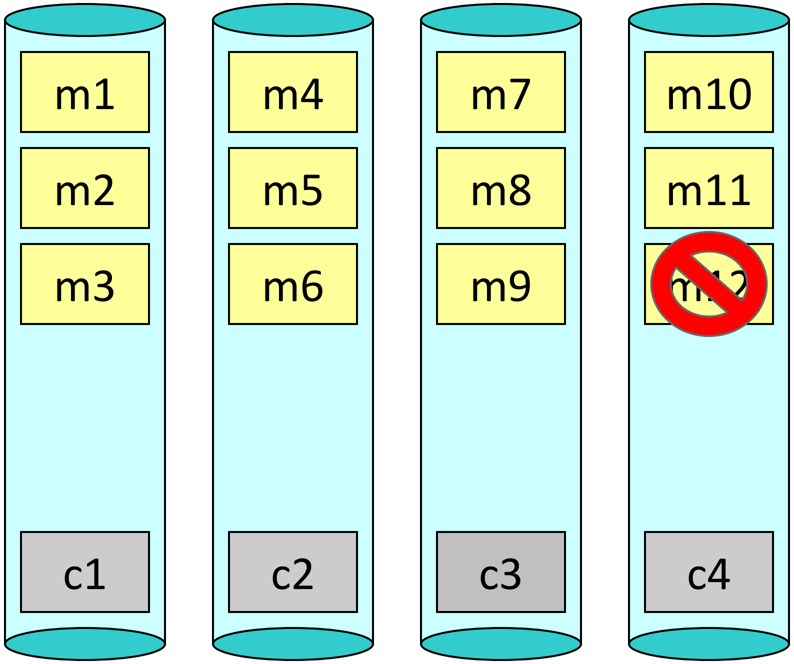
"Interference Cancellation"
If we know c4 = m1 + 2 x m2 + 3 x m3 + … + 11 x m11 + 12 x m12

Regenerating Codes

Data stream of native blocks



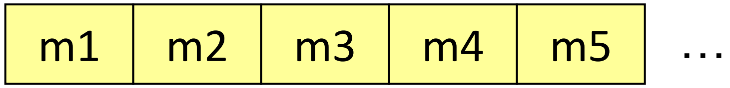
Generate & send:
7 x m7 + 8 x m8 + 9 x m9

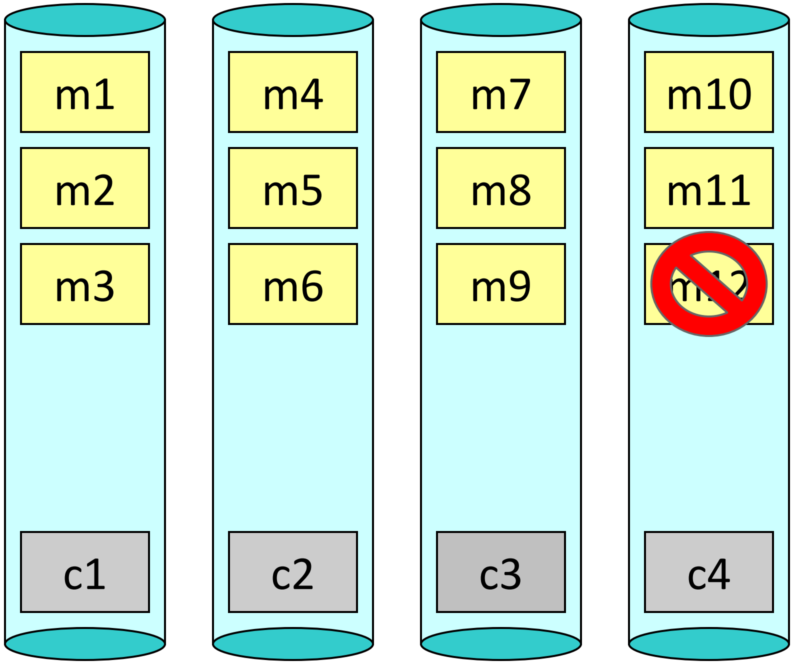


"Interference Cancellation"
If we know c4 = m1 + 2 x m2 + 3 x m3 + … + 11 x m11 + 12 x m12

Regenerating Codes

Data stream of native blocks

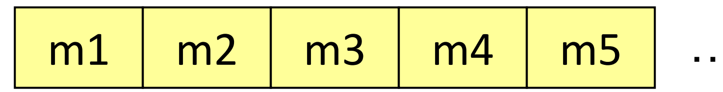


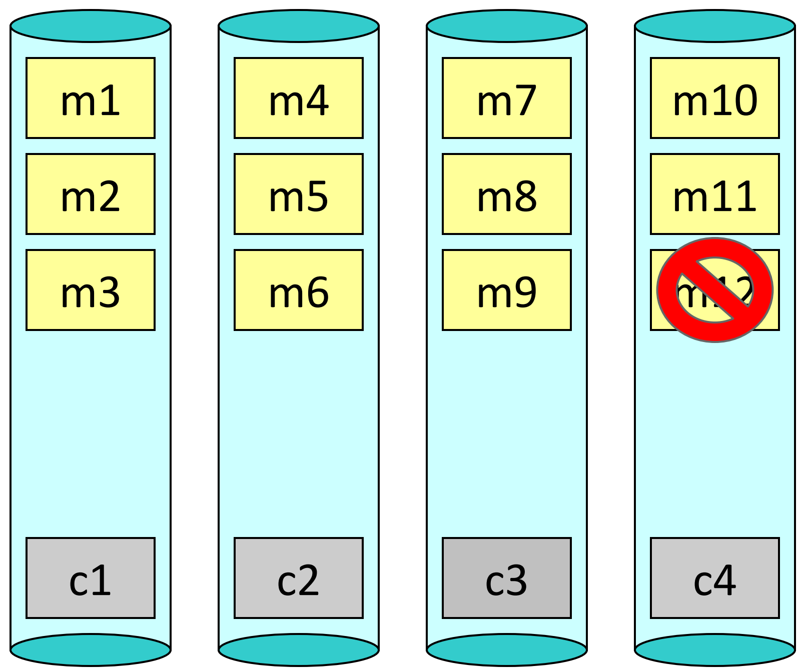


XOR factors from c4 to obtain
10 x m10 + 11 x m11 + 12 x m12

Regenerating Codes

Data stream of native blocks

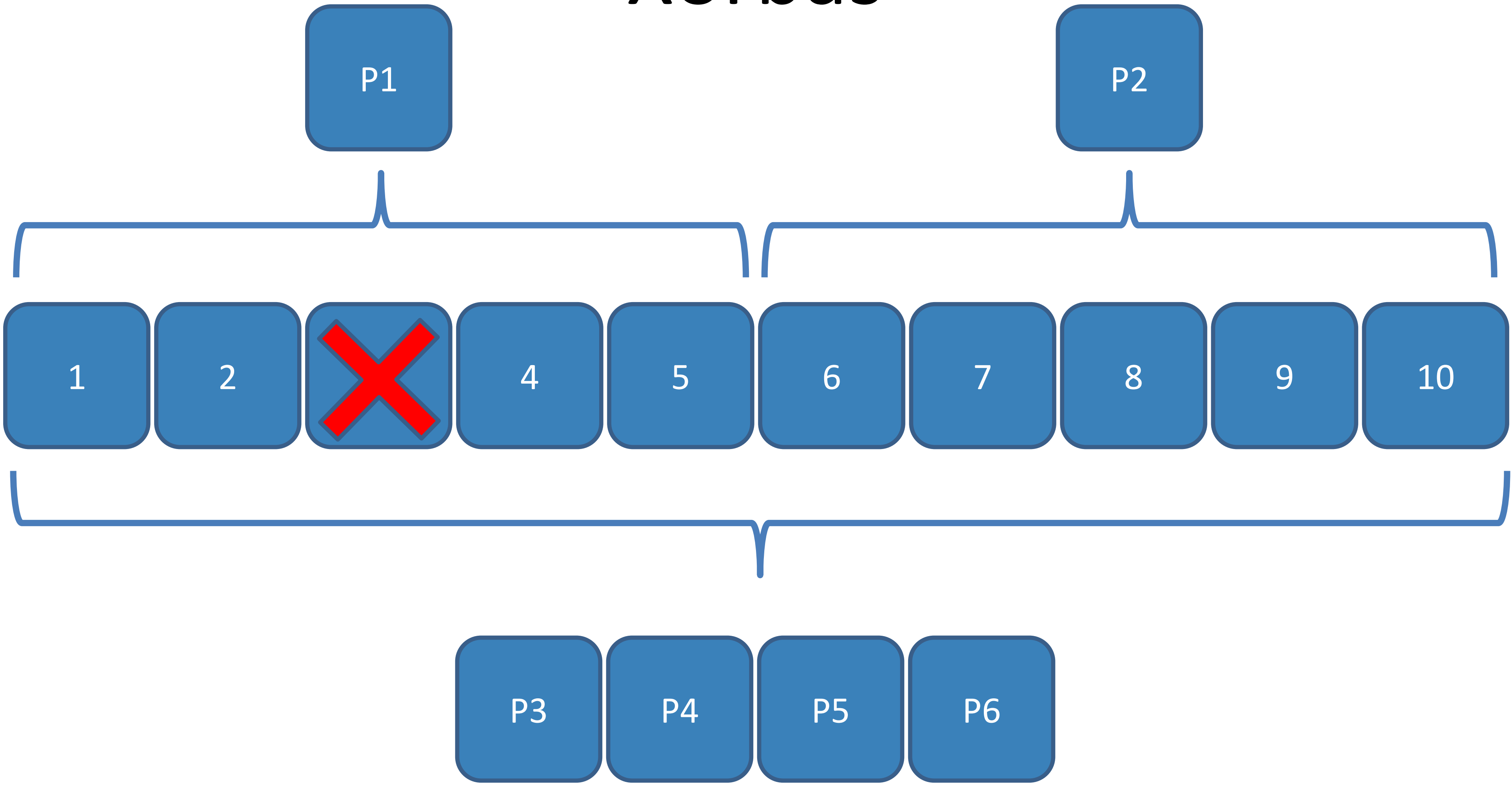




Remove 10 x m10 + 11 x m11 from the local data to get **12 x m12**
(Multiply by 12^{-1})
Exact repair with a Total of 3 inter-rack transmissions :-)

Local Repair

Xorbas



One failure results in 5 traffic units

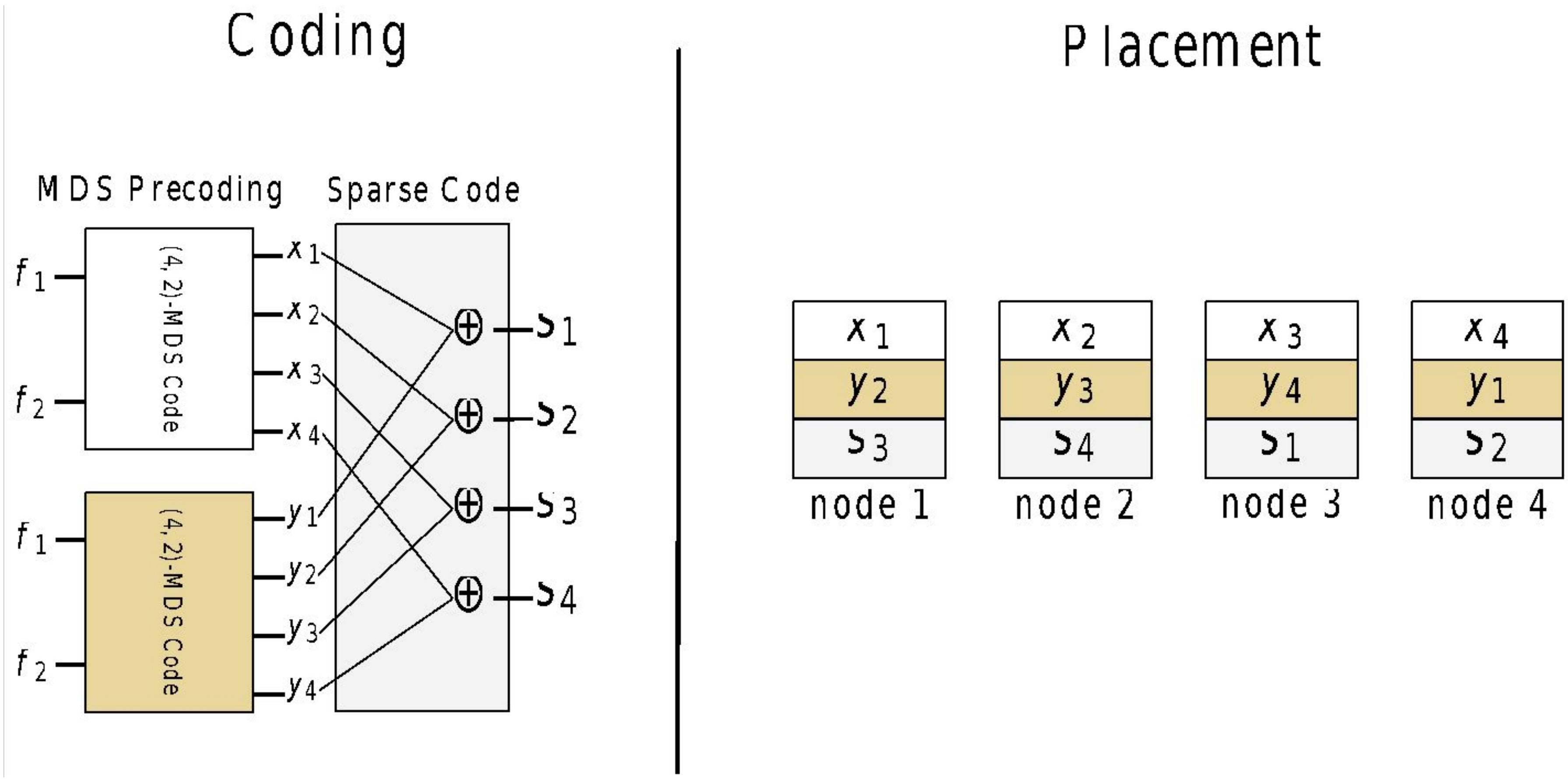
Local Repairability

Theorem 1 For locality r, there is a related storage cost

$$(r\ ,\gamma) = \left(r, \alpha_{MDS}\left(1 + \frac{1}{r}\right)\right)$$

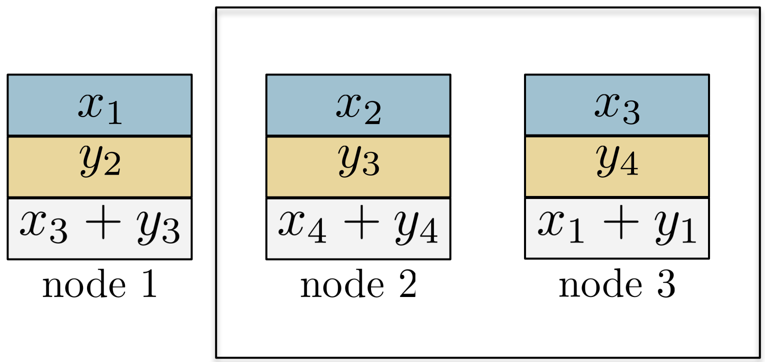
Theorem 2 This is the optimal trade-off between locality and storage

Simple example



Adapted slides from A. Dimakis

(4,2) example

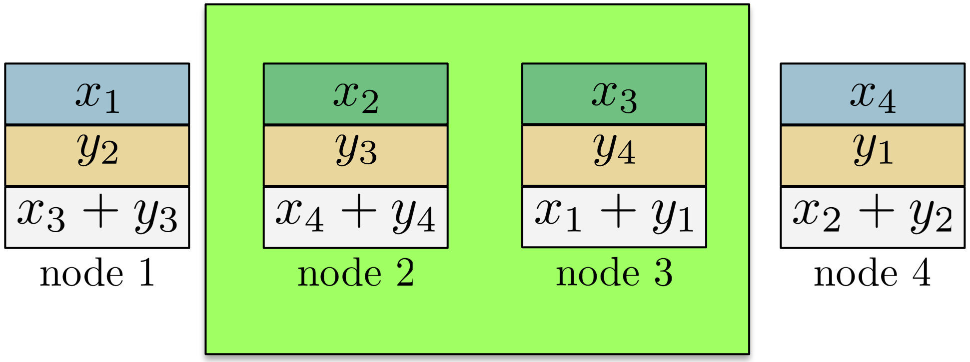


- n=4 nodes, each node stores 2 data packets and one fork (f=2).

Any k=2 nodes can recover (even without using the forks)

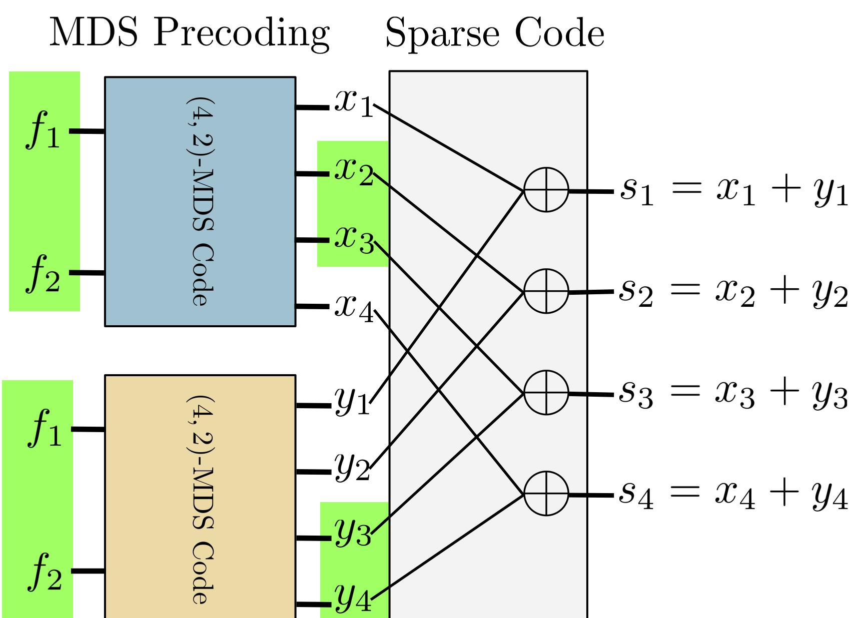
Adapted slides from A. Dimakis

(4,2) example



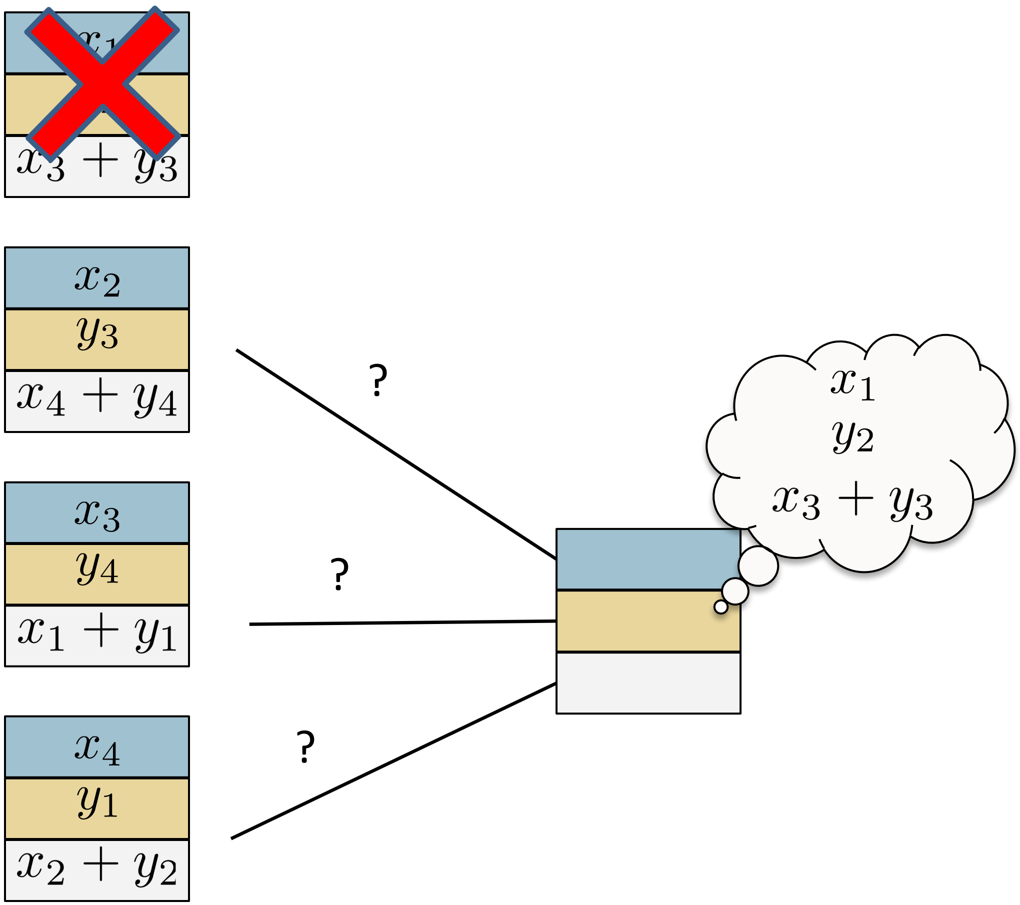
- n=4 nodes, each node stores 2 data packets and one fork (f=2).

Any k=2 nodes can recover the file (f1,f2)



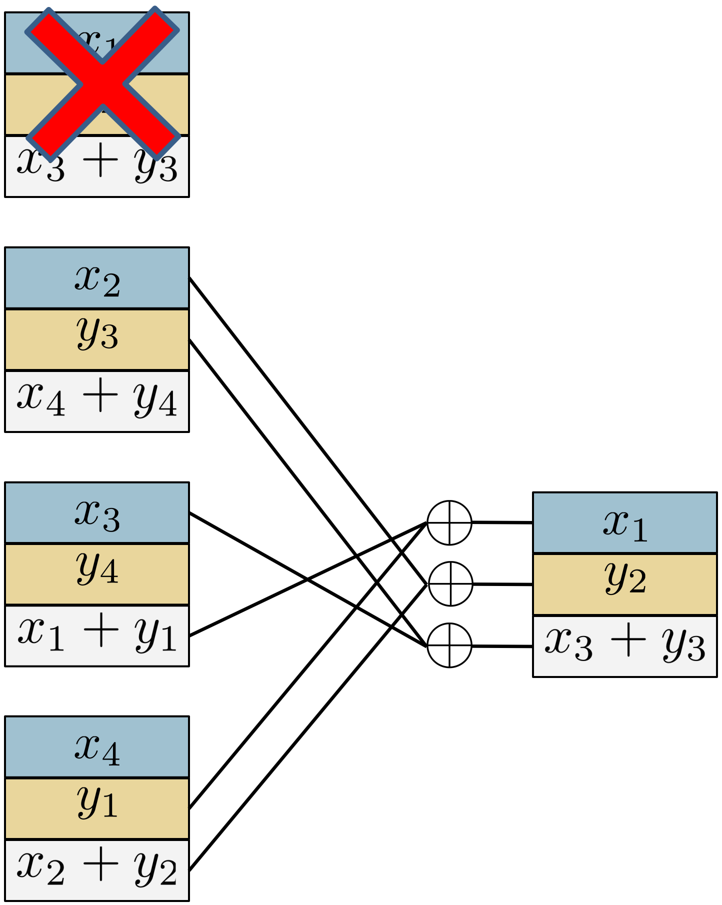
Adapted slides from A. Dimakis

(4,2) example- exact repair



Adapted slides from A. Dimakis

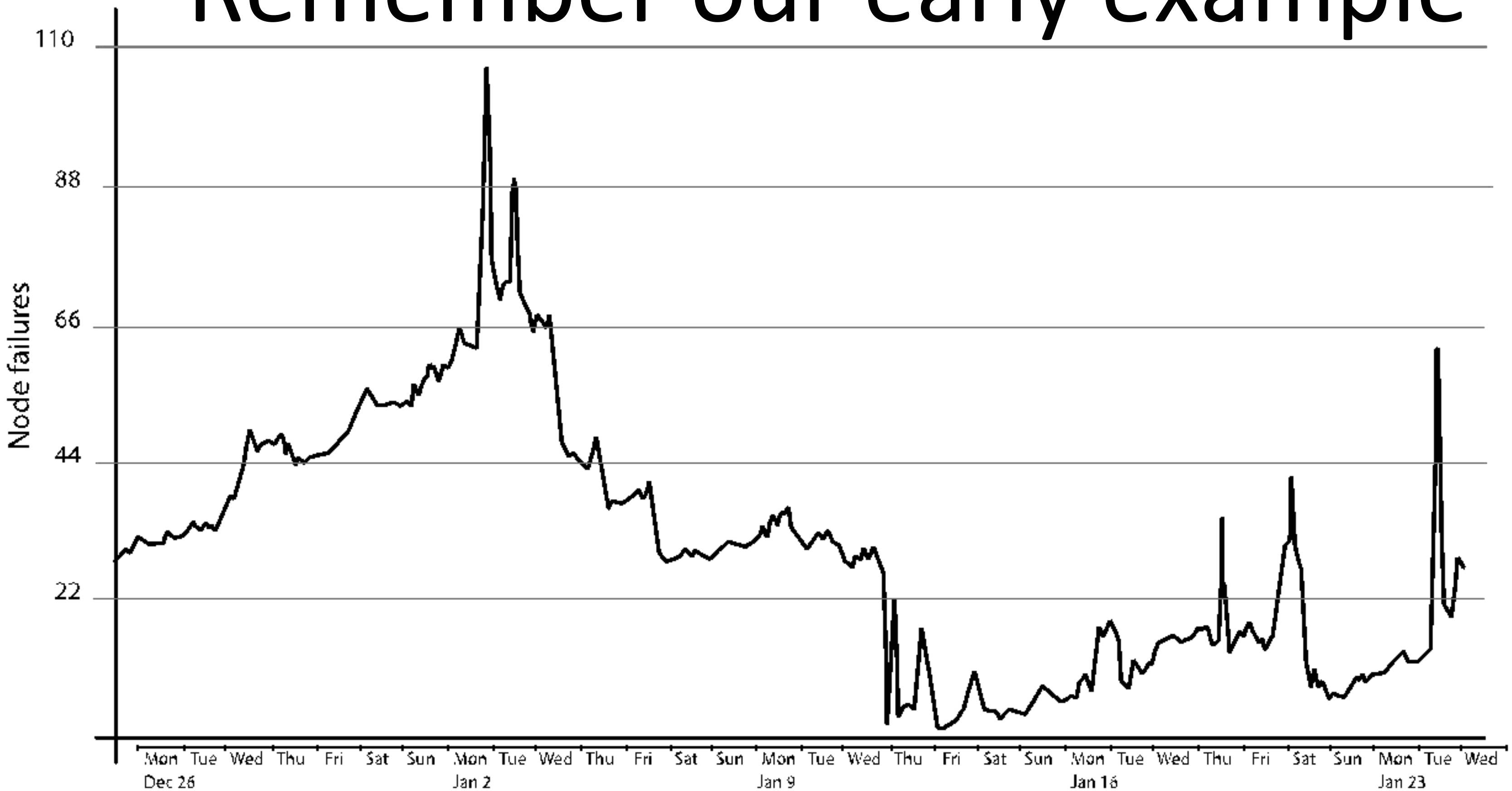
(4,2) example- exact repair



- Outer MDS codes used to provide the (n,k) safety
- Must ensure that the 'sparse' combinations and its parents are stored in different nodes

Adapted slides from A. Dimakis

Big Picture
Remember our early example



20 node failures * 15TB = 300TB

if 8% RS coded, 588TB network traffic/day. (average total network: 2PB/day)

~30% of network traffic is repair in a normal day

Adapted from A. Dimakis – Facebook measurements

Big Picture

Goal: *Maintain network use low*

What was Facebook doing?

Only 8-9% of storage with 10:4 code, rest is 3-way replication

- Network use?

- Total storage reduction?

Big Picture

Goal: *Maintain network use low*

What was Facebook doing?

Only 8-9% of storage with 10:4 code, rest is 3-way replication

- This keeps network use to roughly 2 units per lost unit (assume 1 fragment loss at a time)
  - 91 - 92% of cases 1 Unit of repair
  - 8 - 9% of cases: 10 Units of repair

0,91 x 1 + 0,09 x 10 = 1,81 Units

- Total storage reduction?

Big Picture

Goal: *Maintain network use low*

What was Facebook doing?

Only 8-9% of storage with 10:4 code, rest is 3-way replication

- This keeps network use to roughly 2 units per lost unit (assume 1 fragment loss at a time)
- Total storage reduction: ~5%
  - 91 - 92% of cases 3x
  - 8 - 9% of cases: 1.4x

$$\left.\begin{array}{l}91-92\% \text{ of cases } 3x \\ 8-9\% \text{ of cases: } 1.4x\end{array}\right\} 0{,}91 \times 3 + 0{,}09 \times 1.4 = 2{,}856$$

  - Basically, From 3x to 2.856x → 5%

Can XORBAS help?

Maintaining network use to roughly 2 files per lost file as an acceptable measure

- Network traffic (assuming 1 loss at a time):
  - Fraction α of cases: 1 Unit of repair
  - Fraction 1- α of cases: 5 Units of repair
- Storage use:
  - Fraction α of cases: 3.0x
  - Fraction 1- α of cases: 1.6x

Can XORBAS help?

Maintaining network use to roughly 2 files per lost file as an acceptable measure

- Network traffic:
  - $\alpha \times 1 + (1 - \alpha) \times 5 = 2 \quad \rightarrow 5 - 2 = 4 \times \alpha$

    $\rightarrow \alpha = 0{,}75$
- Storage use:
  - Fraction α of cases: 3.0x
  - Fraction $1 - \alpha$ of cases: 1.6x

Can XORBAS help?

Maintaining network use to roughly 2 files per lost file as an acceptable measure

- Network traffic:
  - $\alpha \times 1 + (1-\alpha) \times 5 = 2 \quad \rightarrow 5 - 2 = 4 \times \alpha$

    $\rightarrow \alpha = 0,75$
- Storage use:
  - $\alpha \times 3.0 + (1-\alpha) \times 1.6 \quad = 0,75 \times 3 + 0,25 \times 1.6$

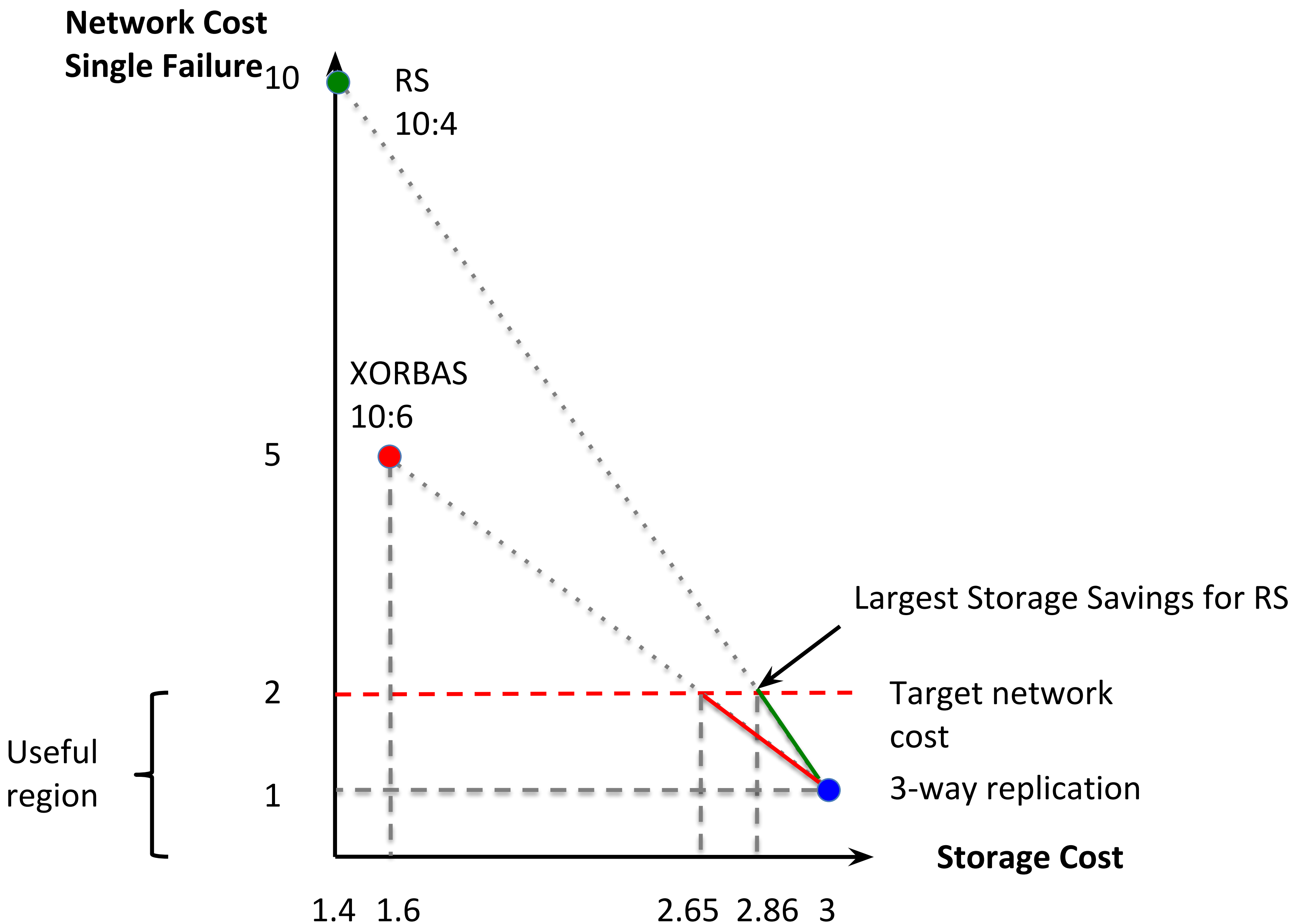
    $= 2,65$

Can XORBAS help?

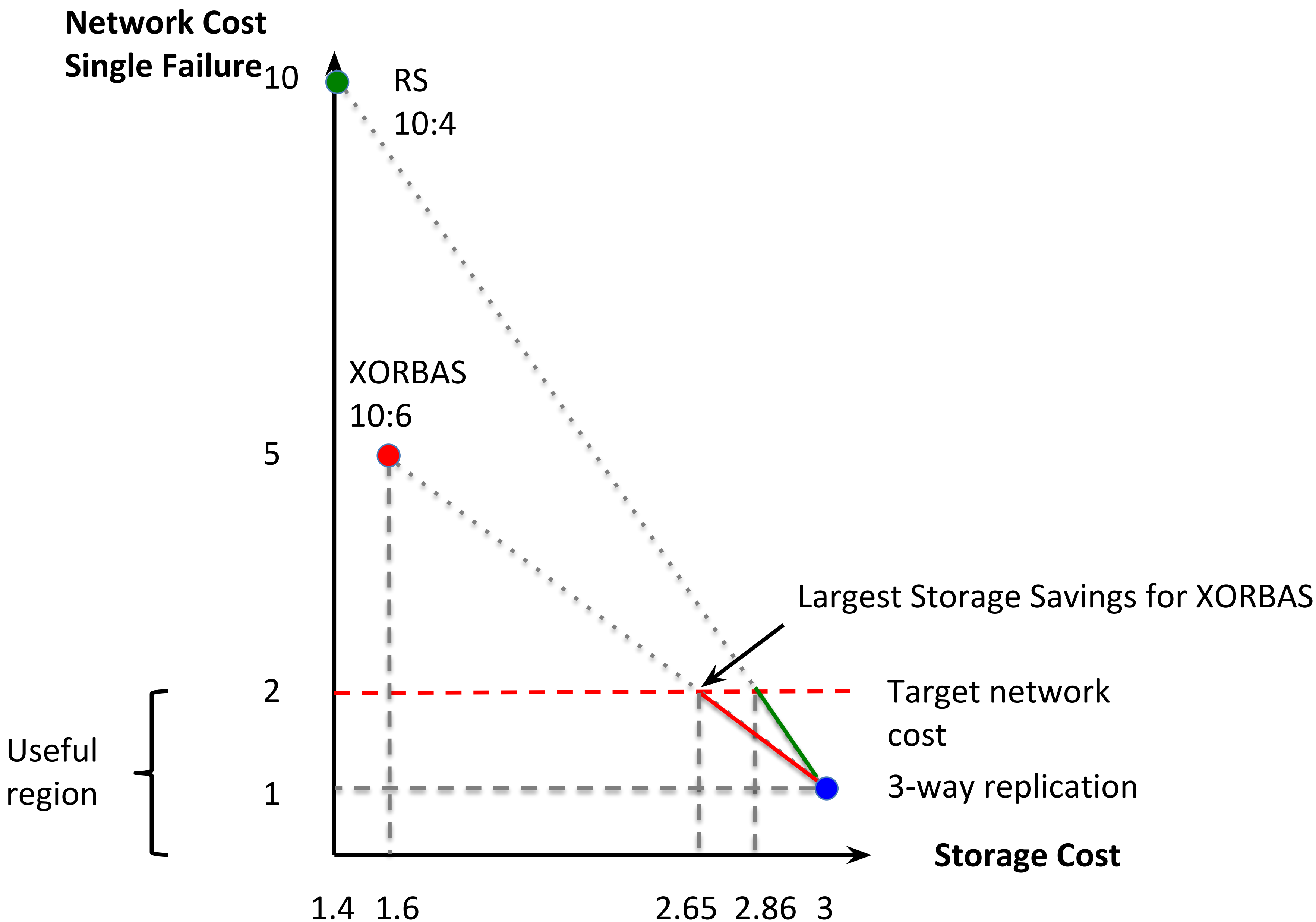
Maintaining network use to roughly 2 files per lost file as an acceptable measure

- 25% of files are coded, 75% are 3-way replication
    - May hurt availability of that 25%
- Total storage reduction
    - From 3x to 2.65x $\longrightarrow$ 13% storage reduction
    - Better than before

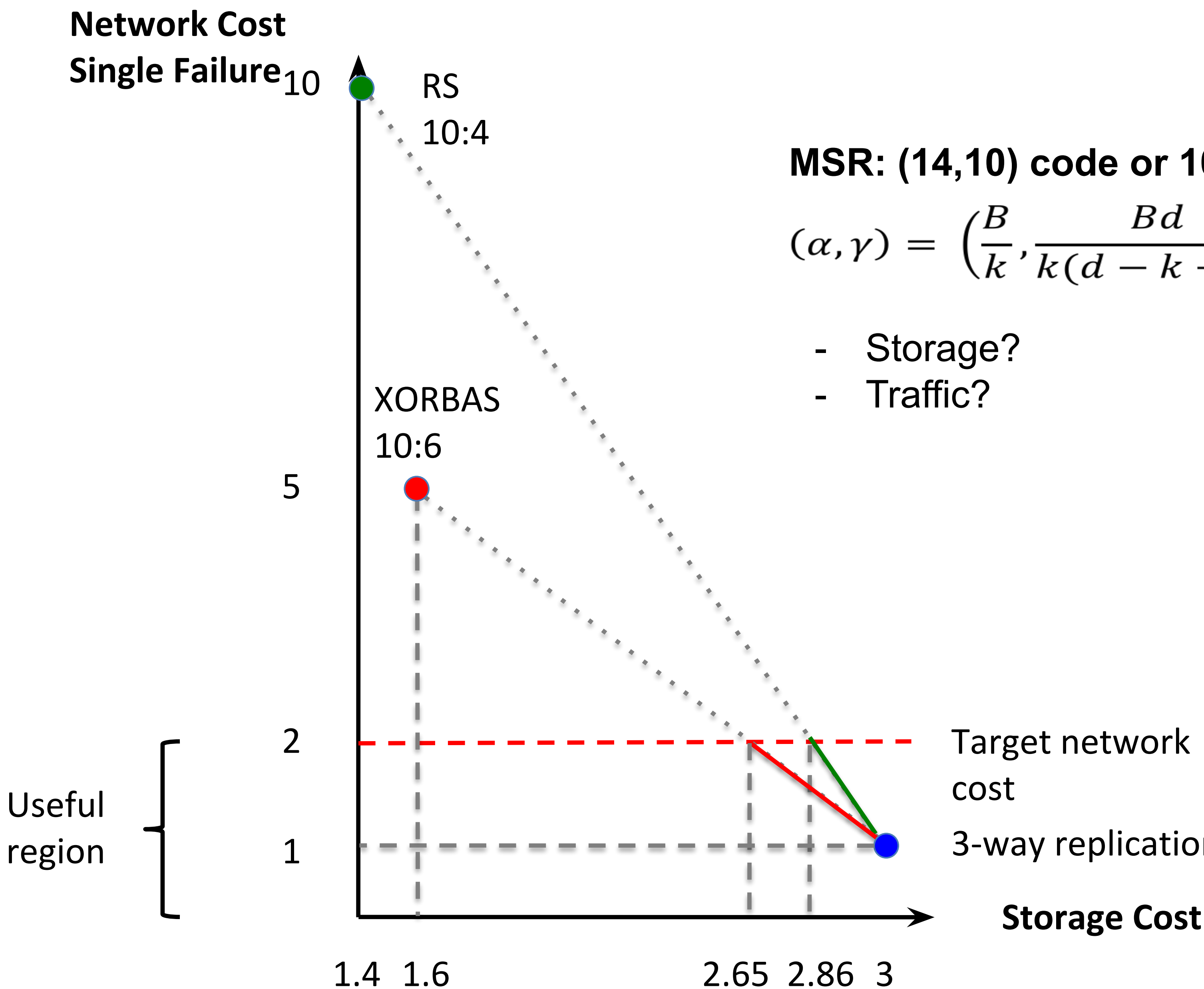
Some Perspective



Some Perspective



What about regenerating codes?

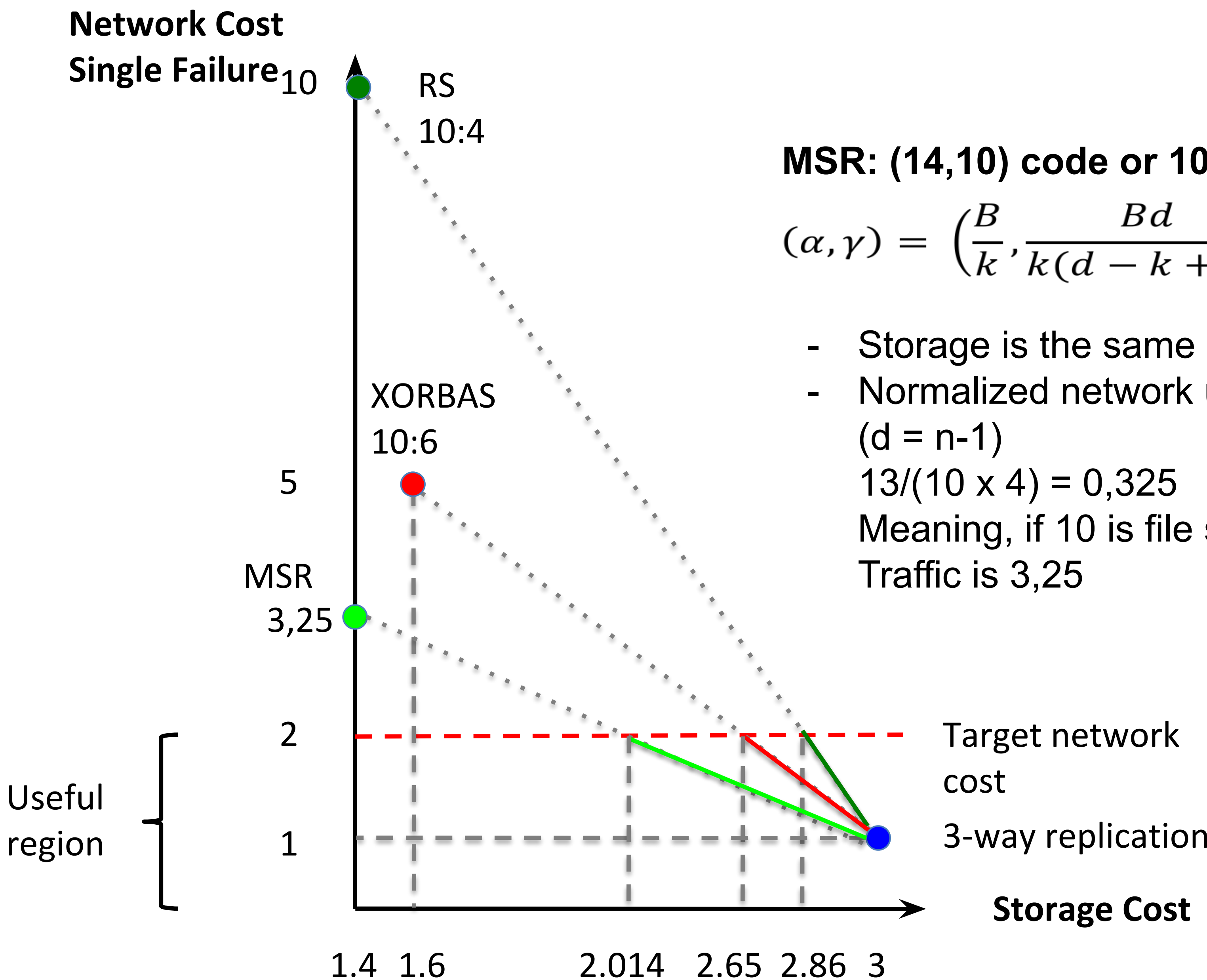


MSR: (14,10) code or 10:4 code

$$(\alpha, \gamma) = \left(\frac{B}{k}, \frac{Bd}{k(d-k+1)}\right)$$

- Storage?
- Traffic?

What about regenerating codes?

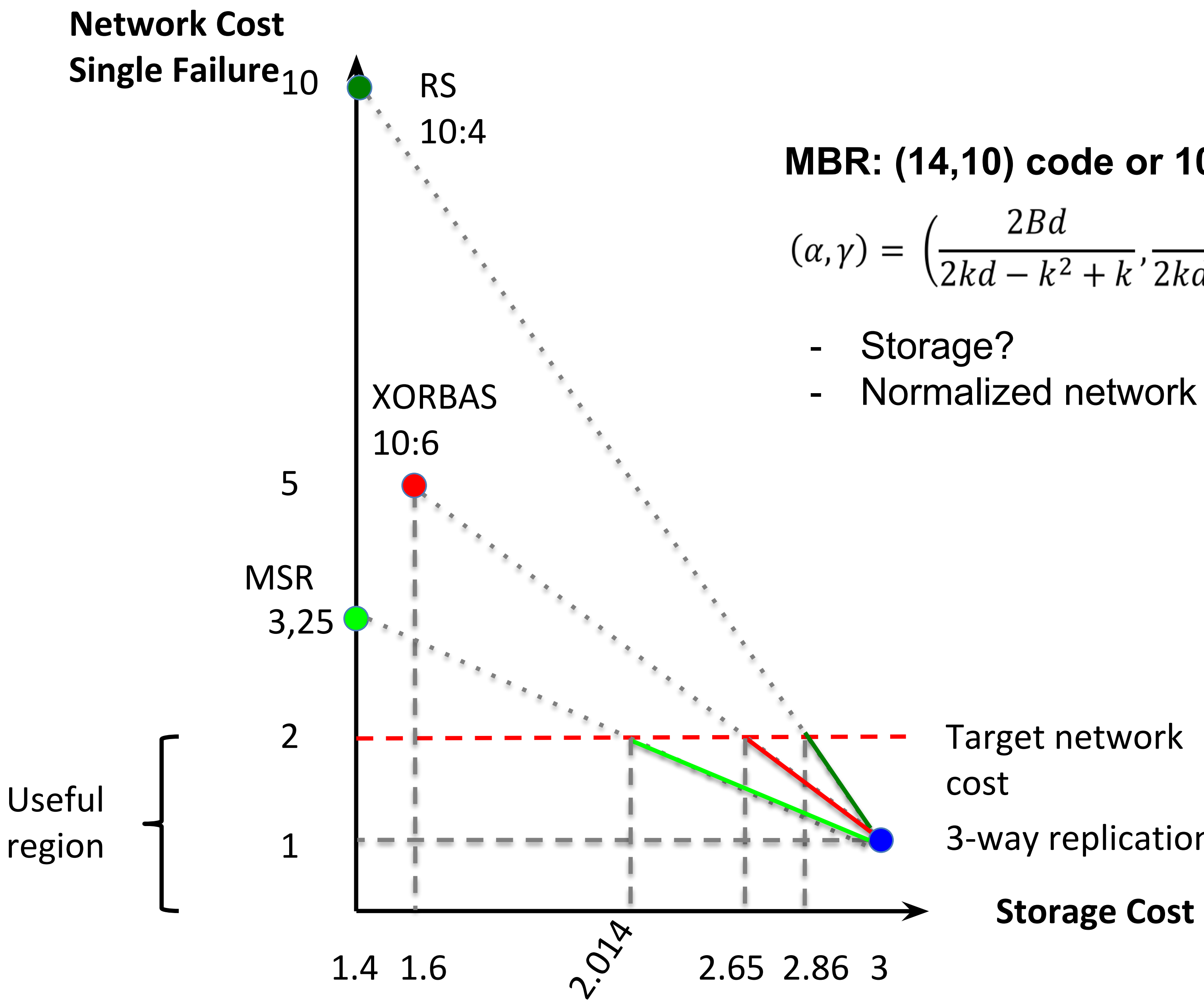


MSR: (14,10) code or 10:4 code

$$(\alpha, \gamma) = \left(\frac{B}{k}, \frac{Bd}{k(d-k+1)}\right)$$

- Storage is the same as RS
- Normalized network use is
  (d = n-1)
  13/(10 x 4) = 0,325
  Meaning, if 10 is file size, then
  Traffic is 3,25

What about regenerating codes?

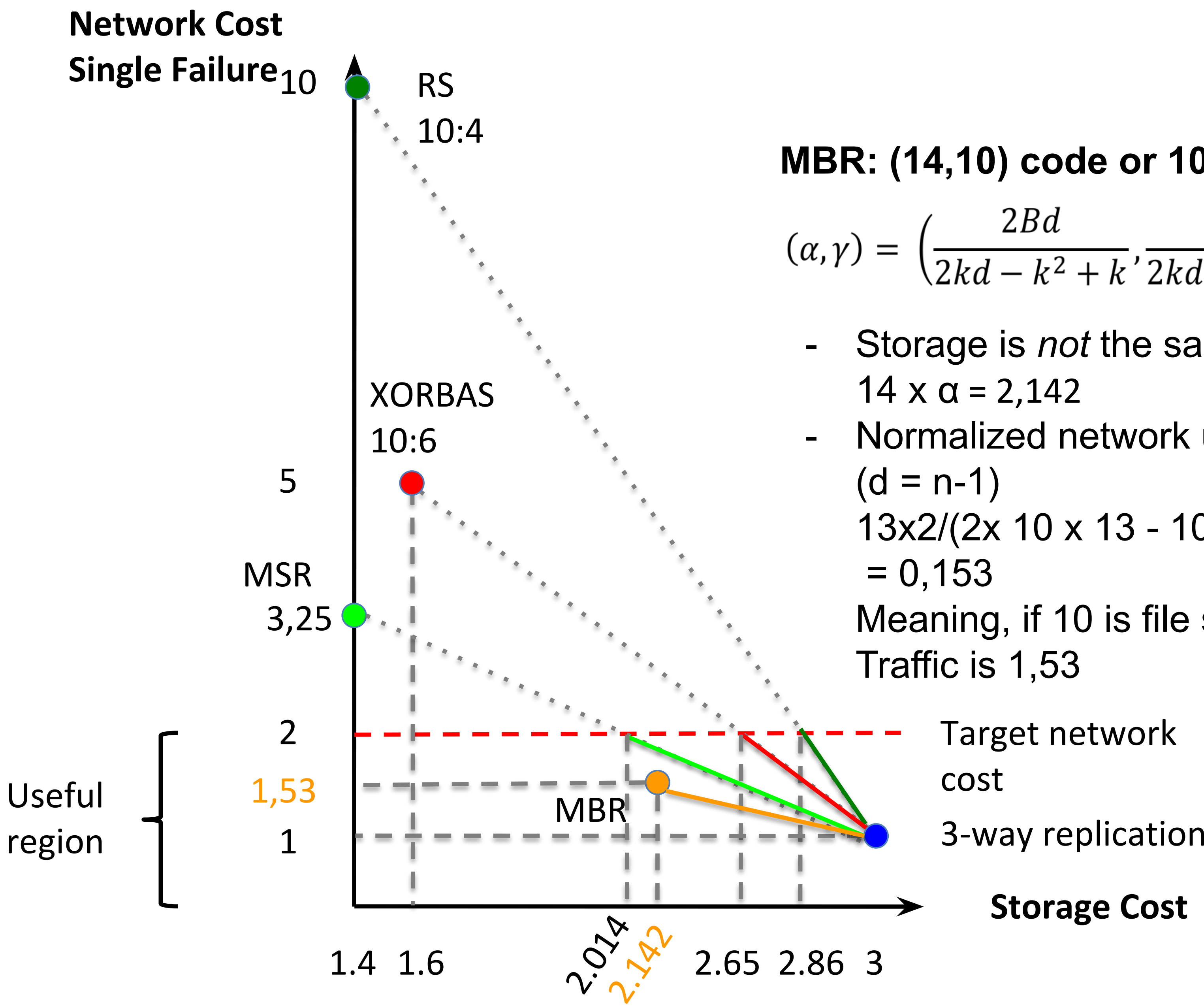


MBR: (14,10) code or 10:4 code

$$(\alpha, \gamma) = \left(\frac{2Bd}{2kd - k^2 + k}, \frac{2Bd}{2kd - k^2 + k}\right)$$

- Storage?
- Normalized network use?

What about regenerating codes?

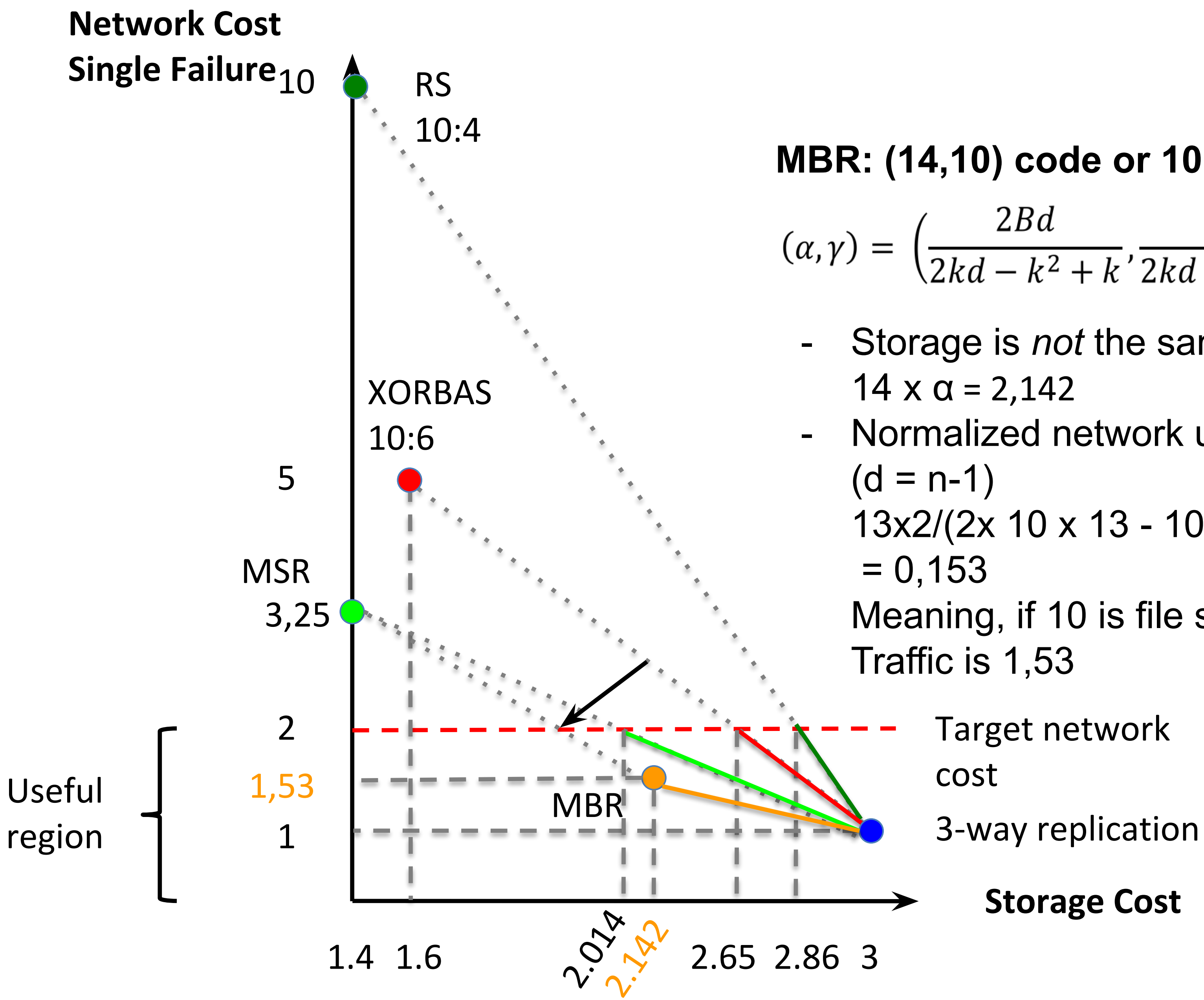


MBR: (14,10) code or 10:4 code

$$(\alpha,\gamma) = \left(\frac{2Bd}{2kd-k^2+k}, \frac{2Bd}{2kd-k^2+k}\right)$$

- Storage is *not* the same as RS
  14 x α = 2,142
- Normalized network use is
  (d = n-1)
  13x2/(2x 10 x 13 - 100 +10)
  = 0,153
  Meaning, if 10 is file size, then
  Traffic is 1,53

What about regenerating codes?



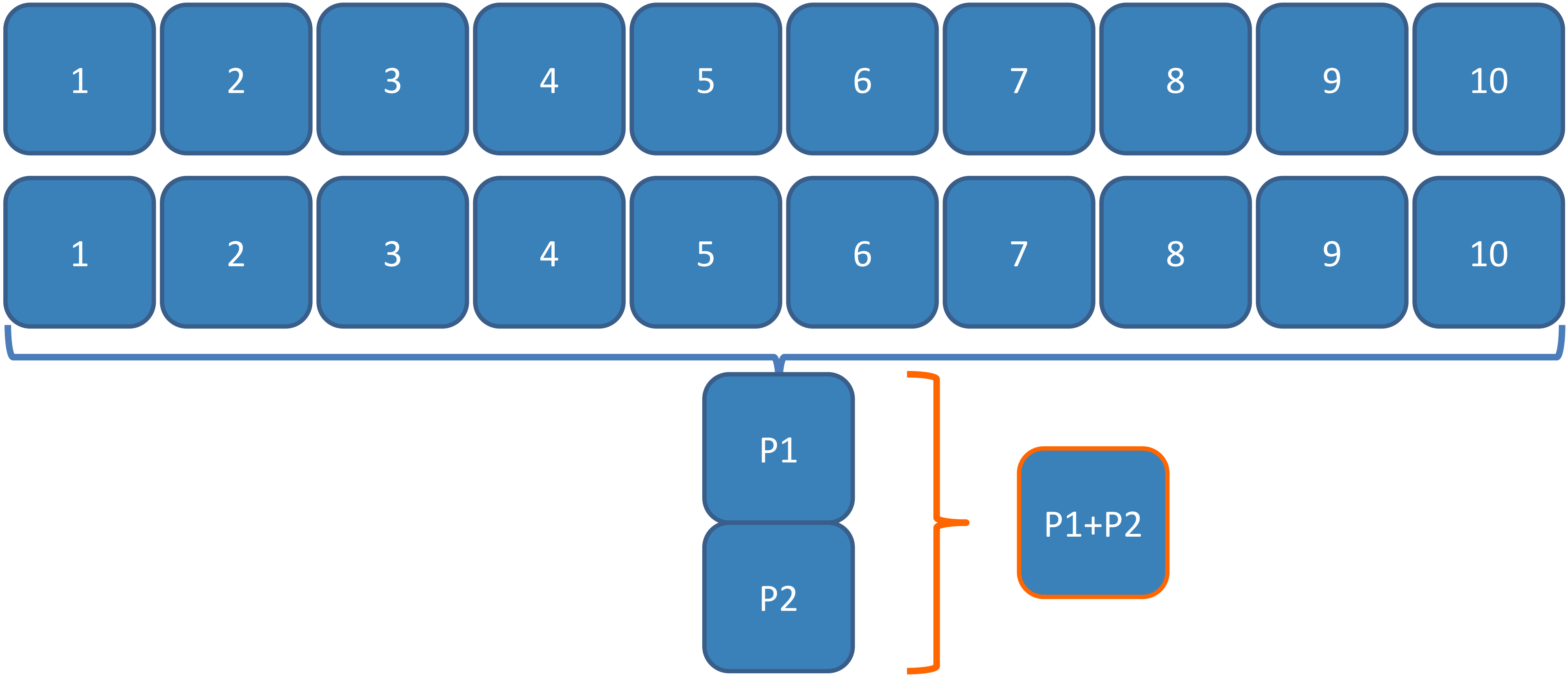
MBR: (14,10) code or 10:4 code

$$(\alpha, \gamma) = \left(\frac{2Bd}{2kd - k^2 + k}, \frac{2Bd}{2kd - k^2 + k}\right)$$

- Storage is *not* the same as RS
  14 x α = 2,142
- Normalized network use is
  (d = n-1)
  13x2/(2x 10 x 13 - 100 +10)
  = 0,153
  Meaning, if 10 is file size, then
  Traffic is 1,53

What about a “silly” solution?

Hybrid Replication and Coding



Storage cost: 2.3 stored files per original file

Network use (recover 1 loss): 26/23 ~ 1.13 file

Reliability: Recovery from at least 4 losses

Processing: encoding 1 file when losing P1, P2, or P1+P2

Can this help?

Maintaining network use to roughly 2 files per lost file as an acceptable measure

- We could in principle have all files in this coding
  - Minimal increase in bandwidth for repair (13% wrt 3-way replication)
  - Higher reliability than 3-way replication
  - Same or higher reliability as RS 10:4 code
  - May hurt availability slightly
  - Very low processing requirement most of the time
- Storage reduction? From 3x to 2.3x → 30.4%!
  - Merely determined by availability
  - Now we are making a dent (from 5% to 30.4%)

Is it “silly”?

1. Network use under control
2. Storage reduction:

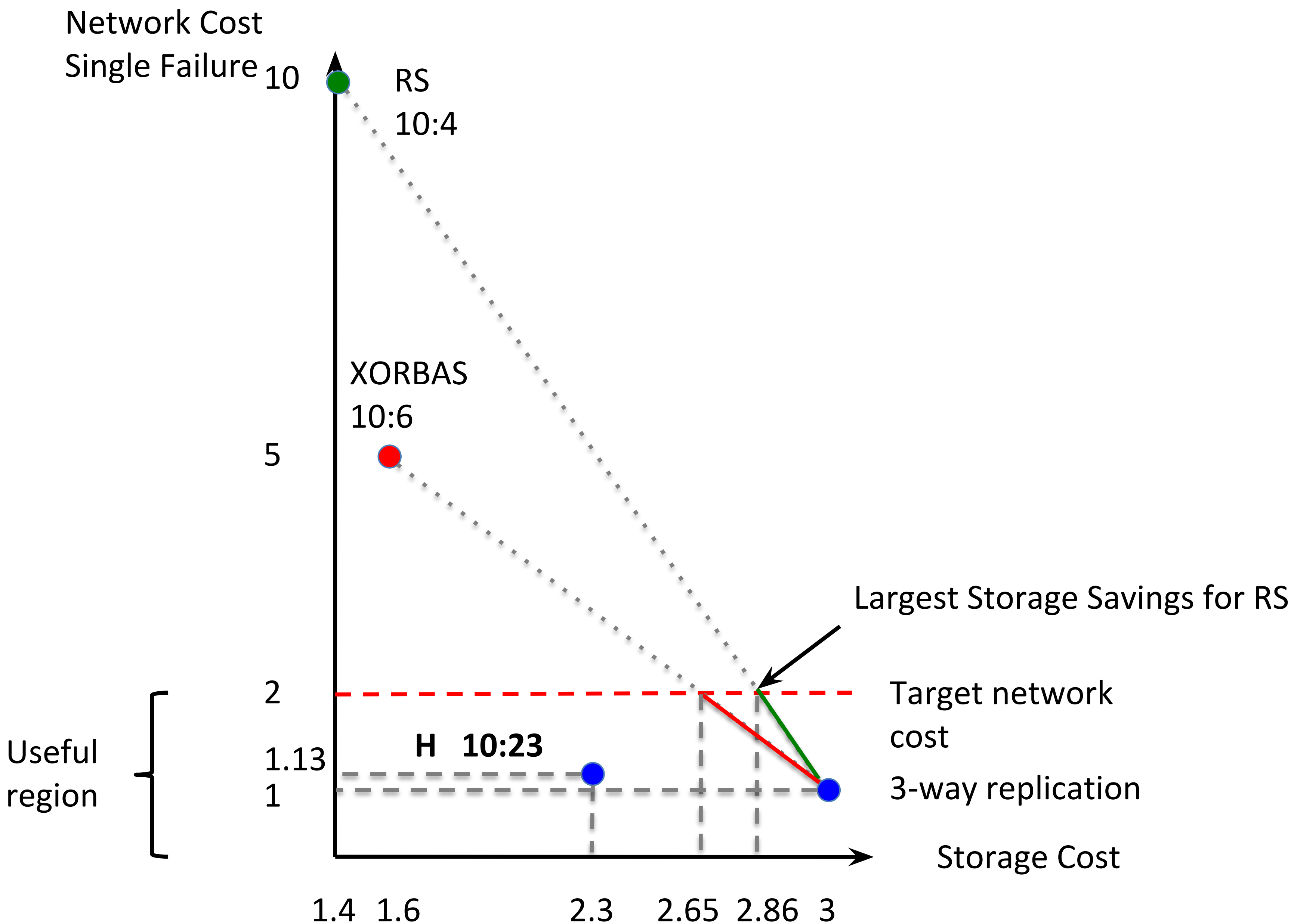
   10:4code → up to 5%

   XORBAS → up to 13%

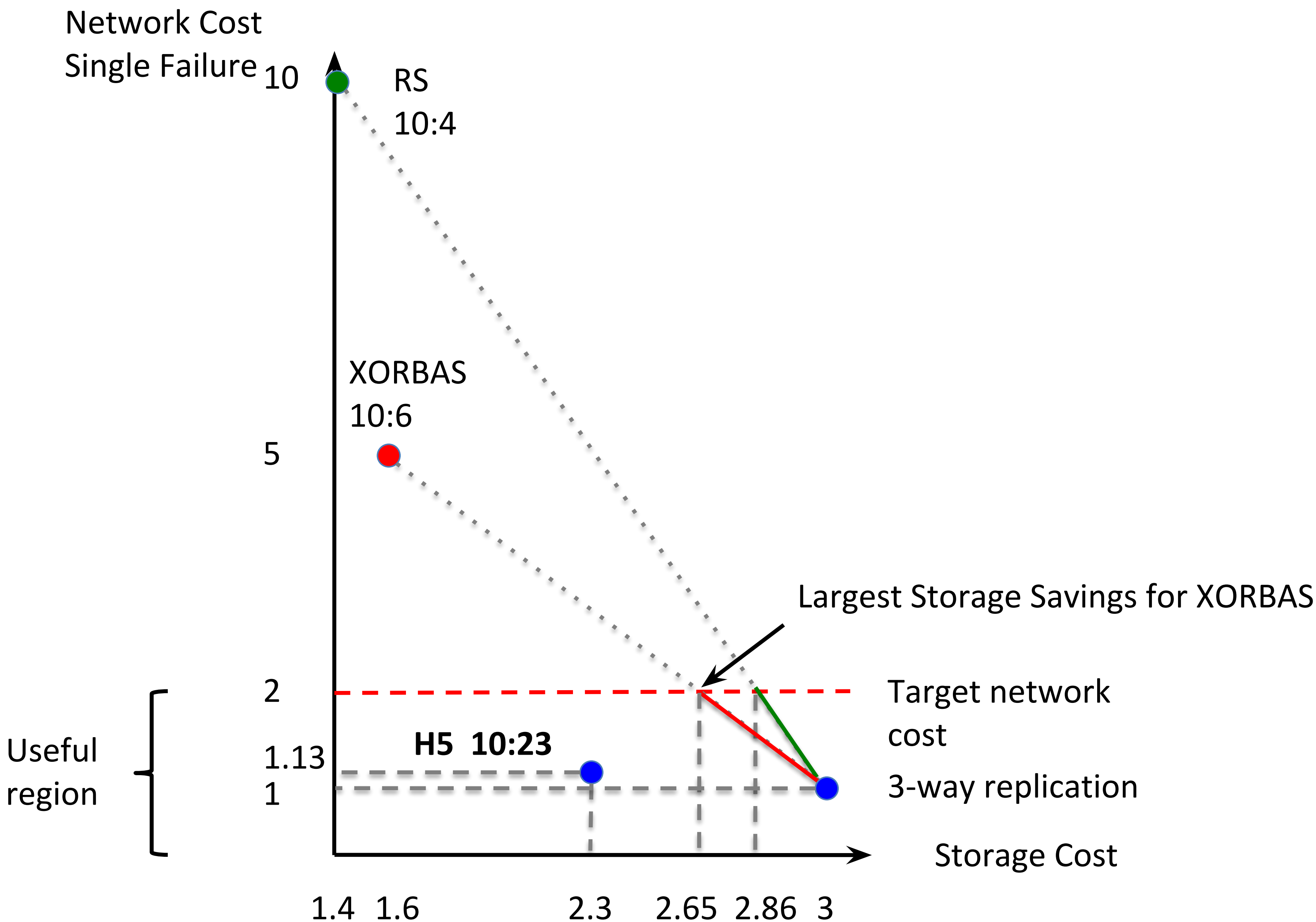
   Best “Silly” approach

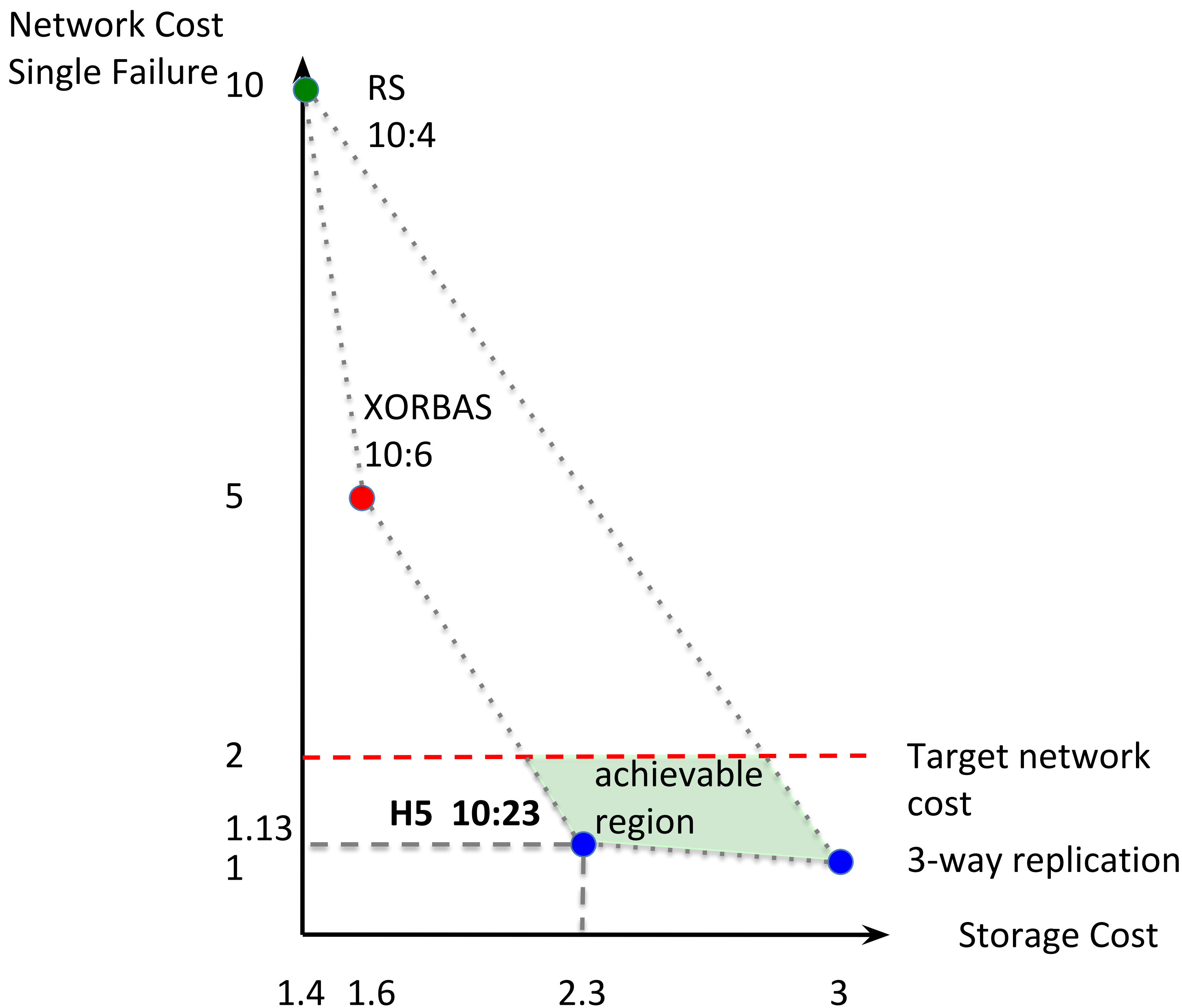
   - Savings of 30.4% with higher reliability than 10:4 code
   - Only 13% traffic increase for the loss of 1 unit

3. More important: the key is to make a mix of the various codes, so far mixed 2…horizon is open to richer mixtures



Network Cost
Single Failure
10
RS
10:4
XORBAS
10:6
5
Largest Storage Savings for RS
2
Target network
cost
Useful
region
1.13
H 10:23
1
3-way replication
Storage Cost
1.4
1.6
2.3
2.65
2.86
3


Network Cost
Single Failure
10
RS
10:4
XORBAS
10:6
5
Largest Storage Savings for XORBAS
2
Target network
cost
Useful
region
1.13
H5 10:23
1
3-way replication
Storage Cost
1.4
1.6
2.3
2.65
2.86
3



Network Cost
Single Failure
10
RS
10:4
XORBAS
10:6
5
2
Target network
cost
achievable
region
H5 10:23
1.13
1
3-way replication
Storage Cost
1.4
1.6
2.3
3

Is it “silly”?

Simpler operations than the others

Only 13% increase in traffic cost

Only other code that could drive all data below the target network cost is MBR (theoretical) with added complexity and without guarantees of systematic structure or exact repair