Distributed Storage Systems

Reliable Storage Continued:
Finite Fields & Linear Coding

Agenda

Reliable storage

Today's topics

- Basics of finite fields used in RAID and other storage systems
- Basics of coding for storage

Goals

This week’s Learning Goals

- Understand finite field arithmetics
- Understand basics of coding for reliable storage

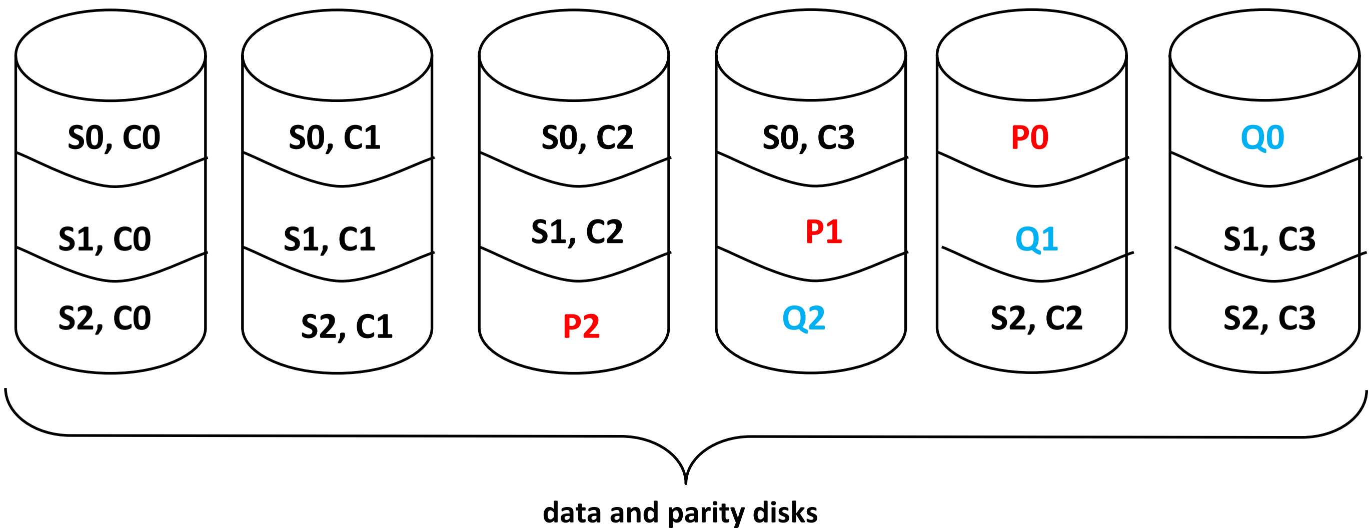
Class Structure

	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

Reliable Storage

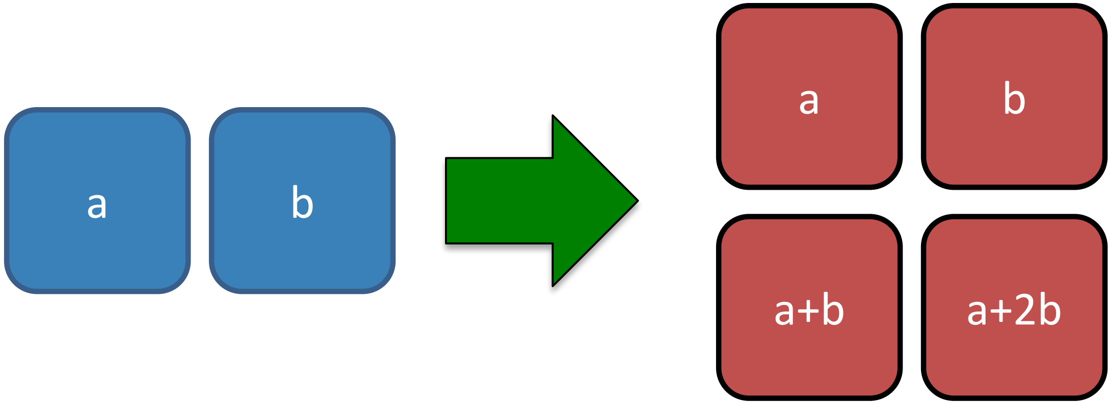
RAID6

- Level 5 with an extra parity
- Can tolerate two failures
- What are the odds of having two concurrent failures?



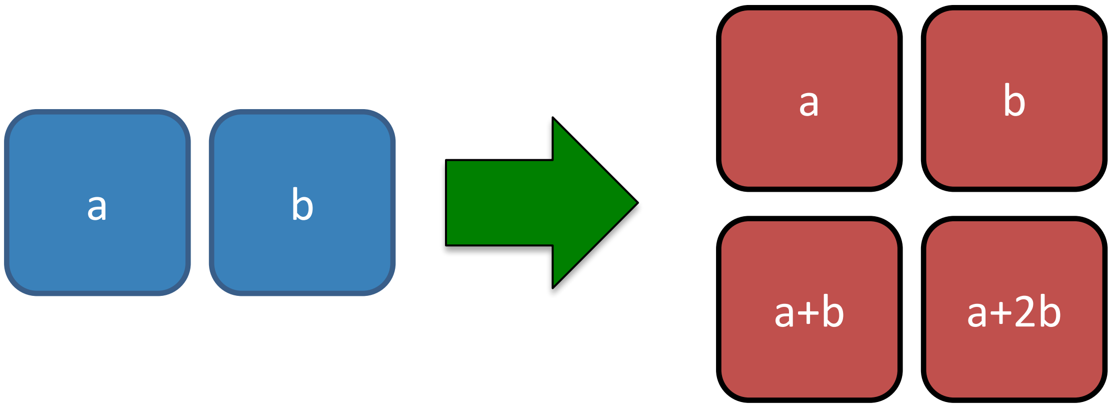
RAID6

- Level 5 with an extra parity
- Can tolerate two failures
- What are the odds of having two concurrent failures?



RAID6

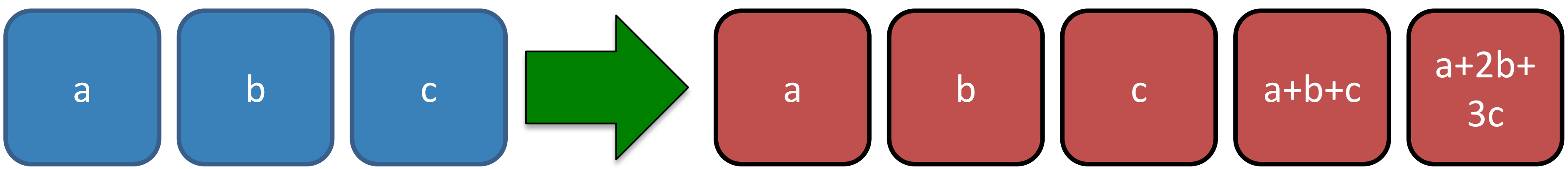
- Level 5 with an extra parity
- Can tolerate two failures
- What are the odds of having two concurrent failures?



How do we do a+2b?

RAID6 - More than 2 stripes?

- Level 5 with an extra parity
- Can tolerate two failures

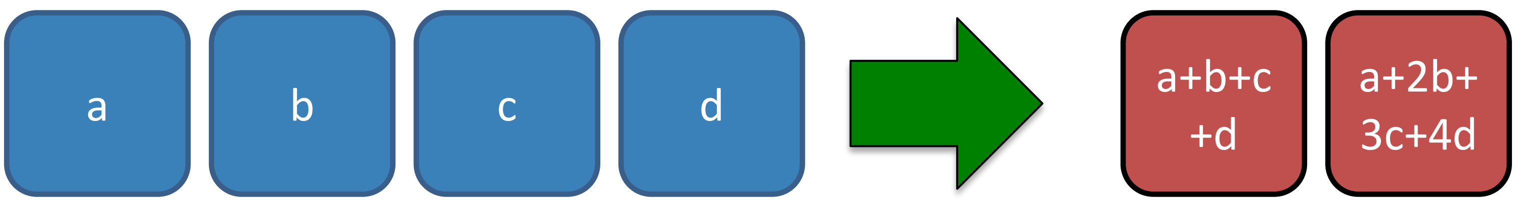


Reed Solomon codes provide the appropriate code construction

RAID6 - More than 2 stripes?

- Level 5 with an extra parity
- Can tolerate two failures

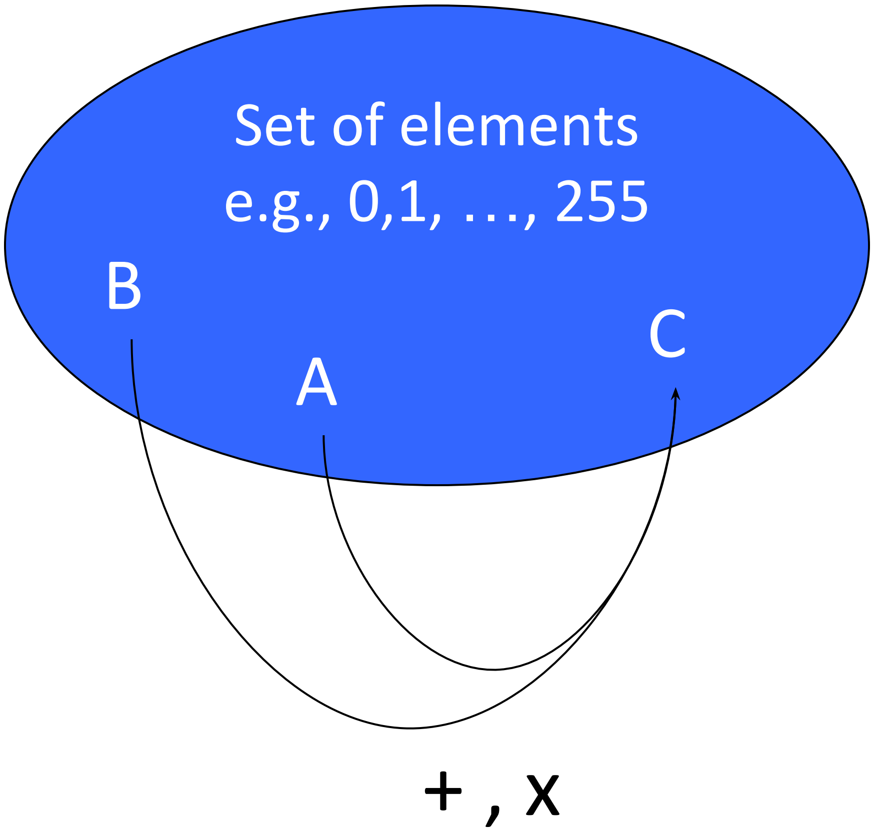
Only redundancy now :)



What about general
number of stripes?

What is a 4 in $GF(2^2)$?
How to do this?

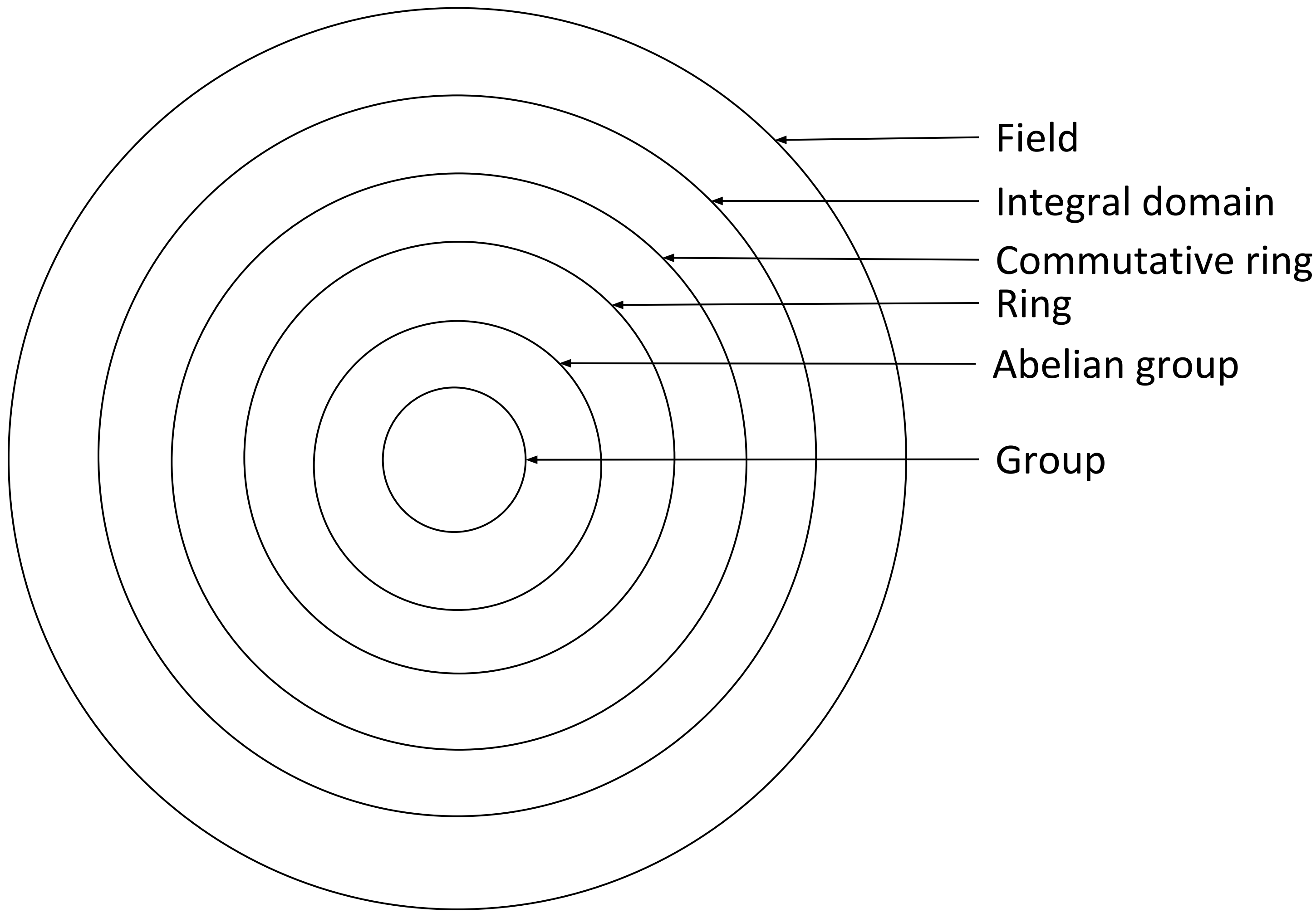
Finite Fields



Operations:
Addition, Multiplication

Property: closure

Groups, Rings, Fields



Groups, Rings, Fields

Groups

A group G, denoted {G, * } is a set of elements with a binary operation * that associates each ordered pair (a,b) of elements in G to an element (a*b) in G following

Axioms

- *Closure*: If a and b in G, then a * b is also in G
- *Associative:* a * (b * c) = (a * b) * c for all a,b,c in G
- *Identity:* Exists e in G, s.t. a*e = e*a = a for all a in G
- *Inverse:* for each a in G, exists a’ in G, s.t.

  a*a’ = a’*a = e

Finite group: finite number of elements

Groups, Rings, Fields

Abelian Group

Group that satisfies

- *Commutative*: If a and b in G, then a * b = b * a

Rings

A ring R, denoted by {R, +, *x*} is a set of elements with two binary operations: addition, multiplication. For all a,b,c in R the following axioms are satisfied

- R is **abelian group** with respect to the addition
- *Closure under multiplication:* ab in R
- *Associativity of multiplication:* a(bc) = (ab)c
- *Distributive laws:* a(b+c) = ab + ac

  (a + b)c = ac + bc

Groups, Rings, Fields

Commutative Ring

A **ring** that also satisfies

- *Commutativity of multiplication:* ab = ba in R

Integral Domain

R is a commutative **ring** that also satisfies

- *Multiplicative identity:* exists ***1***, s.t. a***1*** = ***1*** a = a
- *No zero divisors:* ab = 0 , implies either a = 0 or b = 0

Field

F is a field, { F*, +, x* } that satisfies

- F is an integral domain
- *Multiplicative inverse:* for each a in F, except 0, exists an element a^{-1}, s.t. $a(a^{-1}) = (a^{-1})a = 1$

Finite Fields GF(p)

Can write fields of the form GF(p^n), where p is prime

Addition and multiplication over GF(p) are mod p

Focus on p = 2

Example:

GF(2) addition: equivalent to XOR

multiplication: equivalent to AND

How to divide? Multiply by multiplicative inverse

Finding the multiplicative inverse

1.- Can look for a^{-1} such that $(a^{-1} . a) \equiv 1$

2.- Can use the extended Euclidean algorithm

Finite Fields - Applying GF(2) to NC

Example:

GF(2) addition: XOR

multiplication: AND

Given 2 data packets

P1: 01011001 P2: 10001001

calculate the content of the coded packet P1+P2.

What are the coefficients?

P1 + P2 = 01011001 (XOR bit by bit)

10001001

11010000

Modular Arithmetic

Modulus

If a is an integer, n > 0 integer, we define a mod n to be the remainder when a is divided by n

- The integer n is called the *modulus*
- For any integer a, we can write

$$\mathbf{a = qn + r}, \text{ with } 0 \le r < n, \text{ and } q = \lfloor a/n \rfloor$$

- E.g., 11 mod 7 = 4 , -11 mod 7 = 3

Congruent modulo n

If (a mod n) = (b mod n), and it's expressed

$$a \equiv b \pmod{n}$$

- E.g., $20 \equiv 6 \pmod{7}$

Modular Arithmetic

Properties of congruencies

- $a \equiv b \pmod{n}$ if $n|(a-b)$
- $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$
- $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$

Modular arithmetic operations

- $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
- $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
- $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Rules of ordinary arithmetic involving addition, subtraction, multiplication carry over

Modular Arithmetic

Properties of modular arithmetic

Define Z_n = { 0, 1, …, n-1 } as the set of **residues** or **residue classes** mod n.

Each element of Z_n is a residue class and can define it as

$$[\ j\ ] = \{a : a \text{ is integer}, a \equiv j \pmod{n}\}$$

- Reducing k mod n: finding smallest non-negative integer a, such that $k \equiv a \pmod{n}$

Modular Arithmetic

Properties of modular arithmetic

- (w + x) mod n = (x + w) mod n
- (w *x* y) mod n = (y *x* w) mod n
- ((w + x) + y) mod n = (w + (x + y)) mod n
- ((w *x* j) *x* y) mod n = (w *x* (j *x* y)) mod n
- (w *x* (y + j)) mod n = ((w *x* y) + (w *x* j)) mod n
- (0 + w) mod n = w mod n
- (1 x w) mod n = w mod n

Modular Arithmetic

Properties of modular arithmetic

- If $(a + b) \equiv (a + c) \pmod{n}$, then $b \equiv c \pmod{n}$
- If $(a \times b) \equiv (a \times c) \pmod{n}$, then $b \equiv c \pmod{n}$ if a is relatively prime to n, i.e., $\gcd(a,n) = 1$

Why is the last property important?

What happens when “p” is not a prime?

Will modular arithmetic still work?

Example 1:

If gcd(a,n) ≠ 1, the last equation does not hold

e.g. $6 \times 3 = 18 = 2 \bmod 8$

and $6 \times 7 = 42 = 2 \bmod 8$ but

$3 \bmod 8 \neq 7 \bmod 8$

What happens when "p" is not a prime?

Will modular arithmetic still work?

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

x	0	1	2	3
0	0	0	0	0
1	0	**1**	**2**	**3**
2	0	**2**	**0**	**2**
3	0	**3**	**2**	**1**

What about GF(2^n)?

Since 2^n is not a prime, operations are defined in a different way => **polynomial arithmetic**

Ordinary polynomial arithmetic:

A polynomial of degree n

$$F(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 x^0 = \Sigma a_i x^i$$

a_i are the coefficients, chosen from a set

Operations:

Addition $\quad f(x) + g(x) = \Sigma (a_i + b_i) x^i$

Multiplication $\; f(x) \, x \, g(x) = \Sigma C_i \; x^i$

with $c_k = a_0 b_k + a_1 b_{k-1} + \ldots + a_k b_0$

What about GF(2^n)?

Polynomial arithmetic in GF(2^n):

- Arithmetic follows rules of polynomial arithmetic
- Arithmetic of coefficients is performed modulo *2*
  - i.e.,using GF(2) addition/multiplication for coefficients of the same order
  - e.g., (a_i + b_i) mod 2
- If multiplication results in a polynomial greater than n-1, then the polynomial is reduced modulo an irreducible polynomial p(x)
  - Think of it as a mod p(x) operation: divide by p(x), keep the remainder

Example GF(2^2)

Irreducible polynomial $p(x) = x^2 + x + 1$ $(111)_b$

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

How about 2 +3?

$2 = (10)_b$ and $3 = (11)_b$

As polynomials:

$2 \equiv x$ and $3 \equiv x + 1$

Thus, 2 + 3 becomes

$x + (x + 1) = 1$

Example GF(2^2)

Irreducible polynomial $p(x) = x^2 + x + 1$ $(111)_b$

x	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Equal number of each element: RLNC's properties are maintained when recoding

Multiplicative inverses; easy to spot in table

Can we compute without generating table?

How about 2 x 3?

$x\ (x + 1) = (x^2 + x) \bmod p(x)$

$$
\begin{array}{r|l}
 & \quad 1 \\
\hline
x^2 + x + 1 & x^2 + x \\
 & x^2 + x + 1 \\
\cline{2-2}
 & \quad 1
\end{array}
$$

How to implement multiplication $GF(2^n)$?

A.- Product (shifts+ XORs)

1) Pick one as multiplier (M) and another as multiplicand (m)
2) For each “1” in “M”, left shift the “m” by the position of the “1”
3) XOR shifted versions

B.- Modulo irreducible polynomial (long division)

4) Initialize: F(0) = Result of part A
5) Take irreducible polynomial p(x) left shift until first “1” of polynomial and of value F match
6) F(i+1) = (shifted p(x)) XOR (F(i))
7) Stop if first “1” of F(i+1) occurs in the (n-1)-th bit

How to implement multiplication?

M = $(00010001)_b$ and m = $(10100111)_b$

	101001110000	(m shifted 4 times)
(XOR)	10100111	(m shifted 0 times)
	101011010111	

B.- Modulo irreducible polynomial

$p(x) = x^8 + x^4 + x^3 + x + 1$ $(100011011)_b$

	101011010111	mod p(x)
(XOR)	**100011011**000	
	001000001111	
(XOR)	00**10001101**10	
	000000**111001**	$\rightarrow$ $\mathbf{(00111001)_b}$

Generating Coded Fragments/Stripes

- Generating a linear coded fragment/stripe (C)

$$C_i = \sum a_{ij} P_j$$

Coded stripes/fragments

Coefficients

Original stripes/fragments

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

$M \times M'$

How you pick the coefficients determines properties and performance

Generating Coded Packets

$$
\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{pmatrix}
=
\begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\
a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66}
\end{pmatrix}
\left(\begin{array}{c|c|c|c|c}
p_{11} & p_{12} & p_{13} & p_{14} & p_{15} \\
p_{21} & p_{22} & p_{23} & p_{24} & p_{25} \\
p_{31} & p_{32} & p_{33} & p_{34} & p_{35} \\
p_{41} & p_{42} & p_{43} & p_{44} & p_{45} \\
p_{51} & p_{52} & p_{53} & p_{54} & p_{55} \\
p_{61} & p_{62} & p_{63} & p_{64} & p_{65}
\end{array}\right)
$$

RAID6

- Some data coded, some is not
- Called: 'Systematic' Code

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

RAID6

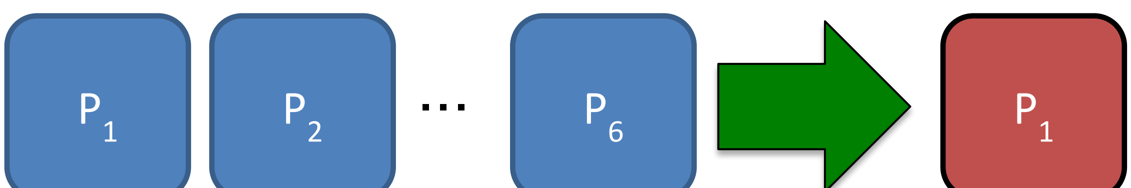
- Some data coded, some is not
- Called: 'Systematic' Code

$$
\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{pmatrix}
=
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 3 & 4 & 5 & 6
\end{pmatrix}
\begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}
$$

RAID6

- Some data coded, some is not
- Called: ‘Systematic’ Code

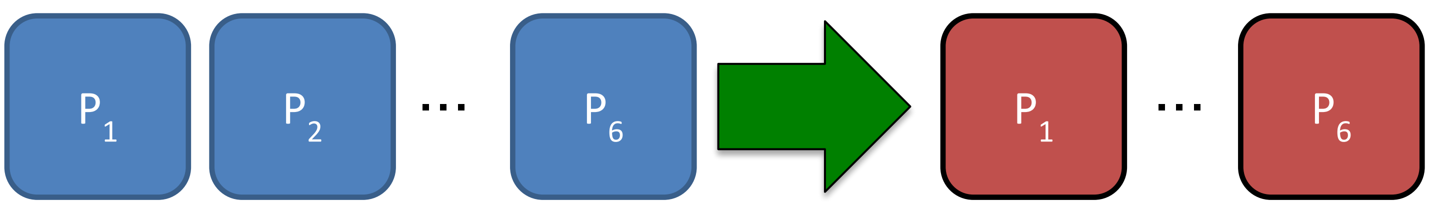
$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$



RAID6

- Some data coded, some is not
- Called: ‘Systematic’ Code

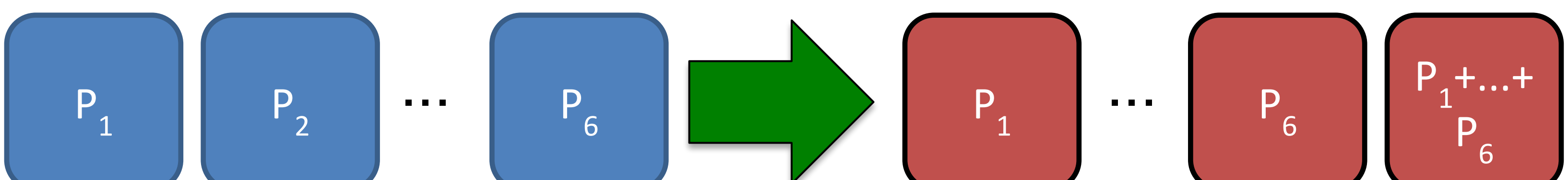
$$
\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{pmatrix}
=
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 3 & 4 & 5 & 6
\end{pmatrix}
\begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}
$$



RAID6

- Some data coded, some is not
- Called: ‘Systematic’ Code

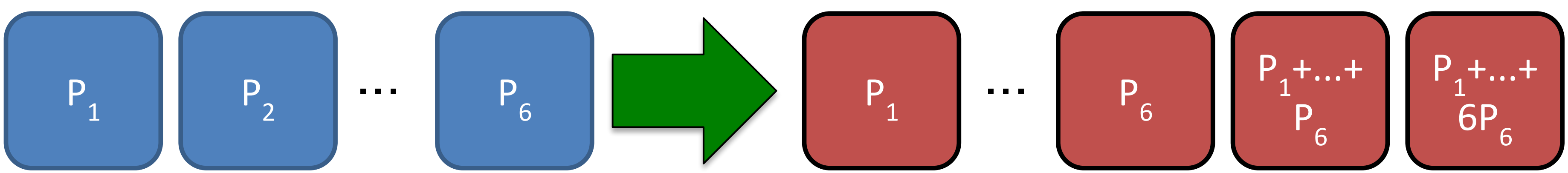
$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$



RAID6

- Some data coded, some is not
- Called: ‘Systematic’ Code

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$



RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

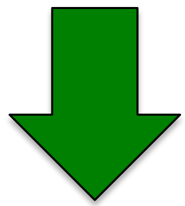
RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$

RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$



$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 3 & 4 & 5 & 6 \end{pmatrix}$$

RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

$$\Big\Downarrow$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 5 & 6 \end{pmatrix}$$

RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 5 & 6 \end{pmatrix}$$

Multiply by 5
and
Add

RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

Multiply by 3^{-1}

RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Add

RAID6 - Decode

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_7 \\ C_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Decoded

What are good constructions of the (coefficient) matrix?

What are good constructions of the (coefficient) matrix?

Depends on performance objective:

- Storage reduction
- Reduced I/O operations
- ...

What are good constructions of the (coefficient) matrix?

Depends on performance objective:

- **Storage reduction:** MDS codes, e.g., Reed Solomon
- Reduced I/O operations
- ...

MDS Codes

- A maximum distance separable code, denoted by **MDS(n, k)**, has the property that any **k** (< n) out of n nodes can be used to reconstruct original native blocks
  – i.e., at most n-k disk failures can be tolerated
- Example:
  – RAID-5 is an MDS(n, n-1) code as it can tolerate 1 disk failure
  – RAID-6 is an MDS(n,n-2) code as it can tolerate 2 disk failures

Slide adapted from Yuchong Hu, CUHK

Reed Solomon - Vandermonde

- Not systematic

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ \vdots \\ \\ C_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ \alpha_1^{1} & \alpha_2^{1} & \alpha_3^{1} & \alpha_4^{1} & \dots & \alpha_k^{1} \\ \alpha_1^{2} & \alpha_2^{2} & \alpha_3^{2} & \alpha_4^{2} & \dots & \alpha_k^{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \\ \alpha_1^{(n-1)} & \alpha_2^{(n-1)} & \alpha_3^{(n-1)} & \alpha_4^{(n-1)} & \dots & \alpha_k^{(n-1)} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ \\ P_k \end{pmatrix}$$

Reed Solomon - Vandermonde

- Not systematic

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ \alpha_1^{\,1} & \alpha_2^{\,1} & \alpha_3^{\,1} & \alpha_4^{\,1} & \dots & \alpha_k^{\,1} \\ \alpha_1^{\,2} & \alpha_2^{\,2} & \alpha_3^{\,2} & \alpha_4^{\,2} & \dots & \alpha_k^{\,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{\,(n-1)} & \alpha_2^{\,(n-1)} & \alpha_3^{\,(n-1)} & \alpha_4^{\,(n-1)} & \dots & \alpha_k^{\,(n-1)} \end{pmatrix}$$

Reed Solomon - Vandermonde

- Not systematic

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & 4 & \cdots & k \\ 1^2 & 2^2 & 3^2 & 4^2 & \cdots & k^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \\ 1^{(n-1)} & 2^{(n-1)} & 3^{(n-1)} & 4^{(n-1)} & \cdots & k^{(n-1)} \end{pmatrix}$$

There is a limit on the value of n $\rightarrow$ n cannot be larger than (size of field -1)
For GF(2^m), $n < 2^m-1$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 2^2 & 3^2 & 4^2 \\ 1 & 2^3 & 3^3 & 4^3 \\ 1 & 2^4 & 3^4 & 4^4 \\ 1 & 2^5 & 3^5 & 4^5 \end{pmatrix}$$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4
- Need GF(2^3) (at least) - polynomial: x^3+x^2+1

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 2^2 & 3^2 & 4^2 \\ 1 & 2^3 & 3^3 & 4^3 \\ 1 & 2^4 & 3^4 & 4^4 \\ 1 & 2^5 & 3^5 & 4^5 \end{pmatrix}$$

Mapping to polynomial

$0 \rightarrow 0$
$1 \rightarrow 1$
$2 \rightarrow x$
$3 \rightarrow x + 1$
$4 \rightarrow x^2$
$5 \rightarrow x^2+1$
$6 \rightarrow x^2+x$
$7 \rightarrow x^2+x+1$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4
- Need GF(2^3) (at least) - primitive polynomial: x^3+x^2+1
- What about the exponents?

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 2^2 & 3^2 & 4^2 \\ 1 & 2^3 & 3^3 & 4^3 \\ 1 & 2^4 & 3^4 & 4^4 \\ 1 & 2^5 & 3^5 & 4^5 \end{pmatrix}$$

Mapping to polynomial

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow x$

$3 \rightarrow x+1$

$4 \rightarrow x^2$

$5 \rightarrow x^2+1$

$6 \rightarrow x^2+x$

$7 \rightarrow x^2+x+1$

Mapping to polynomial

$2^2 \rightarrow x^2 \rightarrow \mathbf{4}$

$2^3 \rightarrow x^3 \rightarrow R(x^3/x^3+x^2+1) \rightarrow x^2+1 \rightarrow \mathbf{5}$

$2^4 \rightarrow x^4 \rightarrow x(x^2+1) \rightarrow x^2+x+1 \rightarrow \mathbf{7}$

$2^5 \rightarrow x.x^4 \rightarrow x(x^2+x+1) \rightarrow x+1 \rightarrow \mathbf{3}$

$2^6 \rightarrow x.x^5 \rightarrow x(x+1) \rightarrow x^2+x \rightarrow \mathbf{6}$

$2^7 \rightarrow x.x^6 \rightarrow x(x^2+x) \rightarrow x^3+x^2 \rightarrow \mathbf{1}$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4
- Need GF(2^3) (at least) - primitive polynomial: x^3+x^2+1
- What about the exponents?

$$
H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 3^2 & 4^2 \\ 1 & 5 & 3^2 & 4^2 \\ 1 & 7 & 3^2 & 4^2 \\ 1 & 3 & 3^5 & 4^5 \end{pmatrix}
$$

Mapping to polynomial

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow x$

$3 \rightarrow x+1$

$4 \rightarrow x^2$

$5 \rightarrow x^2+1$

$6 \rightarrow x^2+x$

$7 \rightarrow x^2+x+1$

Mapping to polynomial

$2^2 \rightarrow x^2 \rightarrow \mathbf{4}$

$2^3 \rightarrow x^3 \rightarrow R(x^3/x^3+x^2+1) \rightarrow x^2+1 \rightarrow \mathbf{5}$

$2^4 \rightarrow x^4 \rightarrow x(x^2+1) \rightarrow x^2+x+1 \rightarrow \mathbf{7}$

$2^5 \rightarrow x.x^4 \rightarrow x(x^2+x+1) \rightarrow x+1 \rightarrow \mathbf{3}$

$2^6 \rightarrow x.x^5 \rightarrow x(x+1) \rightarrow x^2+x \rightarrow \mathbf{6}$

$2^7 \rightarrow x.x^6 \rightarrow x(x^2+x) \rightarrow x^3+x^2 \rightarrow \mathbf{1}$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4
- Need GF(2^3) (at least) - primitive polynomial: x^3+x^2+1
- What about the exponents?

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 3^2 & 4^2 \\ 1 & 5 & 3^3 & 4^3 \\ 1 & 7 & 3^4 & 4^4 \\ 1 & 3 & 3^5 & 4^5 \end{pmatrix}$$

Mapping to polynomial

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow x$

$3 \rightarrow x+1$

$4 \rightarrow x^2$

$5 \rightarrow x^2+1$

$6 \rightarrow x^2+x$

$7 \rightarrow x^2+x+1$

Mapping to polynomial

$3^2 \rightarrow (x+1)^2 \rightarrow x^2+1 \rightarrow \mathbf{5}$

$3^3 \rightarrow (x^2+1)(x+1) \rightarrow x \rightarrow \mathbf{2}$

$3^4 \rightarrow x(x+1) \rightarrow (x^2+x) \rightarrow \mathbf{6}$

$3^5 \rightarrow (x^2+x)(x+1) \rightarrow x^2+x+1 \rightarrow \mathbf{7}$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4
- Need GF(2^3) (at least) - primitive polynomial: x^3+x^2+1
- What about the exponents?

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 5 & 4^2 \\ 1 & 5 & 2 & 4^3 \\ 1 & 7 & 6 & 4^4 \\ 1 & 3 & 7 & 4^5 \end{pmatrix}$$

Mapping to polynomial

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow x$

$3 \rightarrow x+1$

$4 \rightarrow x^2$

$5 \rightarrow x^2+1$

$6 \rightarrow x^2+x$

$7 \rightarrow x^2+x+1$

Mapping to polynomial

$3^2 \rightarrow (x+1)^2 \rightarrow x^2+1 \rightarrow \mathbf{5}$

$3^3 \rightarrow (x^2+1)(x+1) \rightarrow x \rightarrow \mathbf{2}$

$3^4 \rightarrow x(x+1) \rightarrow (x^2+x) \rightarrow \mathbf{6}$

$3^5 \rightarrow (x^2+x)(x+1) \rightarrow x^2+x+1 \rightarrow \mathbf{7}$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4
- Need GF(2^3) (at least) - primitive polynomial: x^3+x^2+1
- What about the exponents?

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 5 & 4^2 \\ 1 & 5 & 2 & 4^3 \\ 1 & 7 & 6 & 4^4 \\ 1 & 3 & 7 & 4^5 \end{pmatrix}$$

Mapping to polynomial

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow x$

$3 \rightarrow x+1$

$4 \rightarrow x^2$

$5 \rightarrow x^2+1$

$6 \rightarrow x^2+x$

$7 \rightarrow x^2+x+1$

Mapping to polynomial

$4^2 \rightarrow x^4 \rightarrow x^2+x+1 \qquad \rightarrow \mathbf{7}$

$4^3 \rightarrow x^2(x^2+x+1) \rightarrow x^2+x \qquad \rightarrow \mathbf{6}$

$4^4 \rightarrow x^2(x^2+x) \rightarrow x \qquad \rightarrow \mathbf{2}$

$4^5 \rightarrow x^3 \rightarrow x^2+1 \qquad \rightarrow \mathbf{5}$

Reed Solomon - Vandermonde

- Not systematic n = 6, k = 4
- Need GF(2^3) (at least) - primitive polynomial: x^3+x^2+1
- What about the exponents?

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 5 & 7 \\ 1 & 5 & 2 & 6 \\ 1 & 7 & 6 & 2 \\ 1 & 3 & 7 & 5 \end{pmatrix}$$

Mapping to polynomial

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow x$

$3 \rightarrow x+1$

$4 \rightarrow x^2$

$5 \rightarrow x^2+1$

$6 \rightarrow x^2+x$

$7 \rightarrow x^2+x+1$

Mapping to polynomial

$4^2 \rightarrow x^4 \rightarrow x^2+x+1 \rightarrow \mathbf{7}$

$4^3 \rightarrow x^2(x^2+x+1) \rightarrow x^2+x \rightarrow \mathbf{6}$

$4^4 \rightarrow x^2(x^2+x) \rightarrow x \rightarrow \mathbf{2}$

$4^5 \rightarrow x^3 \rightarrow x^2+1 \rightarrow \mathbf{5}$

Reed Solomon - Vandermonde

- Systematic - different procedures starting with H
- Simple:
  - Transpose H
  - Perform Gaussian Elimination on $H^T \longrightarrow H_{sys}{}^T = [\boldsymbol{I} \mid \boldsymbol{\mathcal{R}}]$
  - Transpose again to reach H_{sys}

f = n - k

Reed Solomon - Cauchy

- Systematic by design

$$
H = \begin{pmatrix}
1 & 0 & 0 & 0 & \dots & 0 \\
0 & 1 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \dots & 1 \\
(x_1+y_1)^{-1} & (x_2+y_1)^{-1} & (x_3+y_1)^{-1} & (x_4+y_1)^{-1} & \dots & (x_k+y_1)^{-1} \\
(x_1+y_2)^{-1} & (x_2+y_2)^{-1} & (x_3+y_2)^{-1} & (x_4+y_2)^{-1} & \dots & (x_k+y_2)^{-1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
(x_1+y_{n-k})^{-1} & (x_2+y_{n-k})^{-1} & (x_3+y_{n-k})^{-1} & (x_4+y_{n-k})^{-1} & \dots & (x_k+y_{n-k})^{-1}
\end{pmatrix}
$$

$X = \{x_1, x_2, x_3, \dots, x_k\}$ and $Y = \{y_1, y_2, y_3, \dots, y_{n-k}\}$, where $x_i \neq y_j \ \forall\ i,j$

Reed Solomon - Cauchy

- Systematic by design

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ (x_1+y_1)^{-1} & (x_2+y_1)^{-1} & (x_3+y_1)^{-1} & (x_4+y_1)^{-1} & \cdots & (x_k+y_1)^{-1} \\ (x_1+y_2)^{-1} & (x_2+y_2)^{-1} & (x_3+y_2)^{-1} & (x_4+y_2)^{-1} & \cdots & (x_k+y_2)^{-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (x_1+y_{n-k})^{-1} & (x_2+y_{n-k})^{-1} & (x_3+y_{n-k})^{-1} & (x_4+y_{n-k})^{-1} & \cdots & (x_k+y_{n-k})^{-1} \end{pmatrix}$$

This addition and inverse is in GF(q)

🙂

Example: $k = 2$, $n = 7 \rightarrow X = \{1, 2\}$ and $Y = \{0, 3, 4, 5, 6\}$

Reed Solomon - Cauchy

- n = 6, k = 4 and X = {1, 2, 3, 4} and Y = {0, 5}

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2^{-1} & 3^{-1} & 4^{-1} \\ 4^{-1} & 7^{-1} & 6^{-1} & 1 \end{pmatrix}$$

Find inverses
How?

Reed Solomon - Cauchy

- n = 6, k = 4 and X = {1, 2, 3, 4} and Y = {0, 5}

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2^{-1} & 3^{-1} & 4^{-1} \\ 4^{-1} & 7^{-1} & 6^{-1} & 1 \end{pmatrix}$$

Tables or Computation

$2.2^{-1} = 1$?

$2.6 = x\,(x^2 + x) \rightarrow 1$

Thus, $2^{-1} = 6$

Reed Solomon - Cauchy

- n = 6, k = 4 but with **X = {1, 3, 4, 5} and Y = {0, 2}**

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 3^{-1} & 4^{-1} & 5^{-1} \\ 3^{-1} & 1 & 6^{-1} & 7^{-1} \end{pmatrix}$$

Large number of Cauchy Alternatives - not all built equally

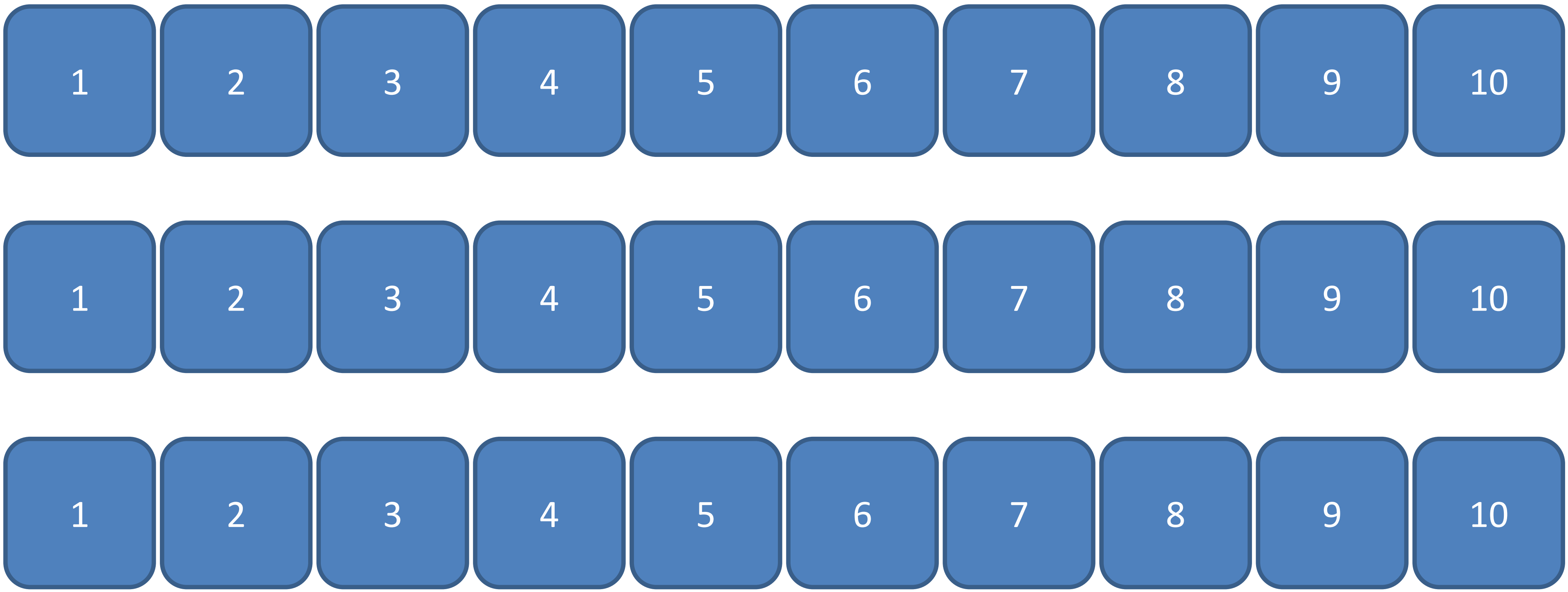
Distributed Storage: Reliability Beyond a Single Server

Distributed storage systems

- Numerous disk failures per day.
- Failures are the norm rather than the exception
- Must introduce **redundancy for reliability**
- Replication or erasure coding?
  - Current question in Hadoop HDFS, OpenStack Swift, Ceph, …

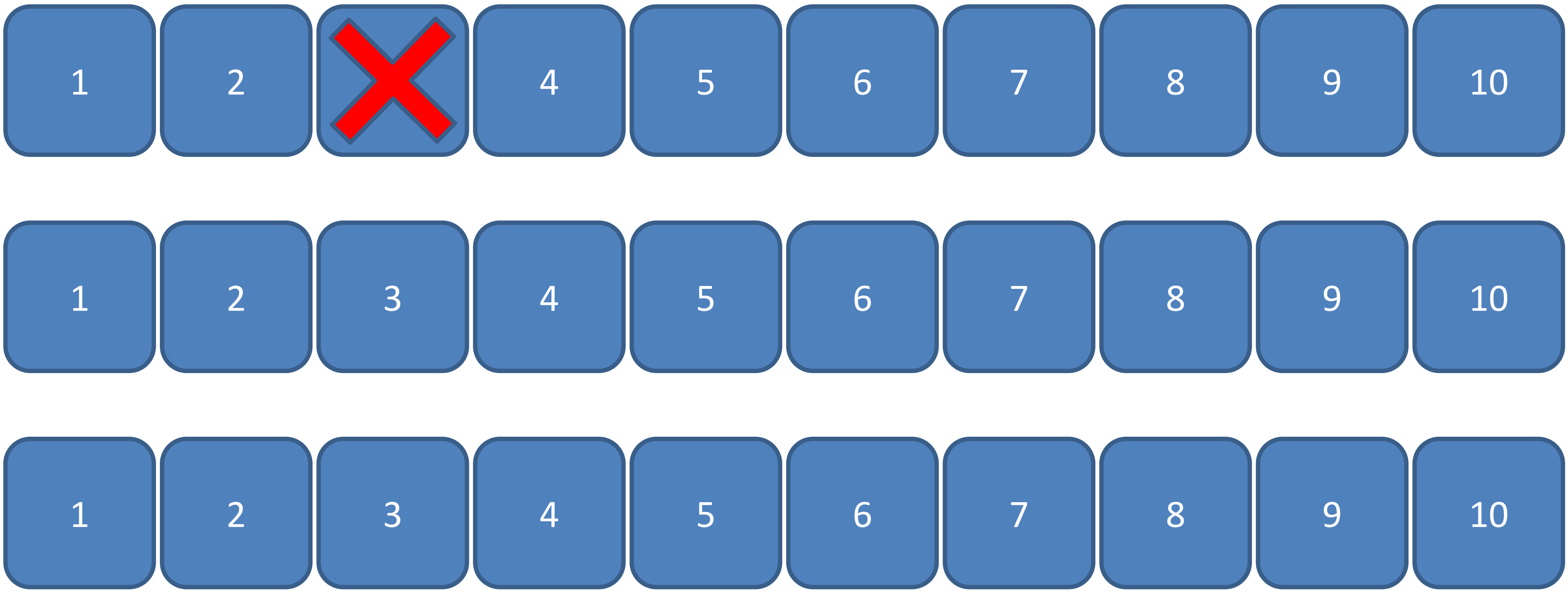
Slide adapted from A. Dimakis

State of the Art



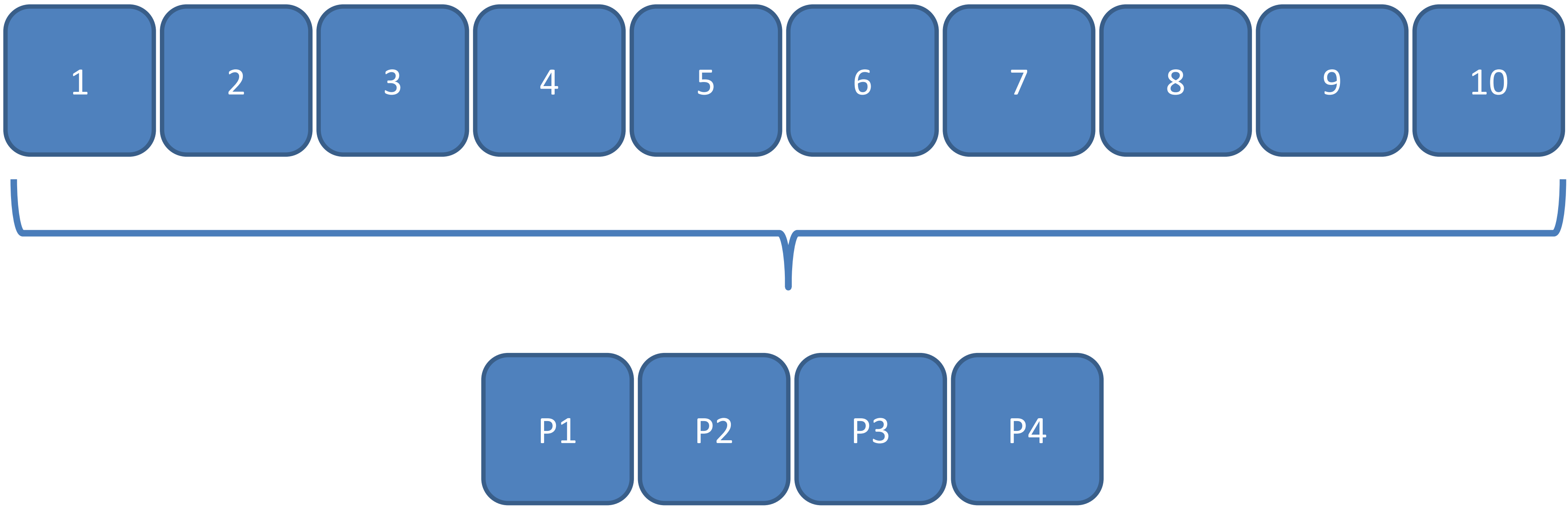
Overhead: 200%

State of the Art



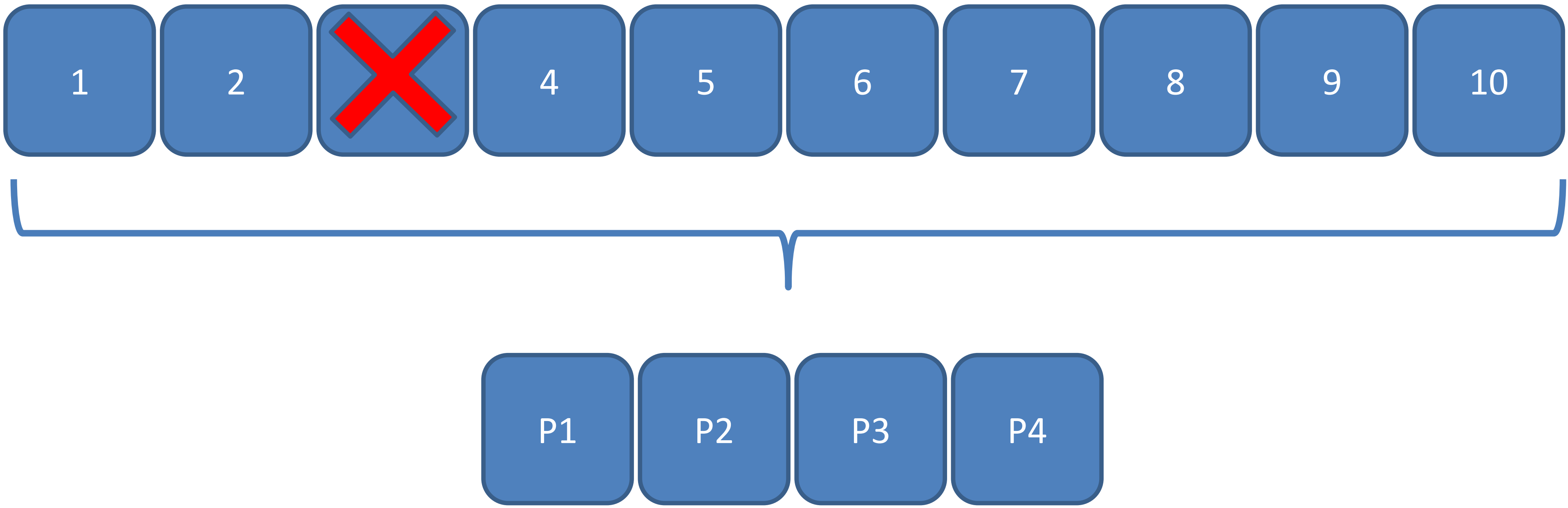
One failure results in one traffic unit

10:4 Code (Facebook)



Overhead: 40%

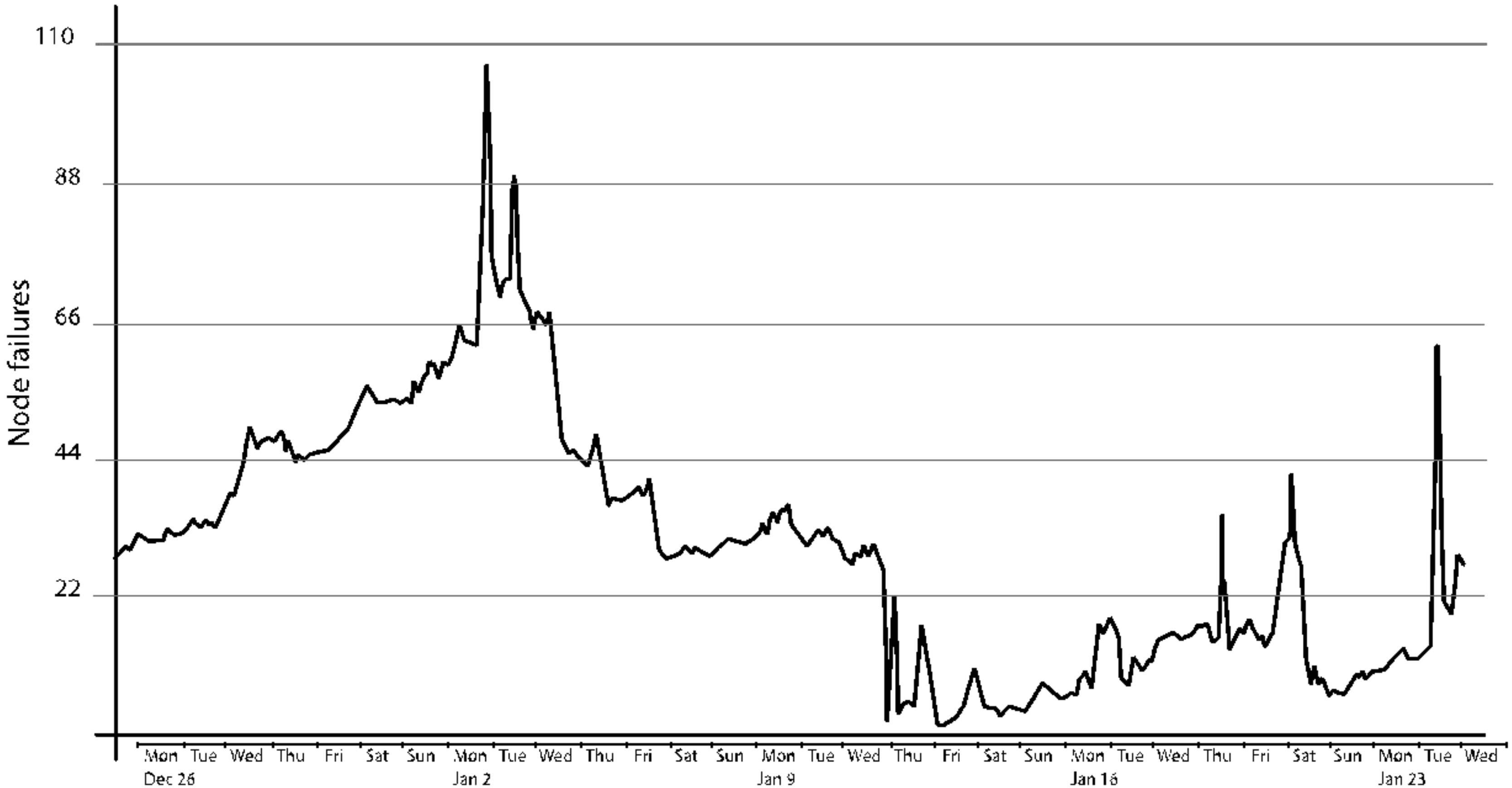
10:4 Code (Facebook)



Overhead: 40%

One failure results in 10 traffic units

Is repair frequent?



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