Regenerating codes for distributed storage

IEEE Information Theory Society

Santa Clara Valley Chapter

May 24, 2017

Mary Wootters

Stanford University

This talk is about

- Regenerating codes for distributed storage

- Main purpose: survey/introduction to regenerating codes
- Time permitting: some of my recent work

Not (intending to) assume any coding theory knowledge!
Ask questions!

- My ulterior motive:
  - I am a theorist!
  - I want to learn from this audience:
    - How might regenerating codes be useful in your work? (if at all)
    - Especially the second part of the talk ☺

Outline

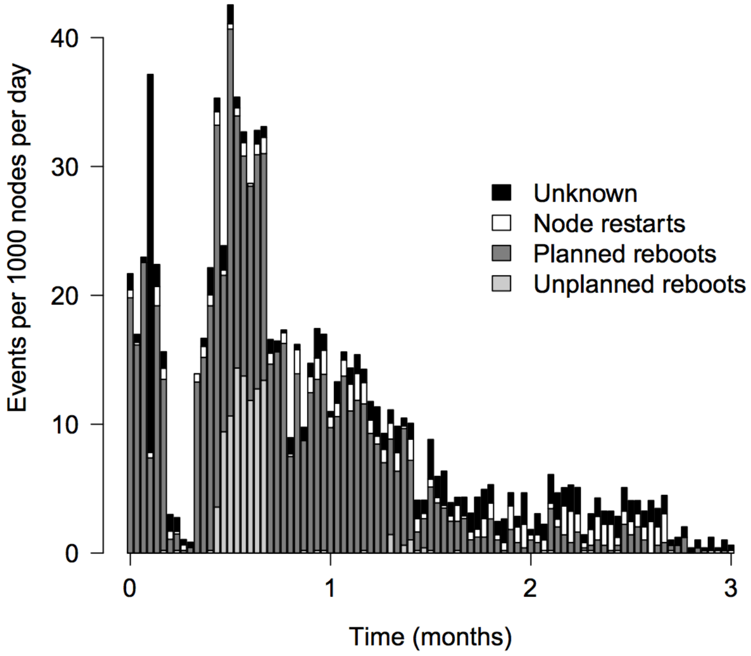
1. Coding for distributed storage: what’s the problem?
2. Coding for distributed storage: how do we solve the problem?
   - Try 1: replication
   - Try 2: classical erasure coding
   - Try 3: regenerating (MSR) codes
3. What can we do with regenerating codes?
   - Basic bounds
4. How about codes I know and love?
   - Reed-Solomon Codes
5. Future work/Open problems

1. What’s the problem?

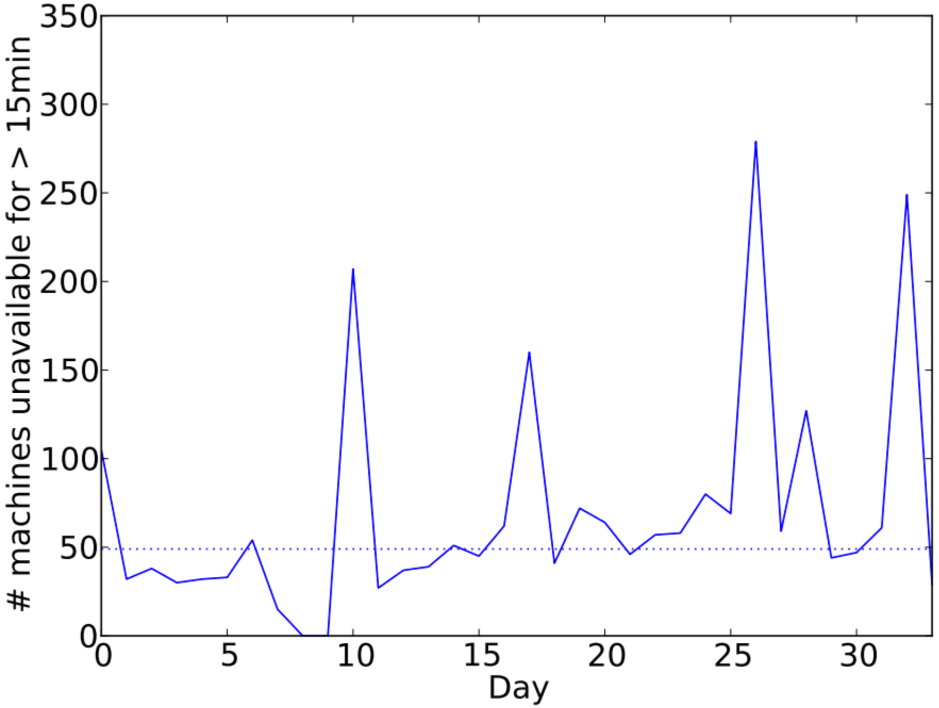
- We want to store a lot of data.
- Think:
  - Facebook HDFS
  - Windows Azure
  - Google Colossus
- We want all the data to be available at all times.
  - Even grumpy cat

Source: the internet

Data might be unavailable



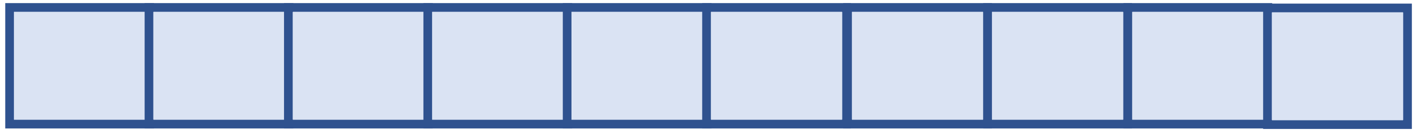
Ford et al. USENIX OSDI 2010
Study at Google



Rashmi at al. USENIX HotStorage 2013
Study on Facebook Warehouse Cluster

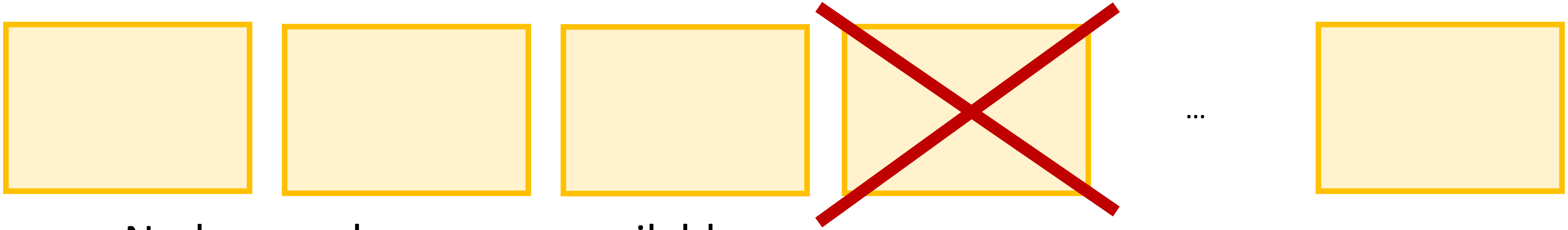
Formally

- For the rest of this talk, data look like this:



A bunch of blocks.
Think of each block as holding a byte.

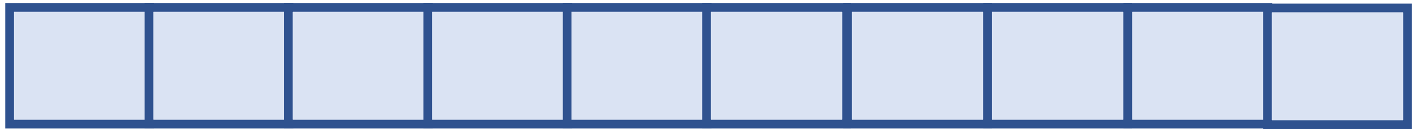
- We want to somehow encode the data and distribute it among n nodes



- Nodes may become unavailable.

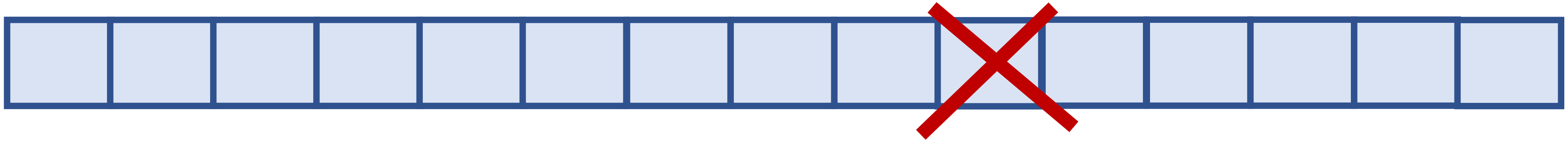
Formally

- For the rest of this talk, data look like this:



A bunch of blocks.
Think of each block as holding a byte.

- We think of encoding the data into n blocks:



and then distributing the blocks, one to each node.

- We will thus think of single blocks as becoming unavailable.

Usually,
nodes become unavailable one at a time

For example, in the Facebook Warehouse Cluster in 2013:

Number of missing blocks	Percent of stripes that have one or more block missing
1	98.08
2	1.87
3	0.036
4	9×10^{-6}
5	9×10^{-9}

But we do need to handle multiple failures sometimes.

Rashmi et al., USENIX Hot Storage 2013

Outline

1. Coding for distributed storage: what's the problem?
2. Coding for distributed storage: how do we solve the problem?
   - Try 1: replication
   - Try 2: classical erasure coding
   - Try 3: regenerating (MSR) codes
3. What can we do with regenerating codes?
   - Basic bounds
4. How about codes I know and love?
   - Reed-Solomon Codes
5. Future work/Open problems

Solution 1: replication

- Just make three (or more) copies of all of the data.
- This is very robust.

- Used (at least until recently) in
  - Hadoop Distributed File System (HDFS)
  - Google File System

- Downside: a lot of storage overhead.

S. Ghemawat, H. Gobioff and S.-T. Leung, "The Google file system", 2003

http://hadoop.apache.org/docs/current/

Solution 2: erasure coding

- Use and MDS code (like Reed-Solomon) to encode the data.
  - We'll see what that means on the next slide

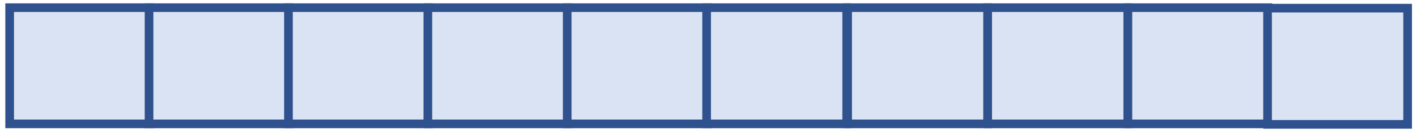
- This can substantially reduce the amount of overhead for the same amount of robustness.

- Used/supported by:
  - HDFS
  - Windows Azure
  - …

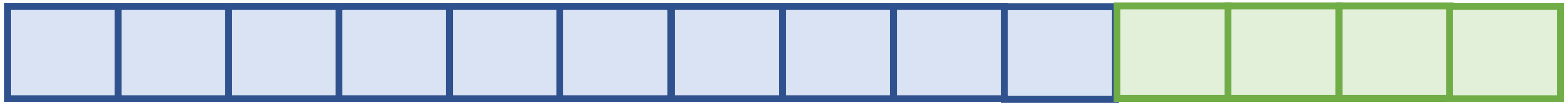
Solution 2: erasure coding

- Break up some data into k blocks:

Say each block stores a byte



- Encode these with a Maximum Distance Separable code into n blocks
  - For example, a Reed-Solomon Code
  - MDS means that any k encoded blocks are enough to recover the original data

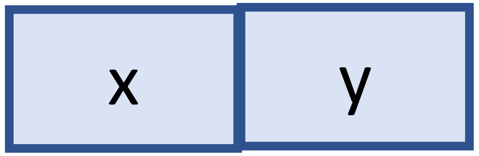


add n-k parity blocks

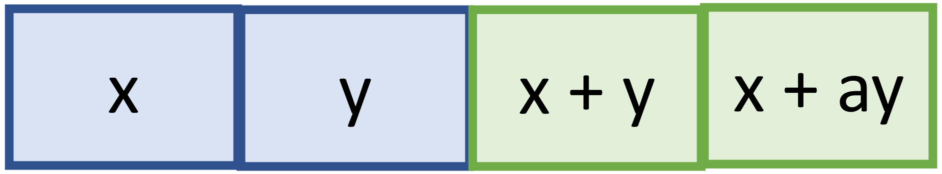
- Send each encoded block off to a different server.

Example: (2,4) MDS code

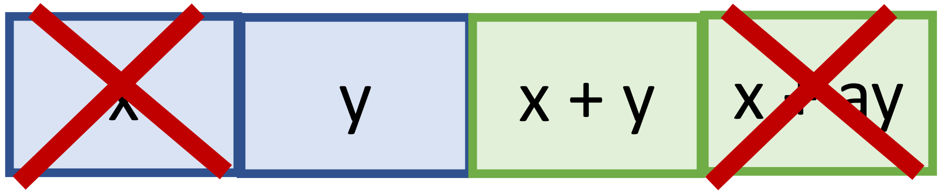
- Say I have two blocks of information:



- Encode this as four blocks:



- Now even if two blocks are erased, I can recover the original data.



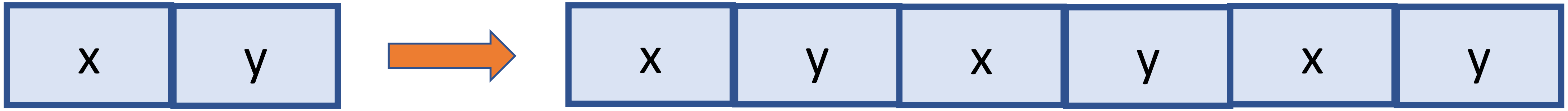
$y = y$

$x = -y + (x+y)$

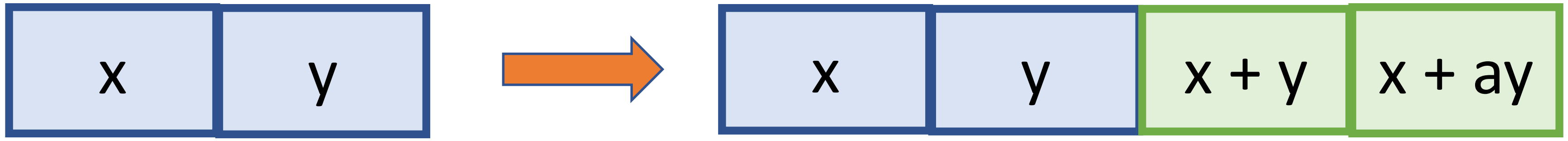
This works no matter which two blocks are erased.

Compare with repetition

- Repetition: 3X overhead to handle two erasures:



- MDS Erasure Coding: 2X overhead to handle two erasures:



That sounds great

- And it is!
- Information-theoretically, we can't do better than an MDS code when it comes to the trade-off between storage overhead and fault tolerance.

So what is this talk about?

Usually,

nodes become unavailable one at a time

For example, in the Facebook Warehouse Cluster in 2013:

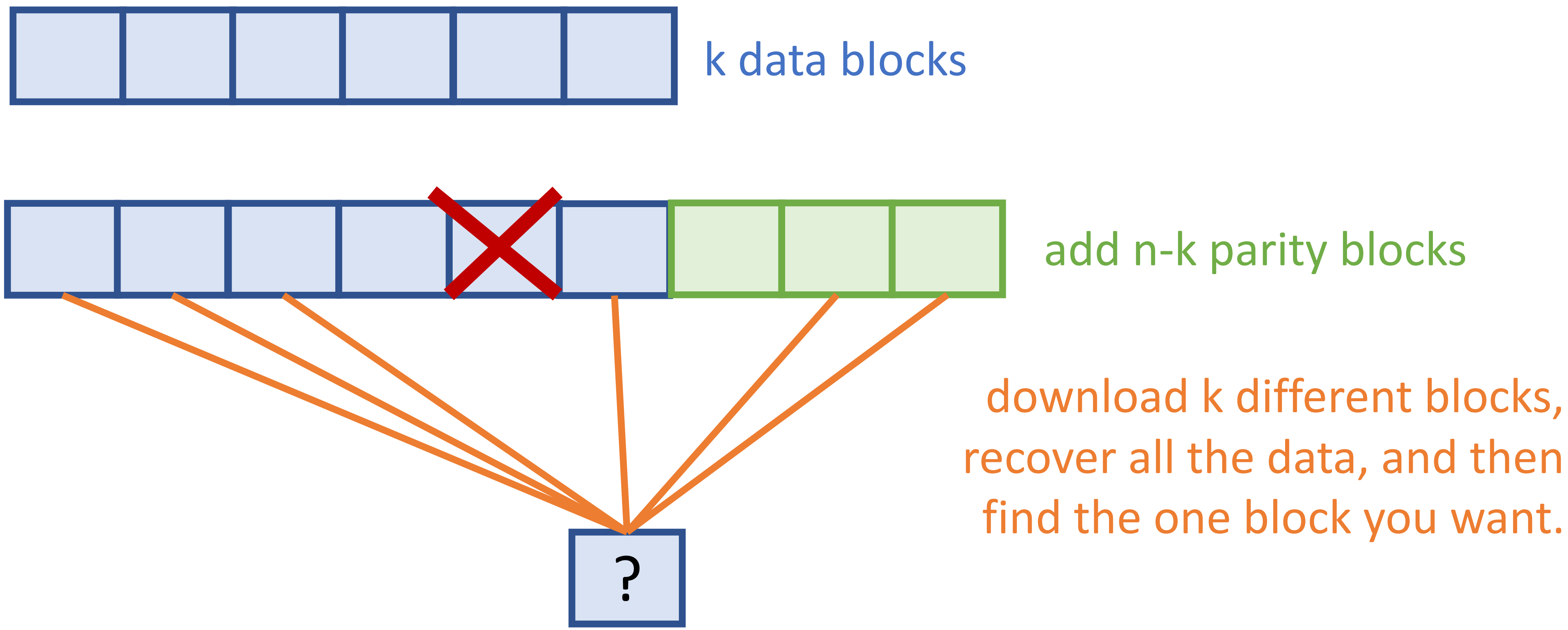
Number of missing blocks	Percent of stripes that have one or more block missing
1	98.08
2	1.87
3	0.036
4	9×10^{-6}
5	9×10^{-9}

But we do need to handle multiple failures sometimes.

Rashmi et al., USENIX Hot Storage 2013

An (n,k) MDS code protects against n-k failures

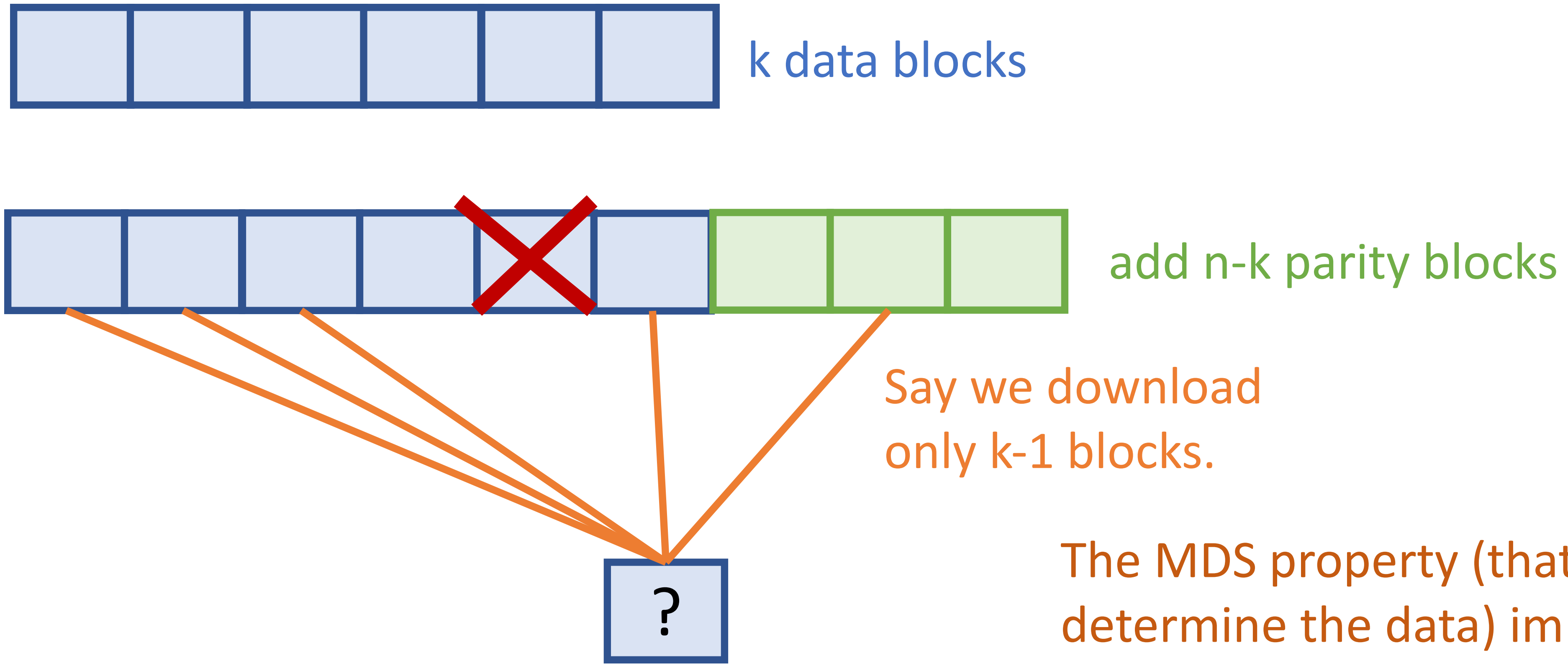
but how does it deal with just one?



This is very wasteful!

can we do better?

Not with an MDS code!



The MDS property (that any k blocks determine the data) implies that the missing block could be **anything.**

We can't do better with an MDS code!

Two solutions

1. Don't use an MDS code
   - This is a reasonable option!
   - Many approaches do this.
   - I'm not going to talk about them.

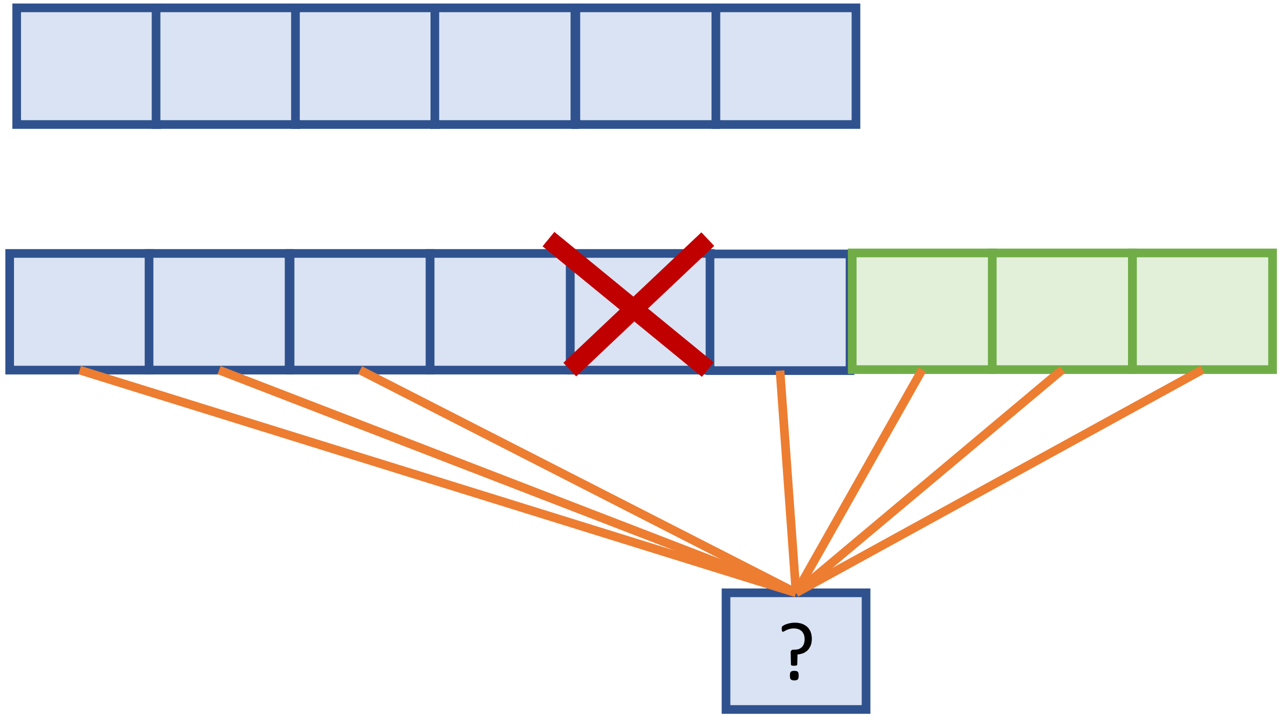
2. Change what we mean by "better."
   - What was the problem?
     - **Network bandwidth**
   - What can't we improve?
     - **Number of nodes we contact**

These aren't necessarily the same.

*Jargon: I'm going to be talking only about Minimum Storage Regenerating (MSR) codes.

Solution 3: Regenerating Codes

- Still MDS codes
  - At least for this talk.*
- But they have an additional property:
  - They allow for low-bandwidth repair of a **single failure**.



Contact more than k nodes…but download less than a whole block from each!

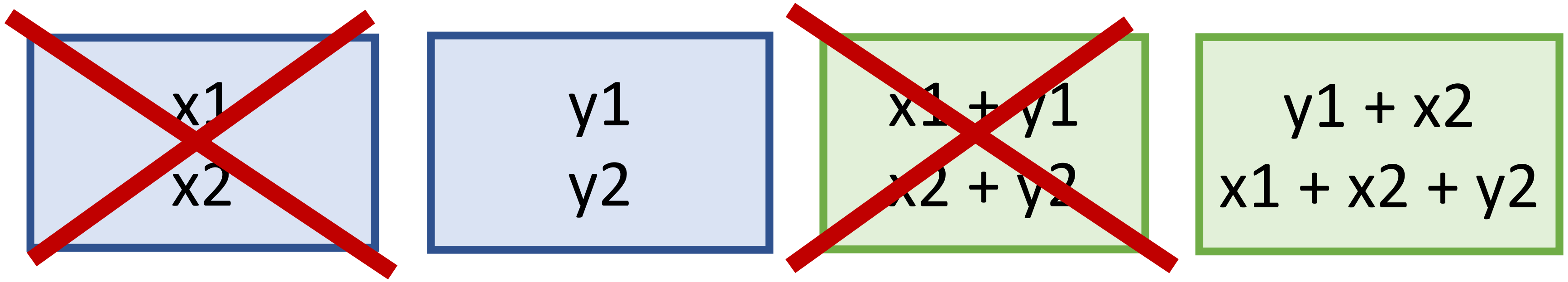
See the Erasure Coding for Distributed Storage Wiki
http://storagewiki.ece.utexas.edu/doku.php
for lots more information!

In particular, the nice survey: Dimakis et al. "A Survey on Network Codes for Distributed Storage" 2011.

All addition is mod 2

Example: regenerating codes

- Each block stores two bits:



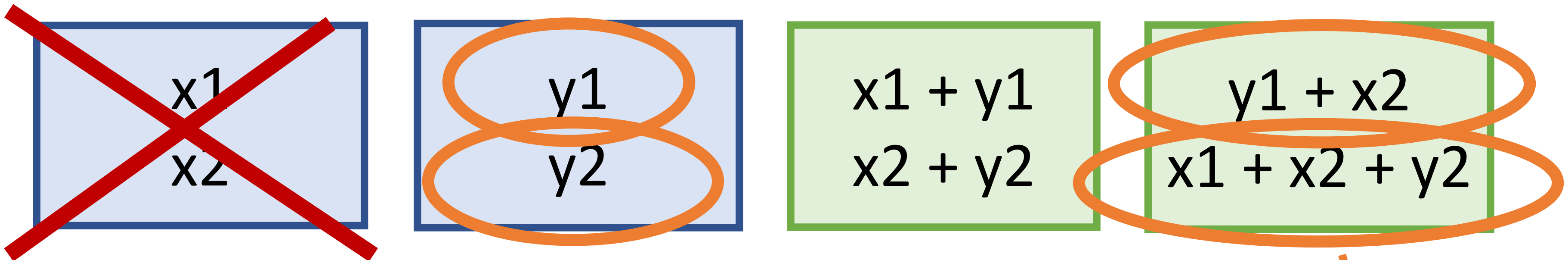
x1 = y1 + y2 + (y1 + x2) + (x1 + x2 + y2)

x2 = y1 + (y1 + x2)

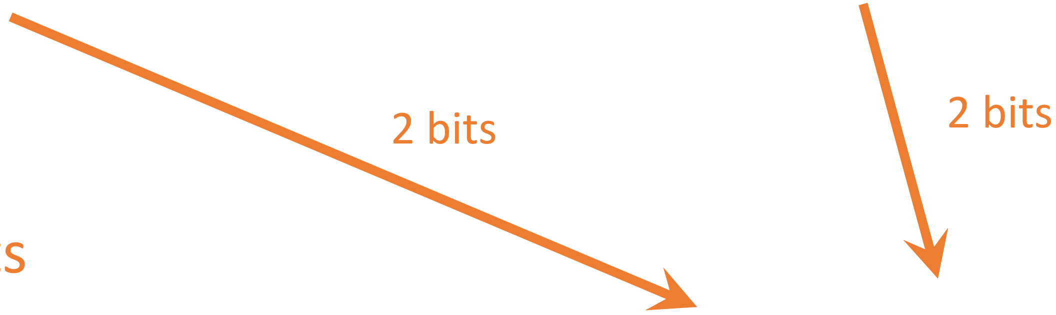
- This is still MDS
  - Can recover the data from any two failures.
  - Notice that this requires four bits of information.

Example: regenerating codes

- Each block stores two bits:



- With just one failure…
  - Naively still use four bits



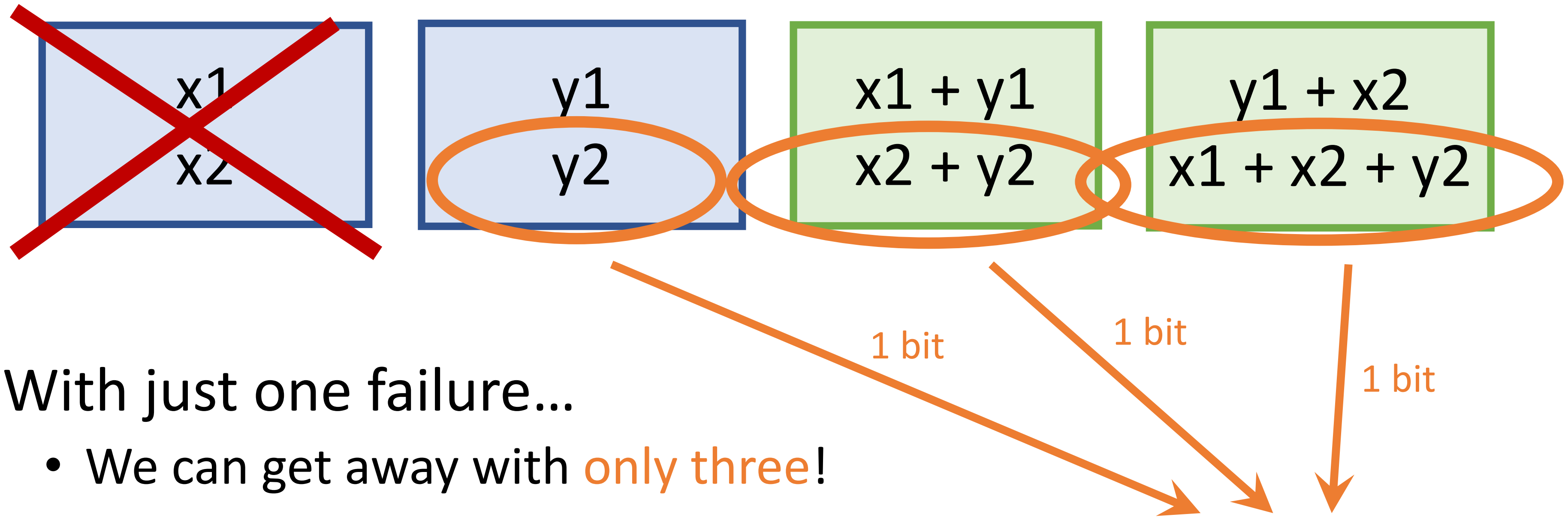
x1 = y1 + y2 + (y1 + x2) + (x1 + x2 + y2)

x2 = y1 + (y1 + x2)

All addition is mod 2

Example: regenerating codes

- Each block stores two bits:



- With just one failure…
  - We can get away with only three!

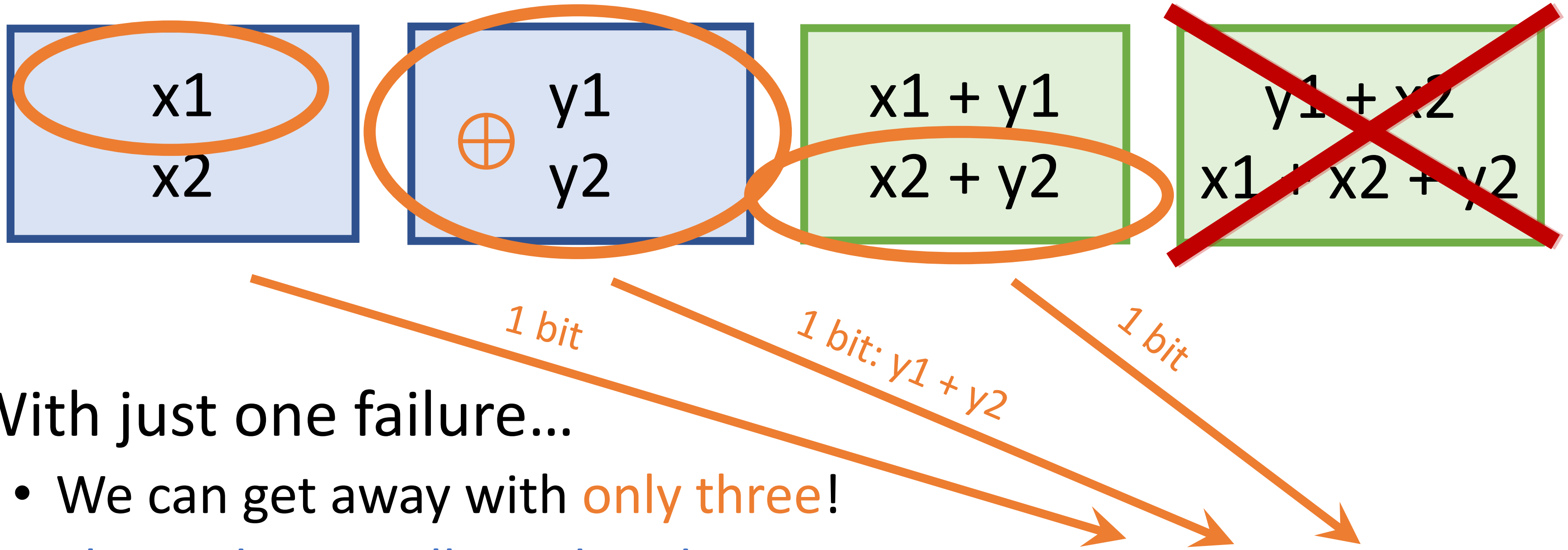
x1 = (x2 + y2) + (x1 + x2 + y2)

x2 = y2 + (x2 + y2)

All addition is mod 2

Example: regenerating codes

- Each block stores two bits:



- With just one failure…
  - We can get away with only three!
  - The nodes are allowed to do some local computation

y1 + x2 = (y1 + y2) + (x2 + y2)

x1 + x2 + y2 = x1 + (x2 + y2)

Regenerating codes

- Same amount of storage overhead as MDS codes
- Much less bandwidth required to repair a single node
  - (Than the naïve MDS scheme)

- Introduced by Dimakis et al. in 2010
- Since then, lots of work, both on the theory side and the systems side
  - There exist good constructions
  - In several parameter regimes, we know the "right" trade-off between bandwidth, storage overhead, and redundancy.

Dimakis et al. "Network coding for distributed storage systems." IEEE Trans. IT, 2010

Erasure Coding for Distributed Storage Wiki: http://storagewiki.ece.utexas.edu/doku.php

Outline

1. Coding for distributed storage: what's the problem?
2. Coding for distributed storage: how do we solve the problem?
   - Try 1: replication
   - Try 2: classical erasure coding
   - Try 3: regenerating (MSR) codes
3. What can we do with regenerating codes?
   - Basic bounds
4. How about codes I know and love?
   - Reed-Solomon Codes
5. Future work/Open problems

Some lower bounds

- $b \geq t \cdot \left(\frac{n-1}{n-k}\right)$

  We want to recover t bits, so we can't do better than t. If t is big, this is the bottleneck.

- $b \geq t + k - 1$

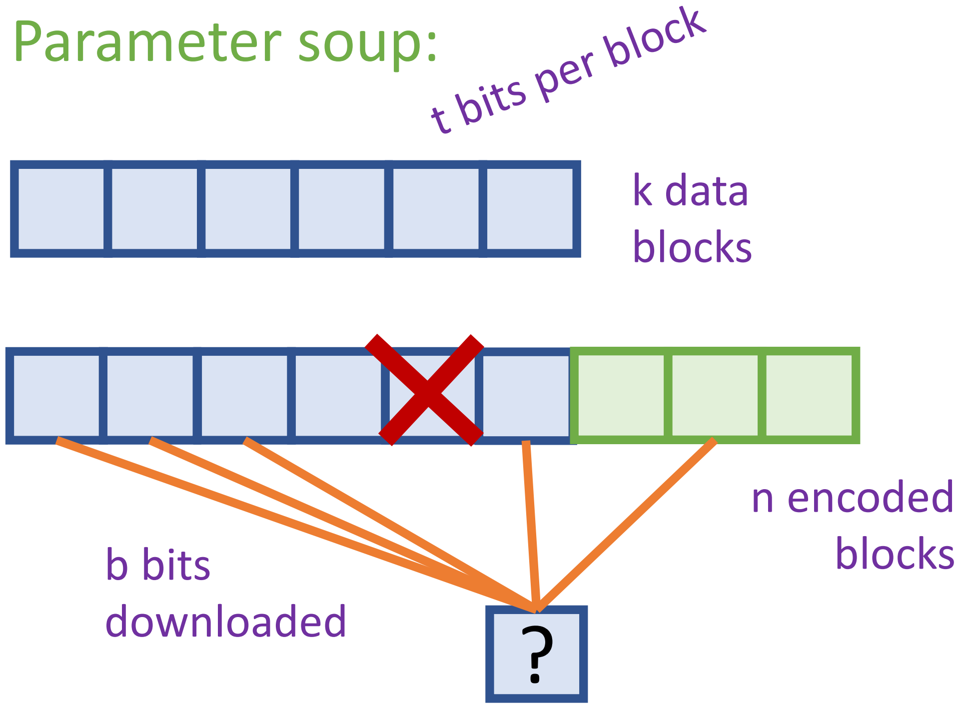
  The MDS property implies we need to at least contact k nodes. If k is big, this is the bottleneck.

- $b \geq (n-1) \cdot \log_2 \left(\frac{n-1}{n-k}\right)$

  You need to download at least some amount (on average) from each non-damaged node.

[Dimakis et al. 2010] [Guruswami, W. 2017]

Parameter soup:



Reasonable settings:

- t = 8
- n = 14
- k = 10
- b = hopefully way less than kt = 80!

The first bound says $b \geq 26$ in this setting.

Upper bounds?

- $b \geq t \cdot \left(\frac{n-1}{n-k}\right)$ **There are constructions that approach this as t gets really big.**

  We want to recover t bits, so we can't do better than t. If t is big, this is the bottleneck.

- $b \geq t + k - 1$ **These exist for k-3 < t < n-k**

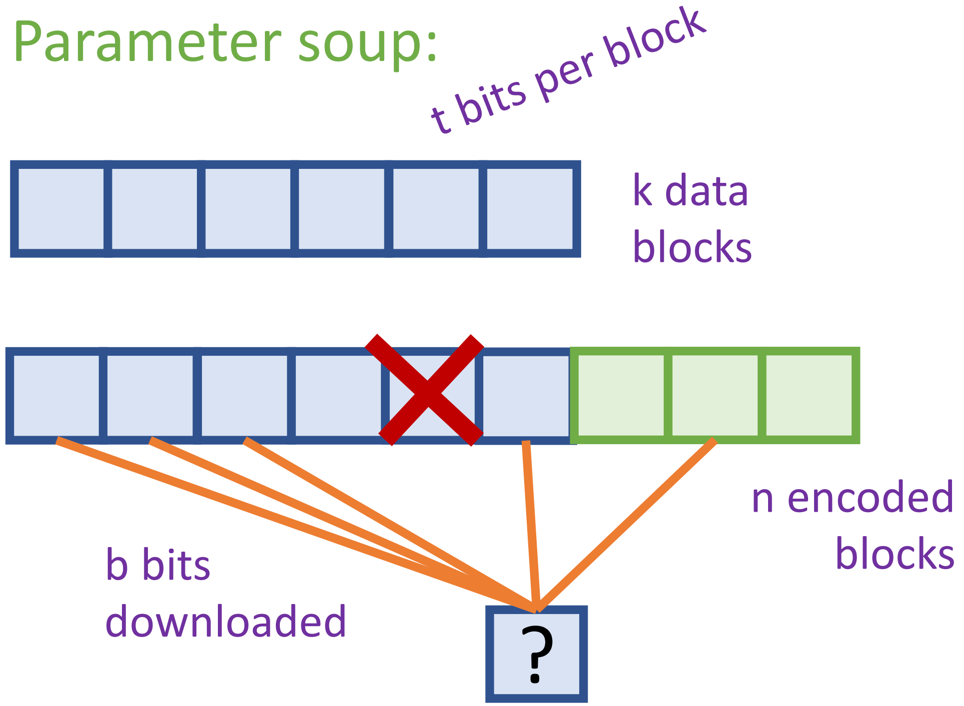
  The MDS property implies we need to at least contact k nodes. If k is big, this is the bottleneck.

- $b \geq (n-1) \cdot \log_2\left(\frac{n-1}{n-k}\right)$ **We can match this when t is log(n).**

  You need to download at least some amount (on average) from each non-damaged node.

[Cadambe et al. 2013] [Sasidharan et al. 2015]
[Dimakis et al. 2010] [Guruswami, W. 2017] [Shah et al. 2012] [Rashmi et al. 2009]

Parameter soup:



Reasonable settings:

- t = 8
- n = 14
- k = 10
- b = hopefully way less than kt = 80!

The first bound says $b \geq 26$ in this setting.

Understanding all the trade-offs is an active area of research!

screenshots from UT distributed storage wiki:

Regenerating Codes

Here we list papers that study the problem of minimizing repair communication (aka. Repair Bandwidth).

* General introduction to the Repair Problem.

Video tutorial on Regenerating Codes

Explicit Constructions of High-Rate MDS Array Codes With Optimal Repair Bandwidth

M. Ye, A. Barg,
IEEE Transactions on Information Theory (Volume: 63, Issue: 4, April 2017)
IEEExplore

A Connection Between Locally Repairable Codes and Exact Regenerating Codes

T. Ernvall, T. Westerbäck, R. Freij-Hollanti and Camilla Hollanti,
Proceedings of IEEE International Symposium on information Theory (ISIT), 2016.
IEEExplore arXiv

On MBR codes with replication

M Nikhil Krishnan, P. Vijay Kumar,
Proceedings of IEEE International Symposium on information Theory (ISIT), 2016.
IEEExplore

A high-rate MSR code w... polynomial sub-packetization level

B. Sasidhar... nd P. V. Kumar,
Proc... al Sympos... formation Theory (ISIT), 2015.

...air linear regenerating codes for the case d= k= n− 1.

...ISIT), 2015 IEEE International Symposium on. IEEE, 2015.

...racterization via new bounds

...plications Workshop (ITA), 2015. IEEE, 2015.
...pdf

... Storage Systems

... (ISIT), 2014.

...ting Codes

... Issue on Communication Methodologies for

...ing Challenges in Cloud Computing

A Piggybacking Design Framework for Read-and Download-efficient Distributed Storage Codes

K. V. Rashmi, Nihar B. Shah, Kannan Ramchandran
arXiv

On Minimizing Data-read and Download for Storage-Node Recovery

Nihar B. Shah
pdf

Repairing Multiple Failures in the Suh-Ramchandran Regenerating Codes

J. Chen, Kenneth W. Shum
arXiv

Exact-Repair Regenerating Codes Via Layered Erasure Correction and Block Designs

C Tian, V Aggarwal, VA Vaishampayan
arXiv

On Weak Dress Codes for Cloud Storage

MK Gupta, A Agrawal, D Yadav
arXiv

High-Rate Regenerating Codes Through Layering

B Sasidharan, PV Kumar
arXiv

Repair for Distributed Storage Systems with Erasure Channels

Majid Gerami, and Ming Xiao
arXiv

Decentralized Minimum-Cost Repair for Distributed Storage Systems

Majid Gerami, Ming Xiao, Carlo Fischione, and Mikael Skoglund
arXiv

Update-Efficient Error-Correcting Regenerating Codes

Yunghsiang S. Han, Hong-Ta Pai, Rong Zheng, and Pramod K. Varshney
arXiv

Update-Efficient Regenerating Codes with Minimum Per-Node Storage

Yunghsiang S. Han, Hong-Ta Pai, Rong Zheng, and Pramod K. Varshney
arXiv

Optimal Locally Repairable and Secure Codes for Distributed Storage Systems

Ankit Singh Rawat, O. Ozan Koyluoglu, Natalia Silberstein, Sriram Vishwanath
arXiv

Secure Cooperative Regenerating Codes for Distributed Storage Systems

O. Ozan Koyluoglu, Ankit Singh Rawat, Sriram Vishwanath
arXiv

Analysis and Construction of Functional Regenerating Codes with Uncoded Repair for Distributed Systems

A Network Coding Based Framework for Construction of Systematic Regenerating Codes for Distributed Storage

Swanand Kadhe, M. Girish Chandra, and Balaji Janakiram,
Status: Submitted to ACM Transactions on Storage, Apr 2011.
Preprint.pdf More

Optimal-Cost Repair in Multi-hop Distributed Storage Systems

Majid Gerami, Ming Xiao, Mikael Skoglund
Proceedings of IEEE International Symposium on information Theory (ISIT), 2011
pdf More

Quasi-cyclic Minimum Storage Regenerating Codes for Distributed Data Compression

B. Gastón, J. Pujol and M. Villanueva
Proceedings of the Data Compression Conference (DCC), 2011
IEEEXplore More

Cooperative Regenerating Codes for Distributed Storage Systems

Kenneth W. Shum
Presented in IEEE International Conf. on Comm. (ICC) 2011.
arXiv More

Enabling Node Repair in Any Erasure Code for Distributed Storage

K. V. Rashmi, Nihar B. Shah and P. Vijay Kumar
IEEE International Symposium on Information Theory (ISIT) 2011.
arXiv More

ExR: A Scheme for Exact Regeneration of a Failed Node in a Distributed Storage System

Balaji Janakiram, Swanand Kadhe, and M. Girish Chandra,
Proceedings of Annual International Conference on Advances in Distributed and Parallel Computing (ADPC), Nov 2010.
pdf More

Distributed Storage Codes with Repair-by-Transfer and Non-achievability of Interior Points on the Storage-Bandwidth Tradeoff

Nihar B. Shah, K. V. Rashmi, P. Vijay Kumar, and Kannan Ramchandran
Nov, 2010.
arXiv More

Distributed Storage Codes Meet Multiple-Access Wiretap Channels

D. S. Papailiopoulos and A. G. Dimakis
Allerton, September 2010.
pdf More

Fractional Repetition Codes for Repair in Distributed Storage Systems

S. El Rouayheb and K. Ramchandran
Allerton, September 2010.
pdf More

Beyond Regenerating Codes

A.-M. Kermarrec, N. Le Scouarnec, and Straub,
INRIA Research Report, September 2010.
pdf
More Superseded by *Repairing Multiple Failures with Coordinated and Adaptive Regenerating Codes*.

A Flexible Class of Regenerating Codes for Distributed Storage

N. B. Shah, K. V. Rashmi, P. Vijay Kumar, and K. Ramchandran,
in Proc. 2010 IEEE Int. Symp. Info. Theory (ISIT), June 2010.
IEEE Xplore More

Self-repairing Homomorphic Codes for Distrib...

F. Oggier, A. Datta
in Proc. 2011 IEEE International Conference on C...
Arxiv, July 2010.
Note: A substantially extended version of this wo...
arXiv More

Explicit and Optimal Exact-Regenerating Code...

K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ra...
in Proc. 2010 IEEE Int. Symp. Info. Theory (ISIT),...
IEEE Xplore More

Cooperative Recovery of Distributed Storage S...

Yuchong Hu, Yinlong Xu, Xiaozhao Wang, Cheng...
IEEE J. on Selected Areas in Comm., vol. 28, no....
IEEE Xplore More

Double Circulant Minimum Storage Regenerat...

Bernat Gastón, and Jaume Pujol,
Status: Deprecated. New version called "Quasi-cy... Compression"
arXiv More

Distributed Data Storage with Minimum Storag... Asymptotically Equally Efficient

V. R. Cadambe, S. A. Jafar, H. Maleki,
in Proc. 2010 Wireless Network Coding (WINC) W...
arXiv More

A Fundamental Trade-off Between The Downlo... Systems

S. Akhlaghi, A. Kiani, and M. R. Ghanavati,
in Proc. 2010 IEEE International Symposium on N...
More

A Practical Network Coding Approach for Pee...

M. Martaló, M. Picone, R. Bussandri, and Michele...
in Proc. 2010 IEEE International Symposium on N...
IEEE Xplore More

Optimal Exact-Regenerating Codes for Distrib... Construction

K. V. Rashmi, Nihar B. Shah, and P. Vijay Kumar
Results: Explicit codes for the MBR point for all fe... k, d ≥ 2k-2].
arXiv Poster, ISIT Recent Results, Austin, Jun...
IEEE Transactions on Information Theory, vol. 57,...
More

Interference Alignment in Regenerating Codes...

Nihar B. Shah, K. V. Rashmi, P. Vijay Kumar, and...
Journal version of the resulsts which appeared in...
arXiv
IEEE Transactions on Information Theory, April 2...
More

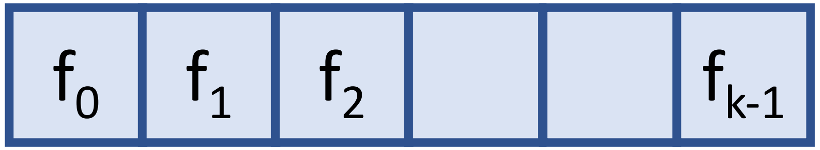
Outline

1. Coding for distributed storage: what's the problem?
2. Coding for distributed storage: how do we solve the problem?
   - Try 1: replication
   - Try 2: classical erasure coding
   - Try 3: regenerating (MSR) codes
3. What can we do with regenerating codes?
   - Basic bounds
4. How about codes I know and love?
   - Reed-Solomon Codes
5. Future work/Open problems

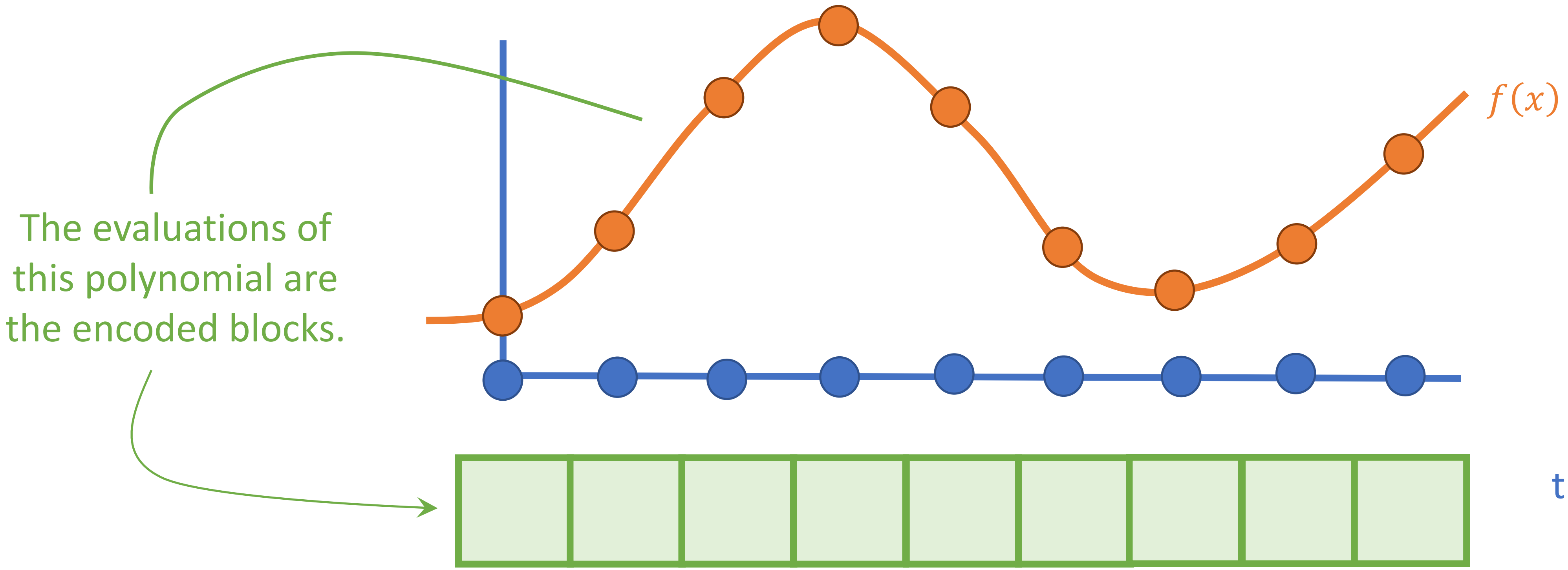
Reed-Solomon Codes

Classical solution for erasure coding

Plus lots of other things!



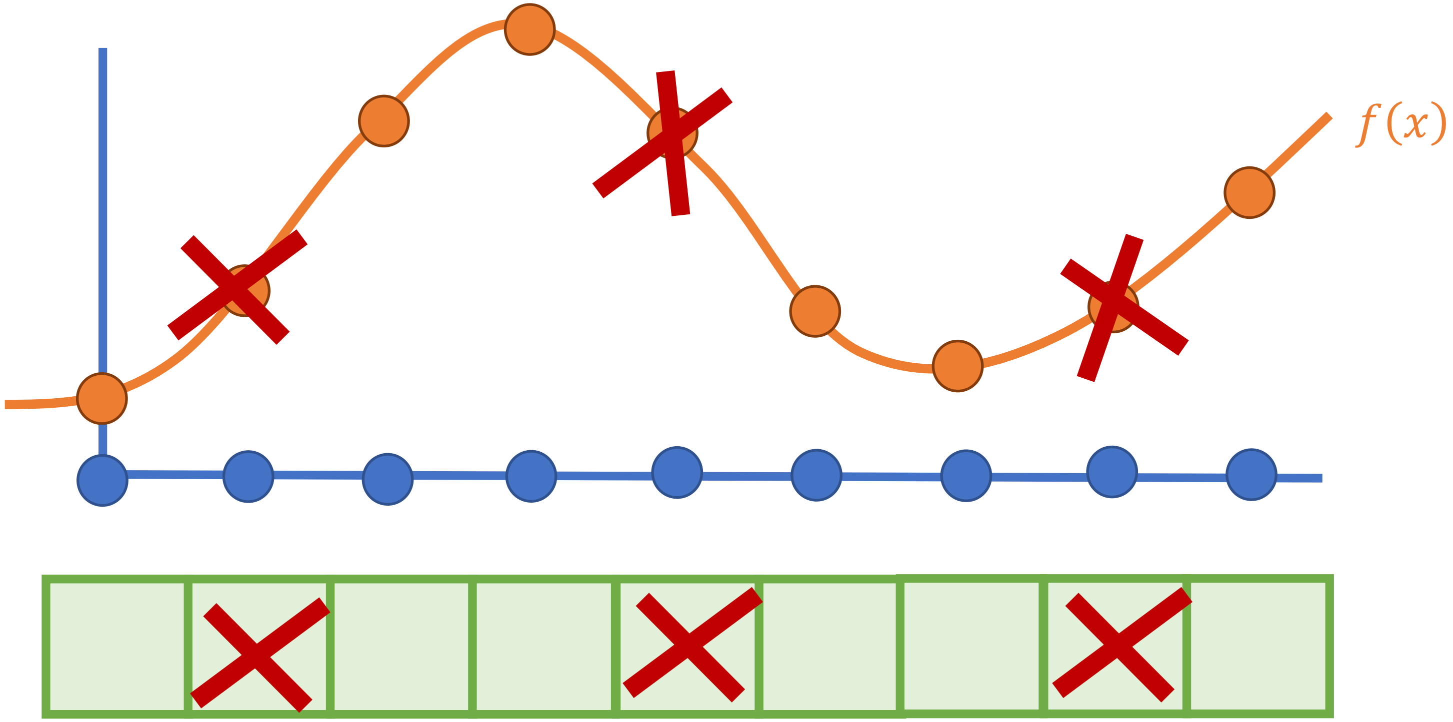
$$f(x) = f_0 + f_1 \cdot x + f_2 \cdot x^2 + \cdots + f_{k-1} \cdot x^{k-1}$$



Technically, the things in these boxes are elements of a **finite field**

Reed-Solomon Codes are MDS codes

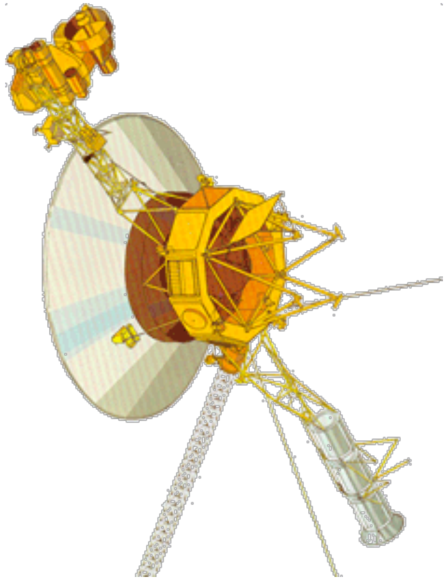
- Any k evaluations of a degree k-1 polynomial suffices for reconstruction.



Reed-Solomon codes are the standard
for erasure coding in distributed storage

- Microsoft Azure uses RS(9,6)
- HDFS supports RS(14,10)

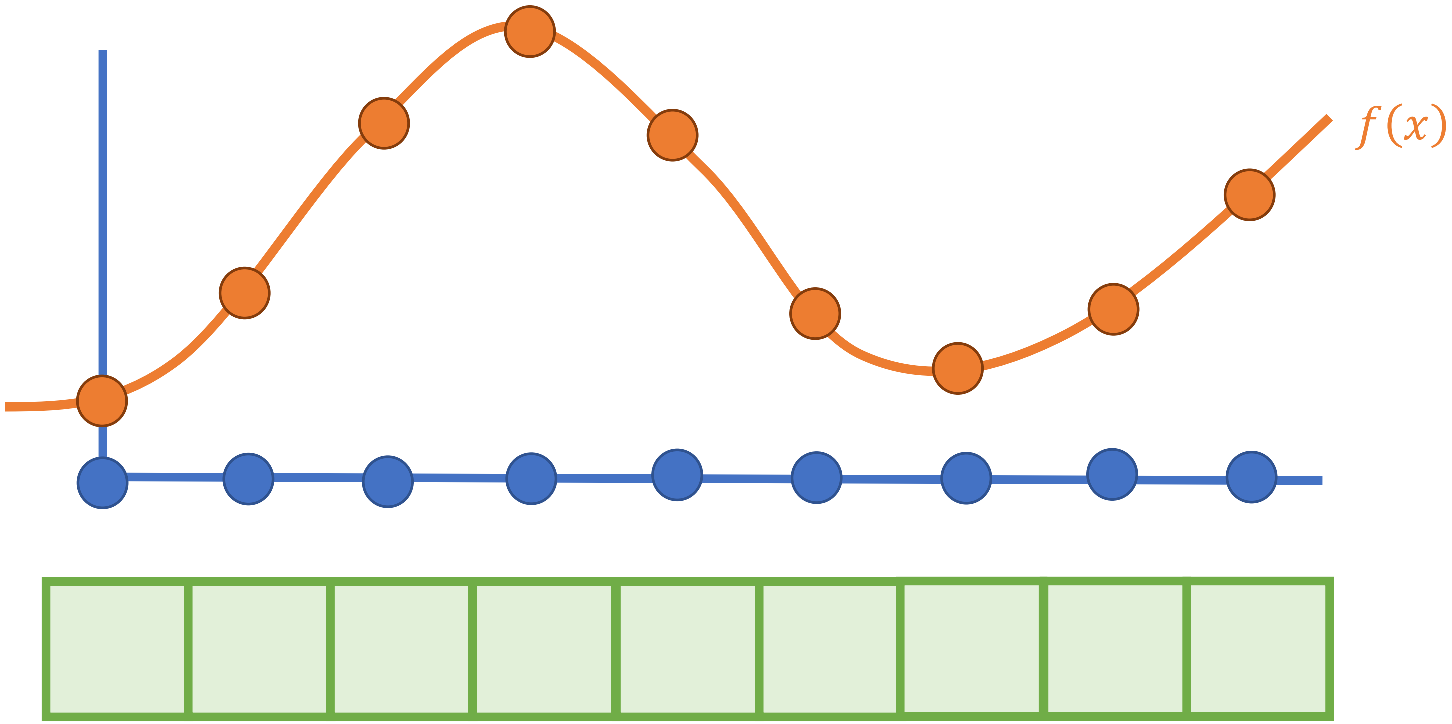
- Reed-Solomon Codes are:
  - Standard
  - Very efficient to manipulate
  - Really nice algebraic structure!



Also RS codes are used for all sorts of other stuff too

Can Reed-Solomon Codes be good regenerating codes?

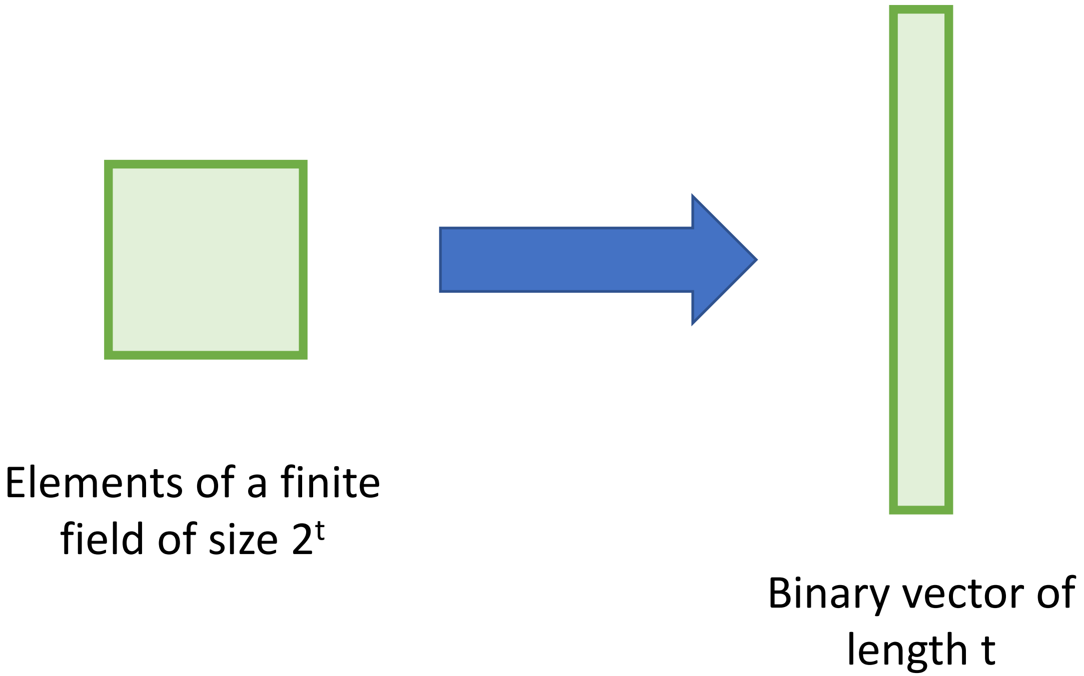
- At first, this doesn't make sense.



These things are elements of a finite field. The example we saw needs them to be binary vectors.

First try

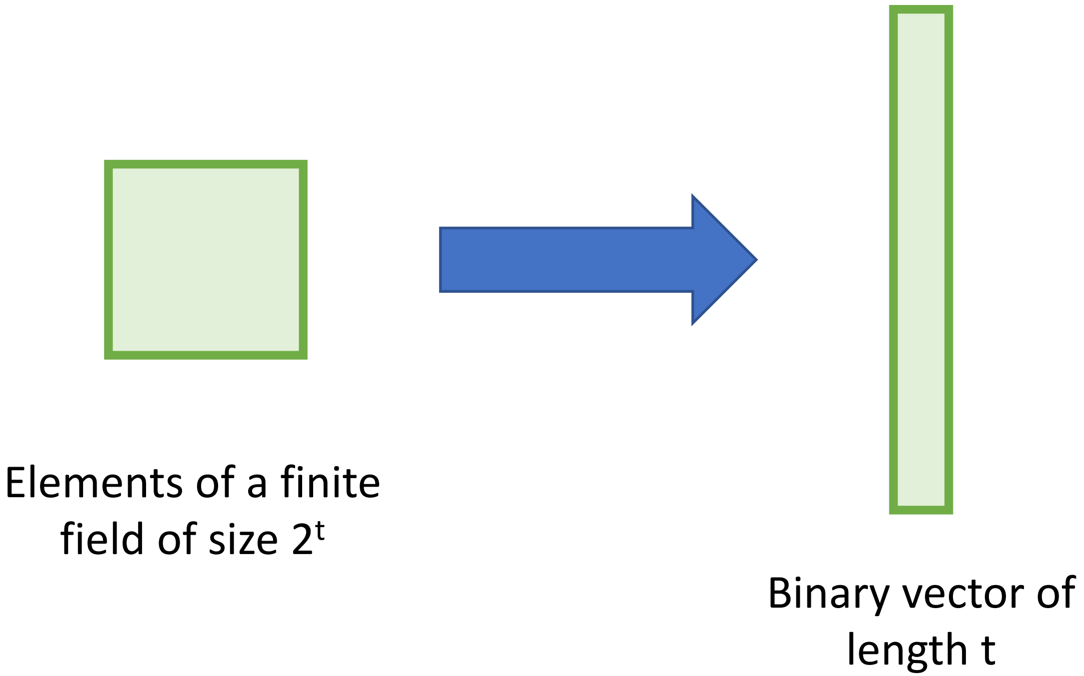
- Define an arbitrary mapping:



The problem with this is that it destroys the nice algebraic structure of Reed-Solomon Codes.

Next try

- This mapping doesn't have to be arbitrary



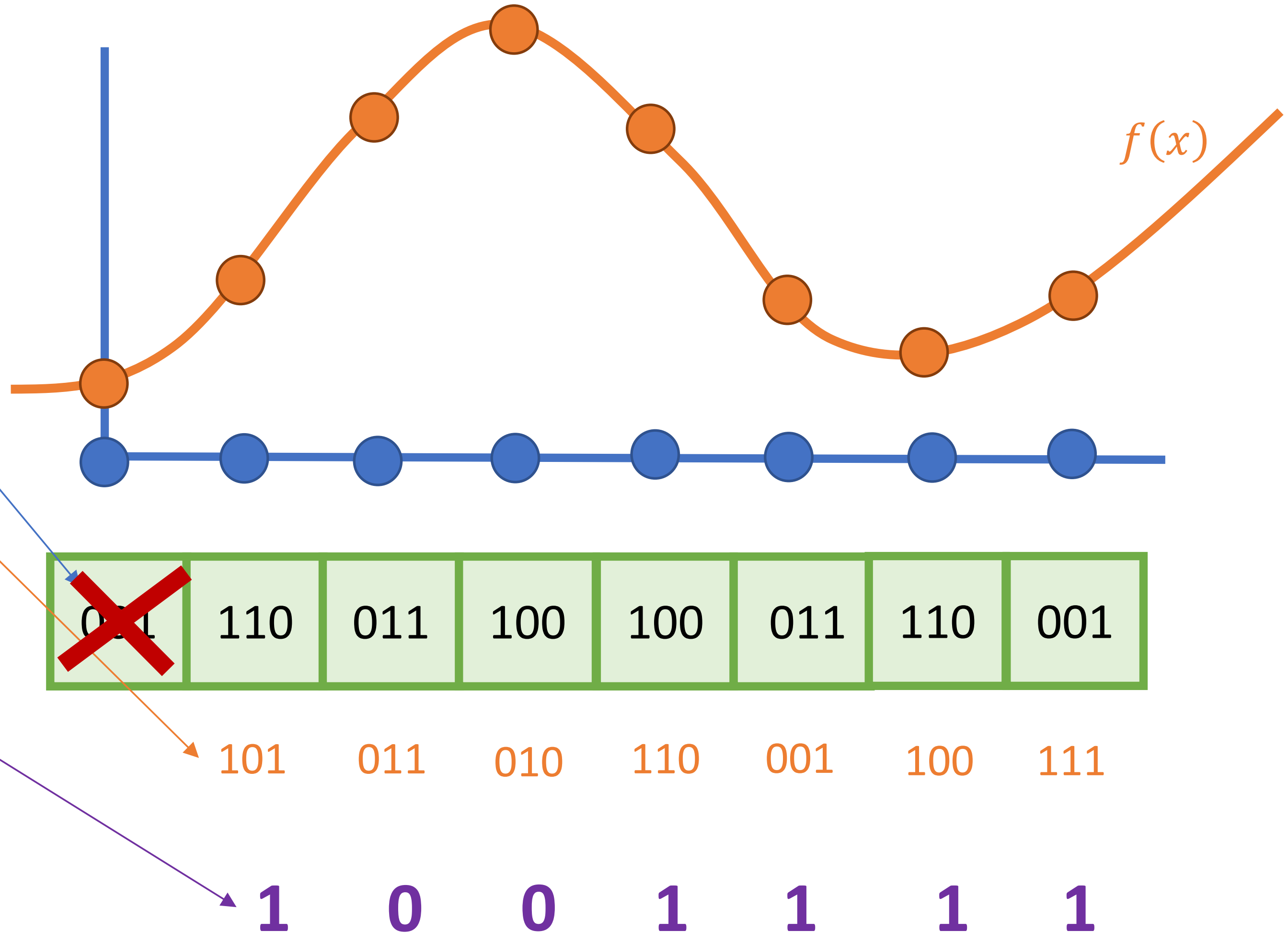
- Actually the finite field of size 2^t is a vector space over the finite field of size 2.
- This means that there's a way to define this mapping that plays nice with the algebra.

This is a pretty simple observation
but it turns out to be pretty powerful

- Reed-Solomon codes themselves are optimal regenerating codes in some parameter regimes!
  - Guruswami, W., STOC 2016, IEEE Trans. IT, 2017
- Follow-up work has extended this to more parameter regimes.
  - Ye, Barg, ISIT 2016
- More follow-up work has extended this to multiple failures.
  - Dau, Duursma, Kiah, Milenkovic, ISIT 2017

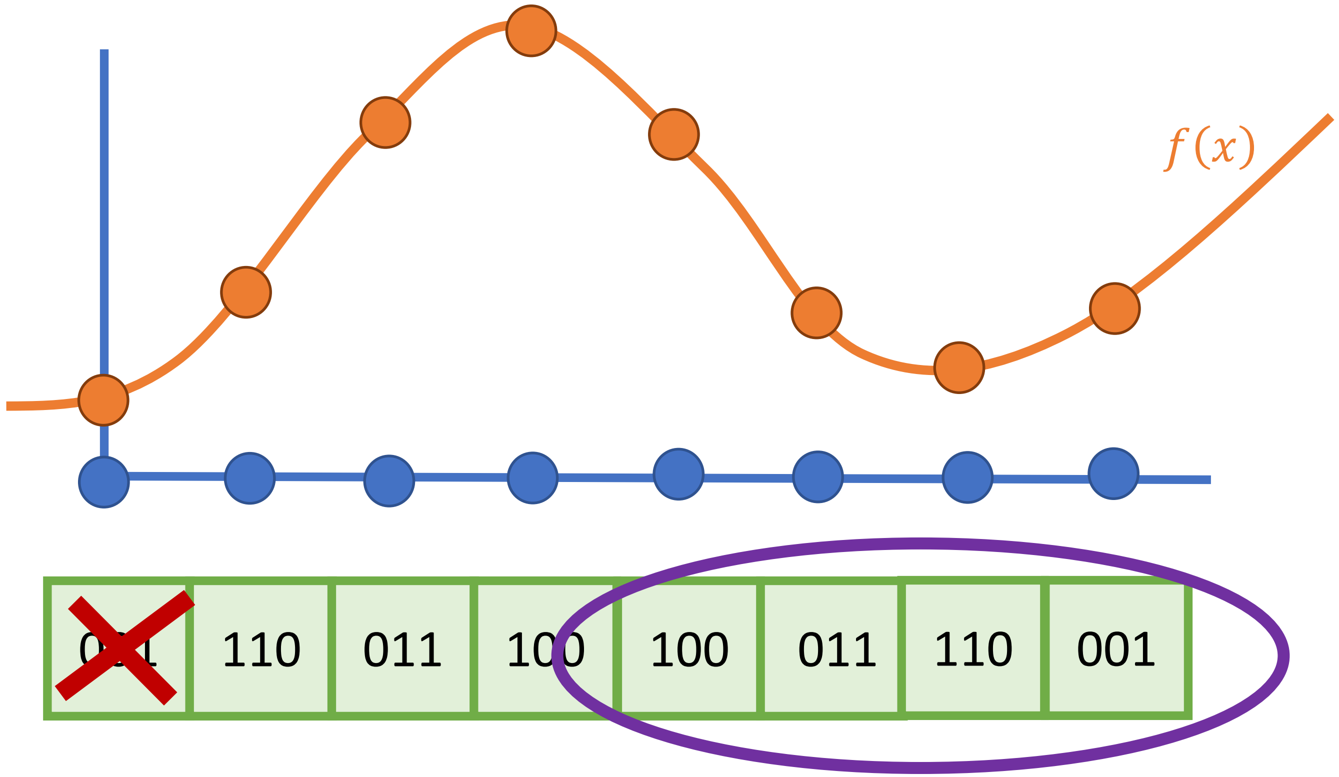
What does this scheme actually look like?

- Say n=8, k=4, t=3
  - We work over the finite field of size $2^t = 8$.
- Each element is stored as a **vector of length 3**.
- Say node 0 is going to fail. This determines a **repair scheme**.
- To do the repair:
  - each node computes the dot product of its contents with the repair vector
  - returns **the resulting bit.**
- The system uses algebra to reconstruct the missing value from these **7 bits.**



Compare to the naïve scheme

- Say n=8, k=4, t=3
  - We work over the finite field of size $2^t = 8$.
- Each element is stored as a **vector of length 3**.
- To do the repair:
  - download the complete contents of any four nodes.
- The system uses algebra to reconstruct the missing value from these **12 bits.**



More generally

with some jargon

- A rate ½ RS code can repair any missing node using only one bit from every surviving node.
- A rate $1 - \epsilon$ RS code can repair any missing node using only $\log_2(1/\epsilon)$ bits from every surviving node.
- This is optimal for MDS codes with linear repair schemes.

Guruswami, W., STOC 2016, IEEE Trans. IT, 2017

Outline

1. Coding for distributed storage: what’s the problem?
2. Coding for distributed storage: how do we solve the problem?
   - Try 1: replication
   - Try 2: classical erasure coding
   - Try 3: regenerating (MSR) codes
3. What can we do with regenerating codes?
   - Basic bounds
4. How about codes I know and love?
   - Reed-Solomon Codes
5. Future work/Open problems

New Directions

- For regenerating codes in general:
  - Pinning down all of the trade-offs.
  - Coming up with good constructions.
  - These ideas seem like they might be useful beyond distributed storage.
- For RS codes as regenerating codes in particular:
  - Repair-by-transfer?
  - Extending these techniques to other algebraic codes.
  - These ideas seem like they might be useful beyond distributed storage.

Thanks for listening!

Questions?

Mary Wootters
marykw@stanford.edu