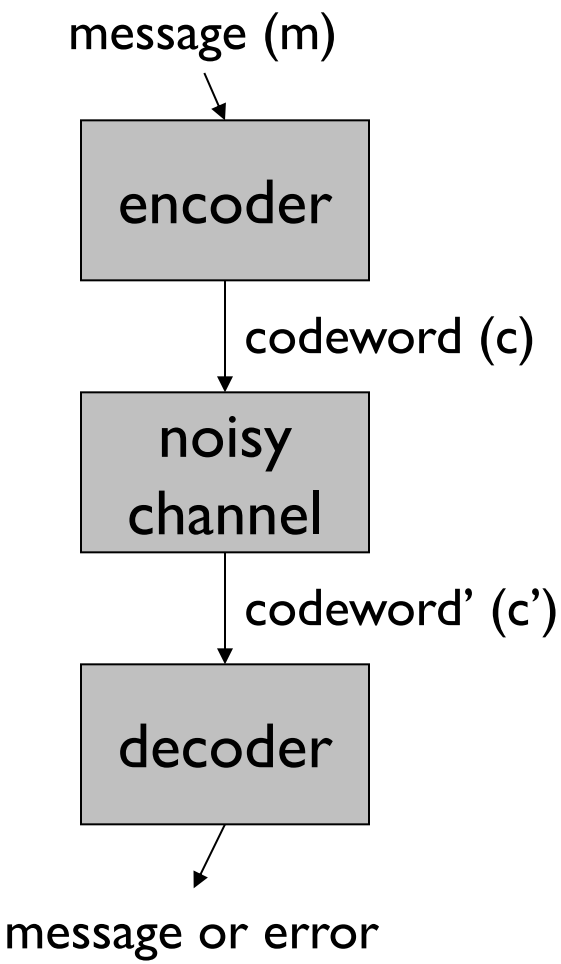
Recap: General Model



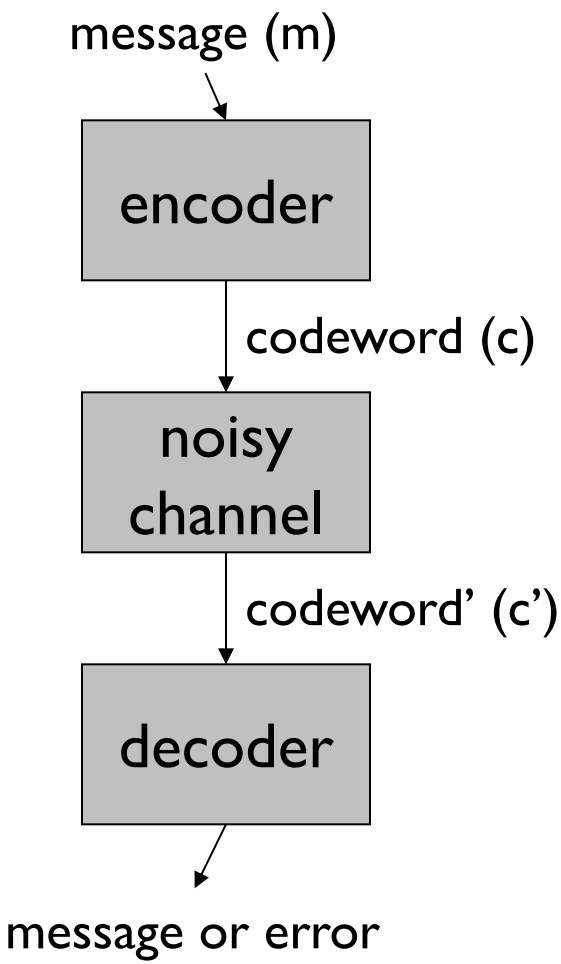
"Noise" introduced by the channel:

- changed fields in the codeword vector (e.g. a flipped bit).
  - Called **errors**
- missing fields in the codeword vector (e.g. a lost byte).
  - Called **erasures**

How the decoder deals with errors and/or erasures?

- **detection** (only needed for errors)
- **correction**

Recap: Block Codes



- Each message and codeword is of fixed size
- Notation:

k = |m| ← "dimension of the code"

length of the message

n = |c| ← "length of the code"

length of the codeword

C = "code" = set of codewords

15-750: Graduate Algorithms

Algorithms for coding
(Error Correcting Codes)

- Real-world application example
  - Cluster storage systems
- Continue with basic concepts
  - minimum distance, systematic property, linear codes, generator and parity-check matrices, Singleton bound
- Reed-Solomon codes

15-750: Graduate Algorithms

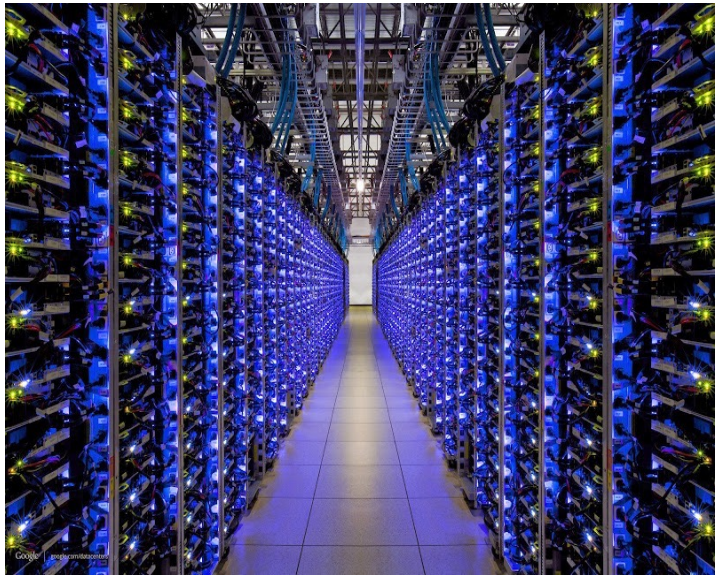
Algorithms for coding
(Error Correcting Codes)

- Real-world application example
  - Cluster storage systems
- Continue with basic concepts
  - minimum distance, systematic property, linear codes, generator and parity-check matrices, Singleton bound
- Reed-Solomon codes

Large-scale distributed storage systems

1000s of interconnected servers

100s of petabytes of data



- Commodity components
- Software issues, power failures, maintenance shutdowns

Large-scale distributed storage systems

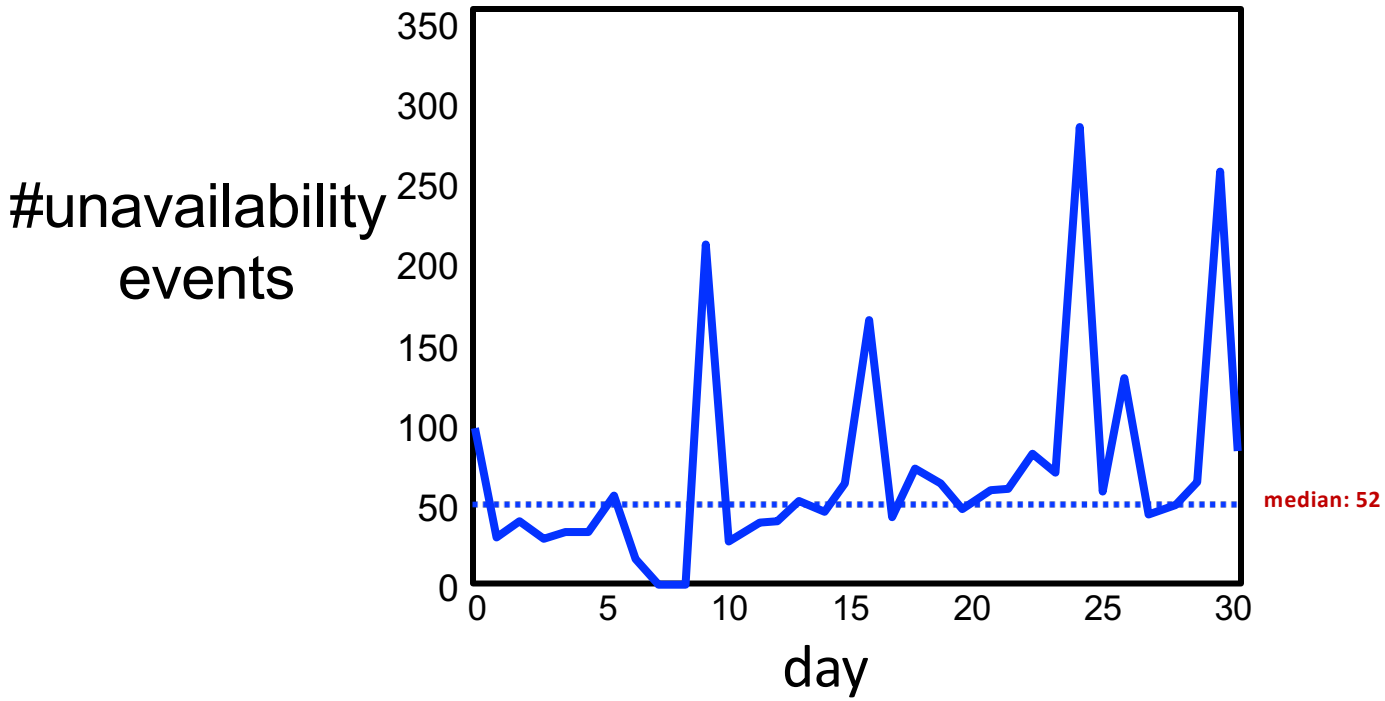

1000s of interconnected servers

Unavailabilities are the norm rather than the exception

- Commodity components
- Software issues, power failures, maintenance shutdowns

Facebook analytics cluster in production: unavailability statistics

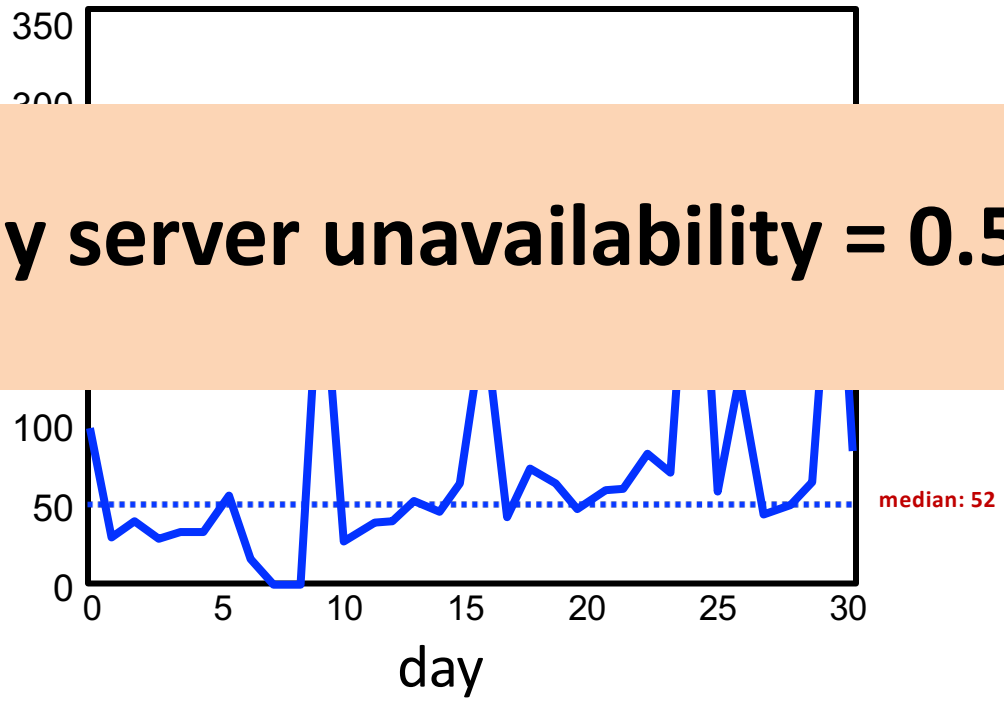
- Multiple thousands of servers
- Unavailability event: server unresponsive for > 15 min



[Rashmi, Shah, Gu, Kuang, Borthakur, Ramchandran, USENIX HotStorage 2013 and ACM SIGCOMM 2014]

Facebook analytics cluster in production: unavailability statistics

- Multiple thousands of servers
- Unavailability event: server unresponsive for > 15 min



Daily server unavailability = 0.5 - 1%

[Rashmi, Shah, Gu, Kuang, Borthakur, Ramchandran, USENIX HotStorage 2013 and ACM SIGCOMM 2014]

Servers unavailable

↓

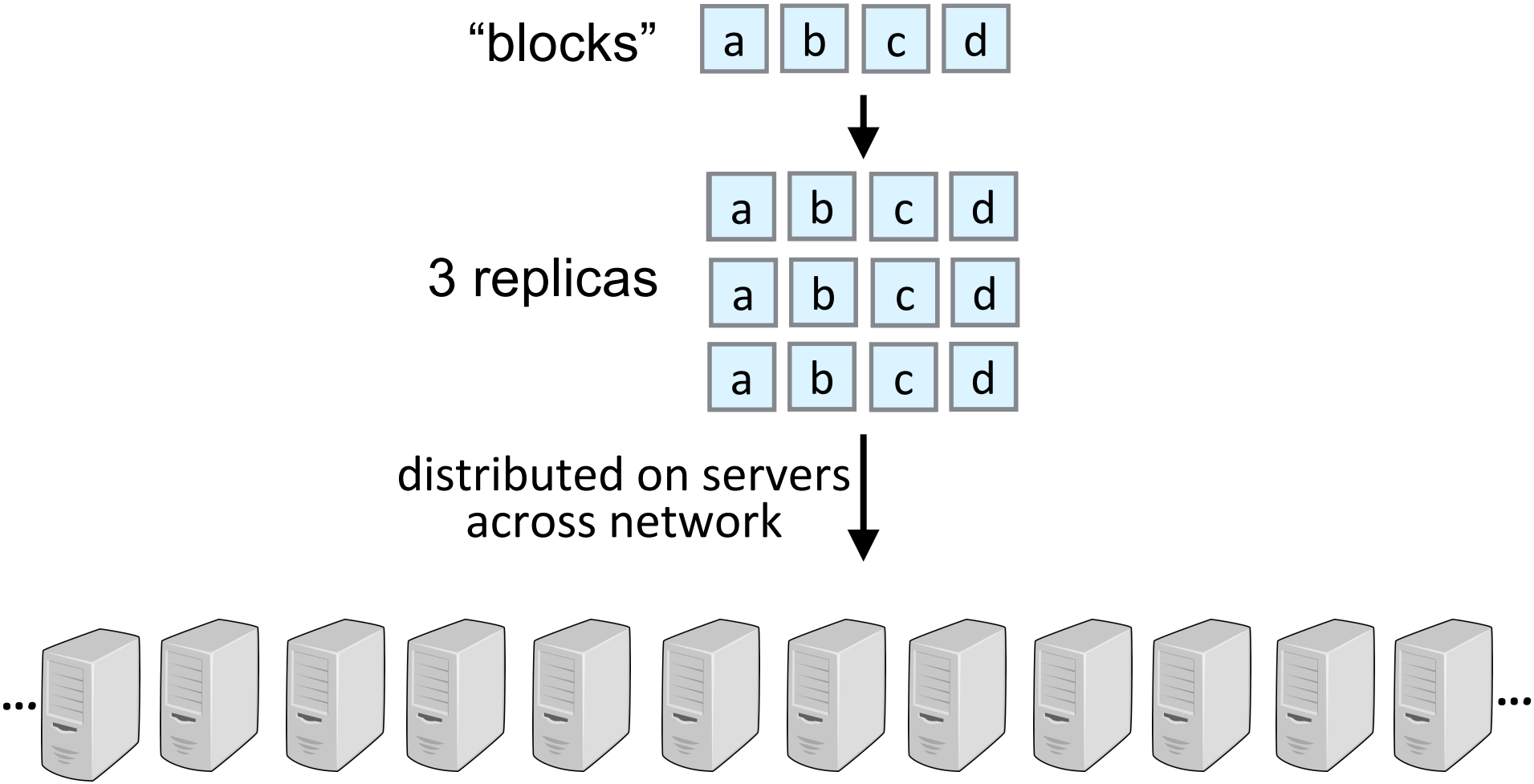
Data inaccessible

↓ Applications cannot wait,
Data cannot be lost

Data needs to be stored in a redundant fashion

Traditional approach: Replication

- Storing **multiple copies** of data: Typically 3x-replication

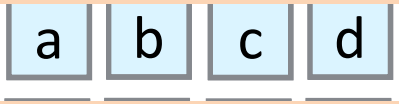


Traditional approach: Replication

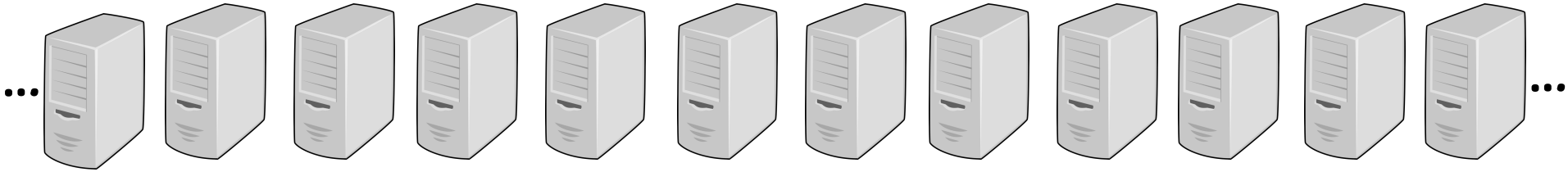
- Storing **multiple copies** of data: Typically 3x-replication

Too expensive for large-scale data

3 replicas

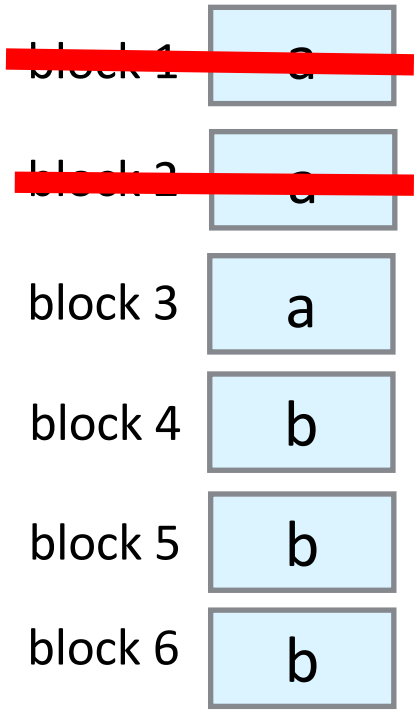


Better alternative: codes!



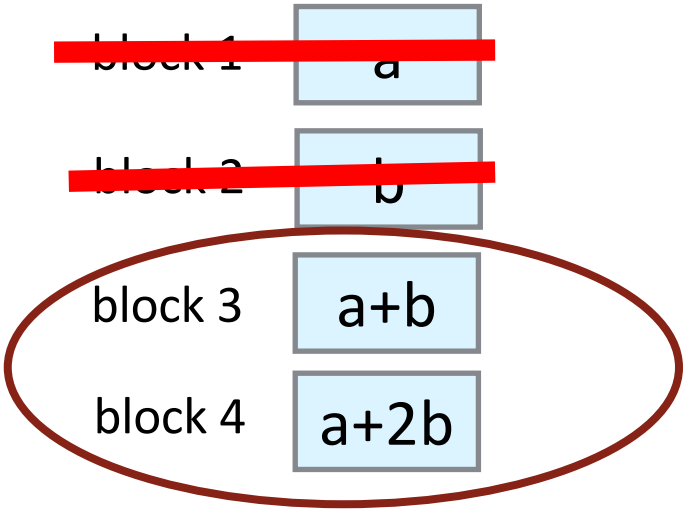
Two data blocks to be stored: a and b

Tolerate any 2 failures



3-replication

Storage overhead = 3x



"parity blocks"

Erasure code

Storage overhead = 2x

Two data blocks to be stored: a and b

Tolerate any 2 failures

Much less storage

for desired fault tolerance

block 5 b

block 6 b

3-replication

Storage overhead = 3x

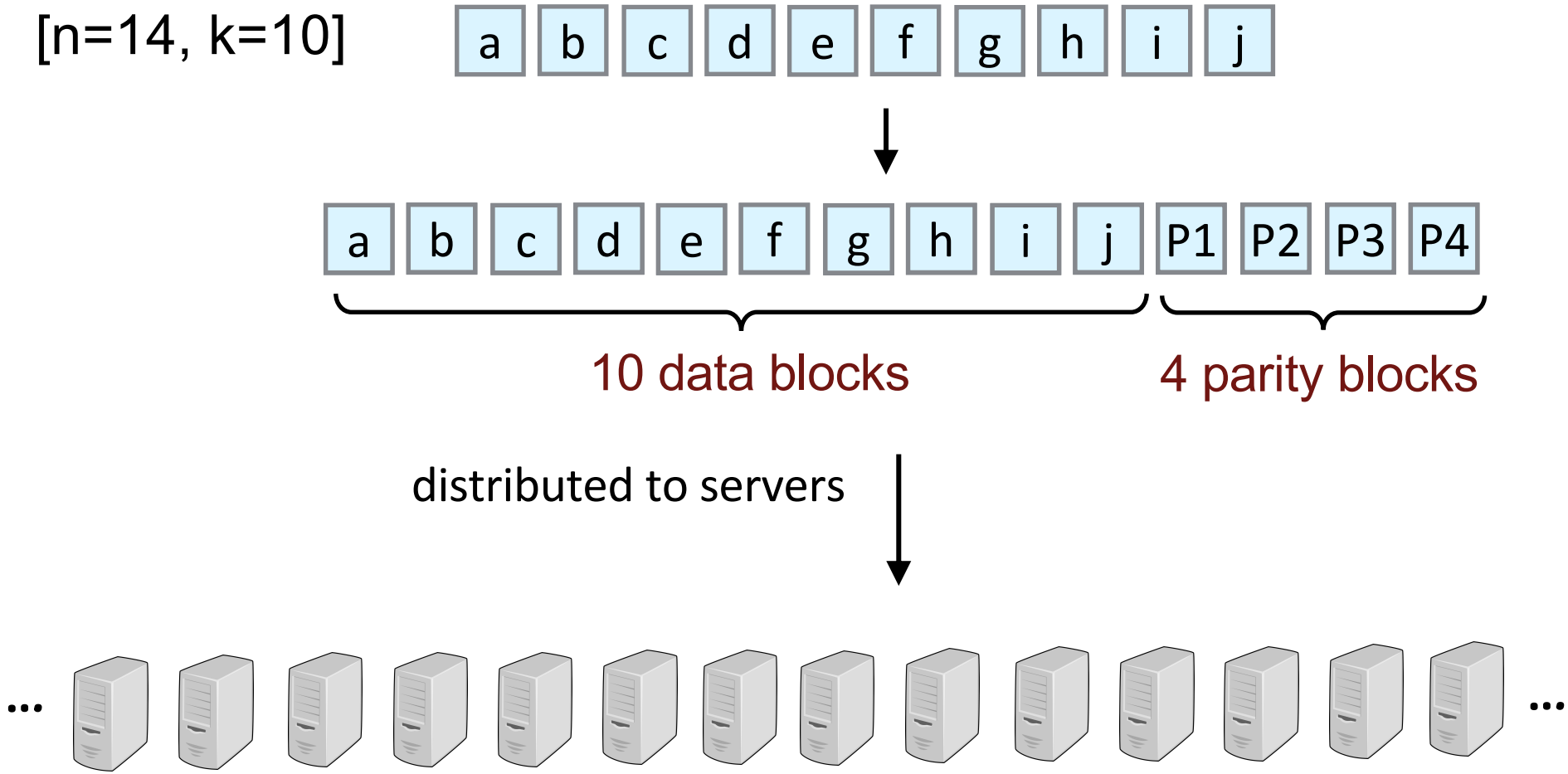
"parity blocks"

Erasure code

Storage overhead = 2x

Erasure codes: how are they used in distributed storage systems?

Example:
[n=14, k=10]



Almost all large-scale storage systems today employ erasure codes

Facebook, Google, Amazon, Microsoft...

"Considering trends in data growth & datacenter hardware, we foresee HDFS **erasure coding** being an **important feature in years to come**"

- Cloudera Engineering (September, 2016)

15-750: Graduate Algorithms

Algorithms for coding
(Error Correcting Codes)

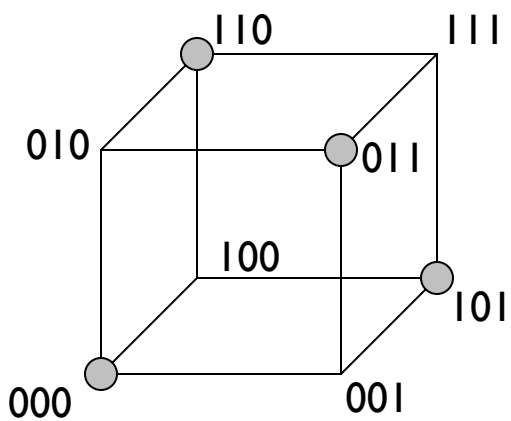
- Real-world application example
  - Cluster storage systems
- Continue with basic concepts
  - minimum distance, systematic property, linear codes, generator and parity-check matrices, Singleton bound
- Reed-Solomon codes

Simple Examples

Single parity check code: k=2, n=3

Message		Codeword
00	->	000
01	->	011
10	->	101
11	->	110

Consider codewords as vertices on a hypercube.



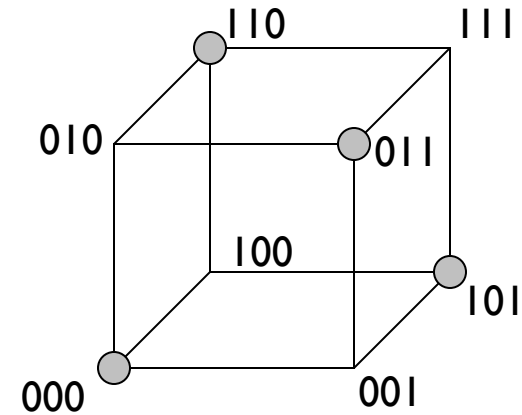
codeword

n = 3 = dimensionality

2^n = 8 = number of nodes

Simple Examples

Single parity check code: k=2, n=3



- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

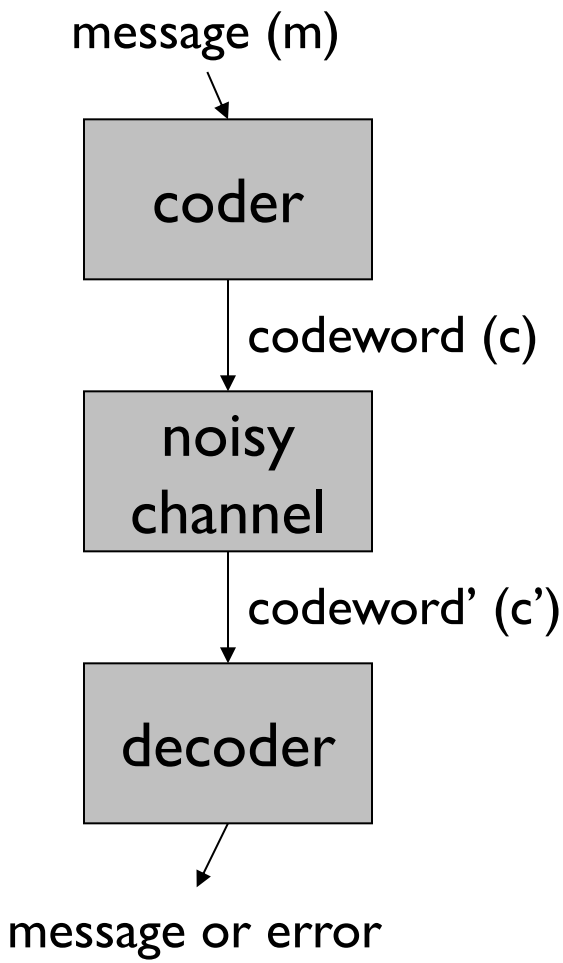
Erasure correction = 1, error detection = 1, error correction = 0

Cannot even correct single error. Why?

Codewords are too "close by"

Let's formalize this notion of distance..

Block Codes



Notion of distance between codewords:

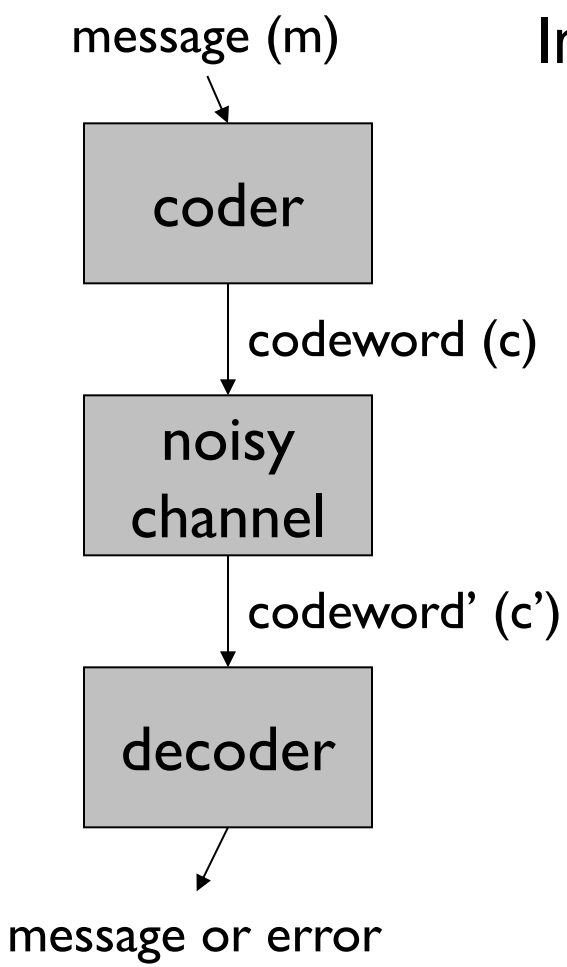
$\Delta(\mathbf{x},\mathbf{y})$ = number of positions s.t. $x_i \neq y_i$

minimum distance of a code

$\mathbf{d} = \min\{\Delta(x,y) : x,y \in C,\ x \neq y\}$

Code described as: $\mathbf{(n, k, d)_q}$

Block Codes



In general, symbols come from an "**alphabet**"

Notation:

Σ = alphabet

q = $|\Sigma|$ = alphabet size

Question:

What alphabet did we use so far?

$\mathbf{C} \subseteq \Sigma^n$ (codewords)

Binary Codes

Today we will mostly be considering $\Sigma = \{0,1\}$ and will sometimes use (n,k,d) as shorthand for $(n,k,d)_2$

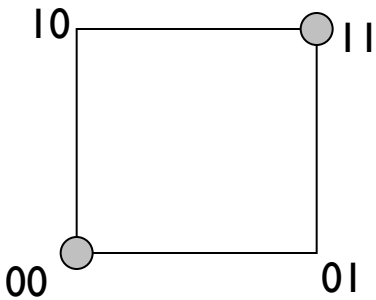
Systematic codes

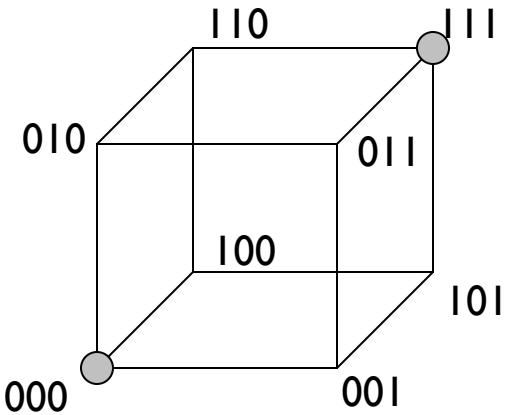
Definition: A **Systematic code** is one in which the message symbols appear in the codeword in uncoded form

message	codeword
000	**000**000
001	**001**011
010	**010**101
011	**011**110
100	**100**110
101	**101**101
110	**110**011
111	**111**000

Error Correcting One Bit Messages

How many bits do we need to correct a one bit error on a one bit message?





2 bits
0 -> 00, 1-> 11
(n=2,k=1,d=2)

3 bits
0 -> 000, 1-> 111
(n=3,k=1,d=3)

In general need d ≥ 3 to correct one error. Why? <board>

Role of Minimum Distance

Theorem:

A code C with minimum distance "d" can:

1. detect any (d-1) errors
2. recover any (d-1) erasures
3. correct any $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

Intuition: <board>

<u>Stated another way:</u>

For s-bit error detection $d \geq s + 1$

For s-bit error correction $d \geq 2s + 1$

To correct a erasures and b errors if $d \geq a + 2b + 1$

Desired Properties

We look for codes with the following properties:

1. Good rate: k/n should be high (low overhead)
2. Good distance: d should be large (good error correction)
3. Small block size k
4. Fast encoding and decoding
5. Others: want to handle bursty/random errors, local decodability, ...

Q:

If no structure in the code, how would one perform encoding?

Gigantic lookup table!

If no structure in the code, encoding is highly inefficient.

A common kind of structure added is **linearity**

Linear Codes

If $\sum$ is a field, then $\sum^n$ is a vector space

Definition: C is a linear code if it is a linear subspace of $\sum^n$ of dimension k.

This means that there is a set of k independent vectors $v_i \in \sum^n$ $(1 \leq i \leq k)$ that span the subspace.

i.e. every codeword can be written as:

$$c = a_1 v_1 + a_2 v_2 + \ldots + a_k v_k \quad \text{where } a_i \in \sum$$

"Basis (or spanning) Vectors"

Some Properties of Linear Codes

1. Linear combination of two codewords is a codeword.

2. Minimum distance (d) = weight of least weight (non-zero) codewords

(Weight of a vector refers to the Hamming weight of a vector, which is equal to the number of non-zero symbols in the vector)

$$d = \min_{\substack{c_i, c_j \in C \\ i \neq j}} |c_i - c_j|$$

$$= \min_{\substack{c \in C \\ c \neq 0}} |c|$$

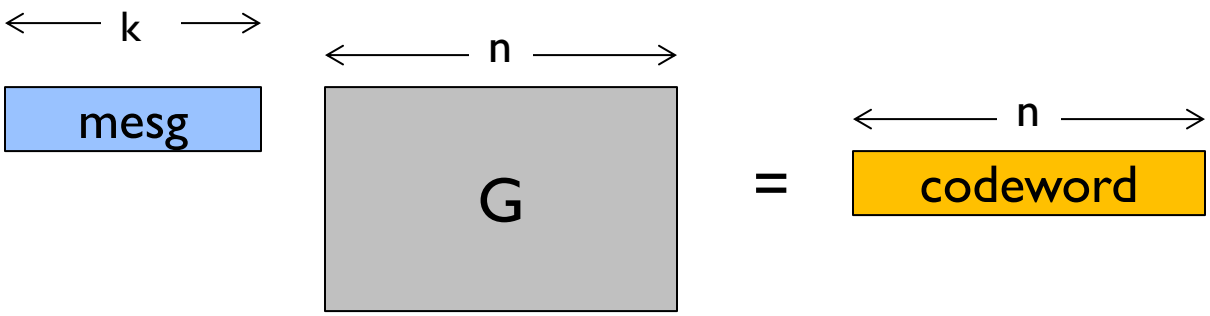
Generator and Parity Check Matrices

3. Every linear code has two matrices associated with it.

1. Generator Matrix:

A k x n matrix **G** such that: $C = \{ xG \mid x \in \Sigma^k \}$

Made from stacking the spanning vectors



To continue …