Recap: Singleton bound

Theorem: For every $(n, k, d)_q$ code, $n \geq (k + d - 1)$

Another way to look at this: $\mathrm{d} \leq (n - k + 1)$

Codes that meet Singleton bound with equality are called **Maximum Distance Separable (MDS)**

Recap: Maximum Distance Separable (MDS)

Only two binary MDS codes!

Q: What are they?

1. Repetition codes (k = 1)
2. Single-parity check codes (n-k = 1)

Need to go beyond the binary alphabet.

Finite fields!

Reed-Solmon (RS) codes

One of the most widely codes

- Storage systems, communication systems
- Bar codes (2-dimensional Reed-Solomon bar codes)

PDF-417

QR code

Aztec code

DataMatrix code

images: wikipedia

RS code: Polynomials viewpoint

Message: $[a_0, a_1, \ldots, a_{k-1}]$ where $a_i \in GF(q)$

k = # msg symbols

q = 256

Consider the polynomial of degree k-1

$$P(x) = a_{k-1} x^{k-1} + \cdots + a_1 x + a_0$$

variable

Coeffs = msg symbols

done in GF(q)

RS code: Codeword: $[P(\alpha_1), P(\alpha_2), \ldots, P(\alpha_n)]$
(distinct α_i's) $[c_0, c_1, \ldots, c_{n-1}]$

0 000
1 111

To make the α_i's in $P(\alpha_i)$ distinct, need field size $q \geq n$

That is, need sufficiently large field size for desired codeword length.

Minimum distance of an (n, k) RS code

Theorem: RS codes have minimum distance $\mathrm{d} = (n - k + 1)$

<u>Proof:</u> Any ideas?

Hint: Is it a linear code?

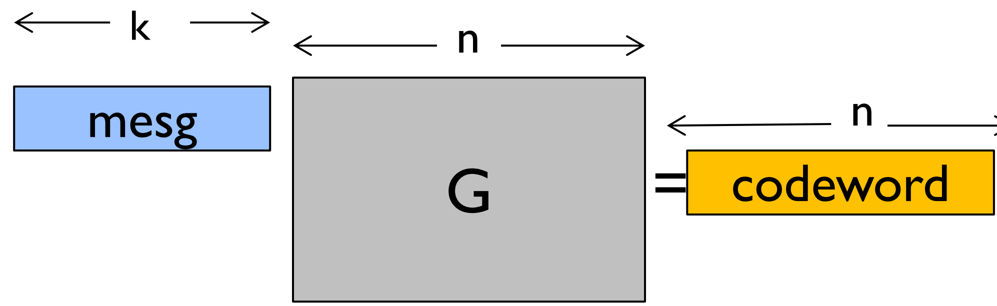
1. *RS is a linear code:* if we add two codewords corresponding to P(x) and Q(x), we get a codeword corresponding to the polynomial P(x) + Q(x). Similarly any linear combination..
2. *So look at the least weight codeword*. It is the evaluation of a polynomial of degree k-1 at some n points. So it can be zero on only k-1 points. Hence non-zero on at most (n-(k-1)) points. This means distance at least n-k+1

Apply Singleton bound

Meets Singleton bound: RS codes are MDS

Generator matrix of RS code

What is the generator matrix?



$[a_0\ a_1 \dots a_{k-1}]$

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & & \alpha_n^2 \\ \vdots & \vdots & & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & & \alpha_n^{k-1} \end{bmatrix}$$

"Vandermonde matrix"

Special property of Vandermonde matrices: Full rank (columns linearly independent)

Very useful in constructing codes.

Polynomials and their degrees

Fundamental theorem of Algebra: Any non-zero polynomial of degree m has at most m roots (over any field).

Corollary 1: If two degree-~~k~~m polynomials P, Q agree on m+1 locations (i.e., if $P(x_i) = Q(x_i)$ for $x_0, x_1, \ldots, x_m$), then P = Q.

Corollary 2: Given any m+1 points (x_i, y_i), there is at most one degree-m polynomial that has $P(x_i) = y_i$ for all these i.

Theorem: Given any m+1 points (x_i, y_i), there is exactly one degree-m polynomial that has $P(x_i) = y_i$ for all these i.

Proof: e.g., use Lagrange interpolation.

Decoding: Recovering Erasures

Recovering from at most (d-1) erasures:

$d = n-k+1$

\# erasures $= n-k$

Received codeword:

[P(α_1), *, P(α_2), …,*, P(α_n)]: at most (d-1) symbols erased (where * = erased)

Ideas?

1. At most n-k symbols erased
2. So have P(α_i) for at least k evaluations
3. Interpolation to recover the polynomial

Matrix viewpoint: Solving system of linear equations

Decoding: Correcting Errors

Correcting s errors: ($d \geq 2s+1$) $d-1 \geq 2s$ $\left\lfloor \frac{d-1}{2} \right\rfloor$

Naïve algo:

- Find k+s symbols that agree on a degree (k-1) poly P(x).
  - There must exist one: since originally k + 2s symbols agreed and at most s are in error

    (i.e., “guess” the n-s uncorrupted locations)

- Can we go wrong?

  Are there k+s symbols that agree on the wrong degree (k-1) polynomial P’(x)? No.
  - Any subset of k symbols will define P’(x)
  - Since at most s out of the k+s symbols are in error, P’(x) = p(x)

Decoding: Correcting Errors

Correcting s errors: (d ≥ 2s+1)

Naïve algo:

- Find k+s symbols that agree on a degree (k-1) poly P(x).
  - There must exist one: since originally k + 2s symbols agreed and at most s are in error
    (i.e., “guess” the n-s uncorrupted locations)

This suggests a brute-force approach, very inefficient.
“guess” = “enumerate”, so time is (n choose s) ~ n^s.

More efficient algorithms exist:
“The Berlekamp Welch Algorithm” (results in solving a system of n linear equations)

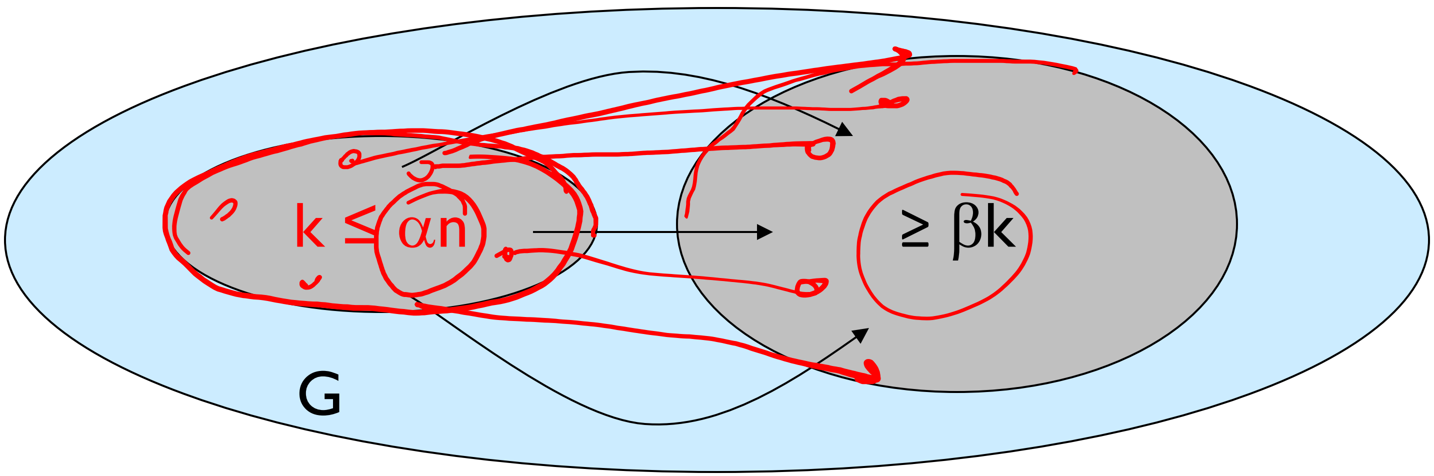
Codes based on graphs

- Optimized for fast (de)coding
- Based on graphical constructions
- Constructions based on properties of expander graphs

LDPC

(α, β) Expander Graphs (non-bipartite)

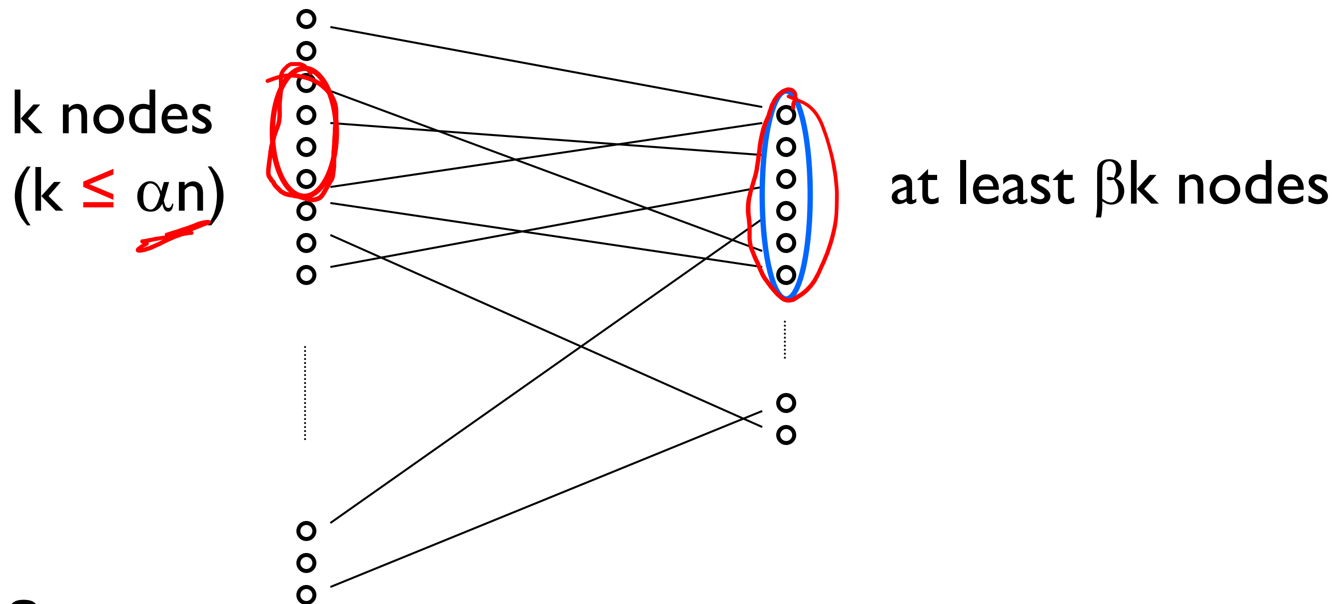
n nodes



Properties

- **Expansion:** every small subset ($k \leq \alpha n$) has many ($\geq \beta k$) neighbors
- **Low degree –** not technically part of the definition, but typically assumed

(α, β) Expander Graphs (bipartite)



Properties

- **Expansion:** every small subset ($k \leq \alpha n$) on left has many ($\geq \beta k$) neighbors on right
- **Low degree –** not technically part of the definition, but typically assumed

d-regular graphs

An undirected graph is **d-regular** if every vertex has d neighbors.

A **bipartite** graph is **d-left-regular** if every vertex on the left has d neighbors on the right.

We consider only d-left-regular constructions.

(And call it d-regular with abuse of notation.)

Expander Graphs: Constructions

Important parameters: size (n), degree (d), expansion (β)

Randomized constructions

- A random d-regular graph is an expander with a high probability
- Time consuming and cannot be stored compactly

Explicit constructions

- Cayley graphs, Ramanujan graphs etc
- Typical technique – start with a small expander, apply operations to increase its size

Expander Graphs: Constructions

Theorem: For every constant 0 < c < 1, can construct bipartite graphs with

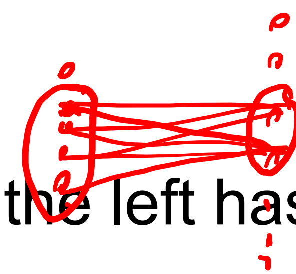
n nodes on left,

cn on right,

d-regular (left),

(α, β)

that are $(\alpha, 3d/4)$ expanders, for constants α and d that are functions of c alone.



"Any set containing at most alpha fraction of the left has (3d/4) times as many neighbors on the right"

TORNADO CODES

Luby Mitzenmacher Shokrollahi Spielman 2001

Tornado codes

Goal: low (linear-time) complexity encoding and decoding

We will focus on **erasure** recovery

- Each bit either reaches intact, or is lost.
- We know the positions of the lost bits.

The random erasure model

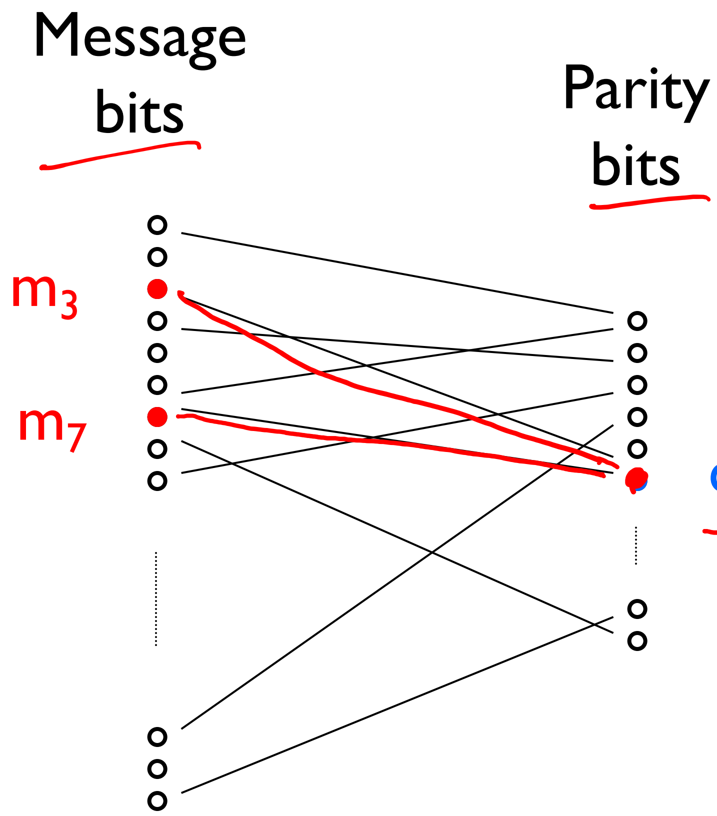
Random erasure model:

- Each bit is erased with some probability p (say ½ here)
- Known: a random linear code with rate < 1-p works (why?)

For simplicity.

Can be extended to worst-case error, and bit corruption with extra effort.

[e.g., Spielman1996]

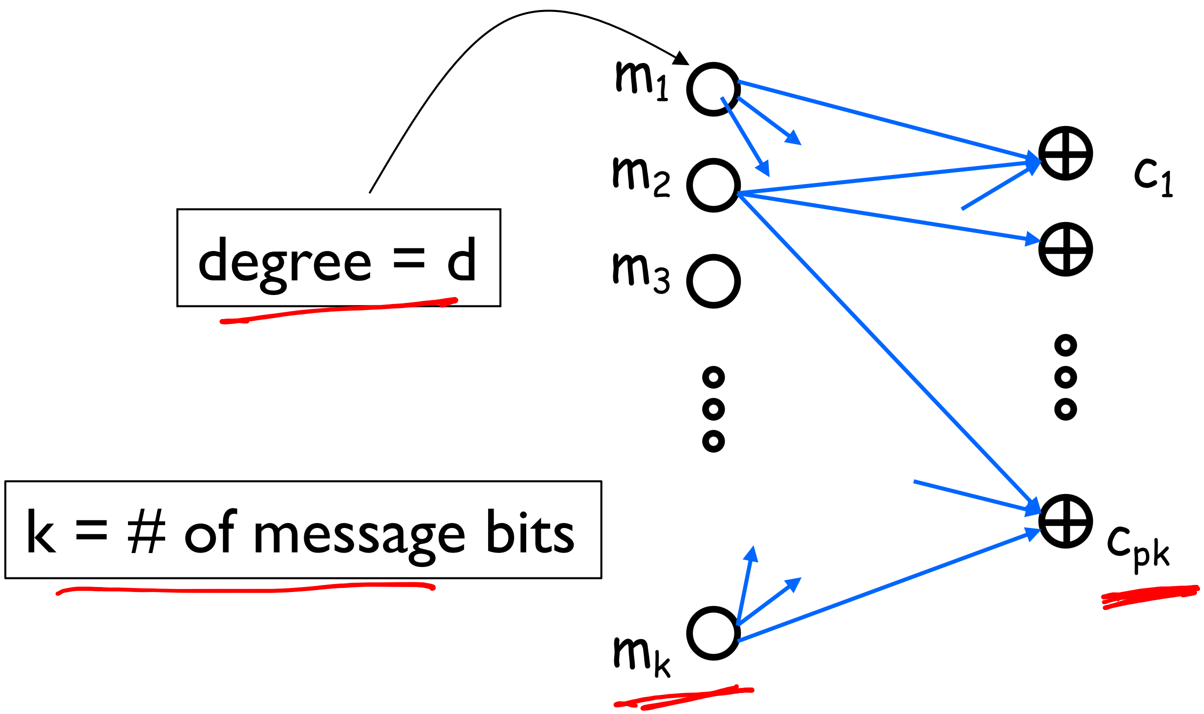


$c_6 = m_3 \oplus m_7$

Tornado codes

- Have d-left-regular bipartite graphs with k nodes on the left and pk on the right.

#edges=kd

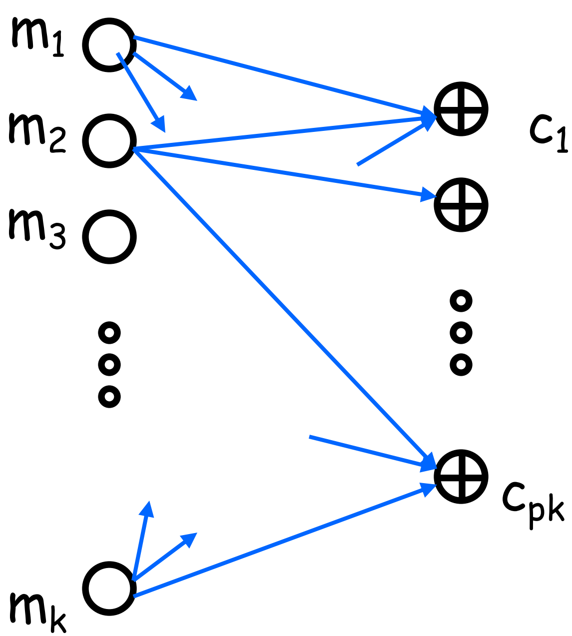


- Let's again assume 3d/4-expansion.

Tornado codes: Encoding

Why is it linear time?

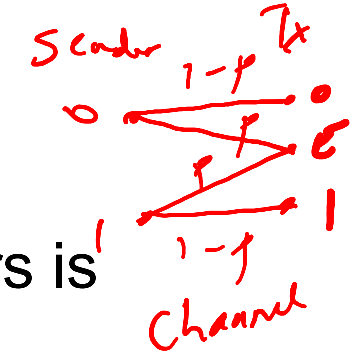
(Hint: Look at the number of edges)



Computes the sum modulo 2 of its neighbors

Number of edges = kd

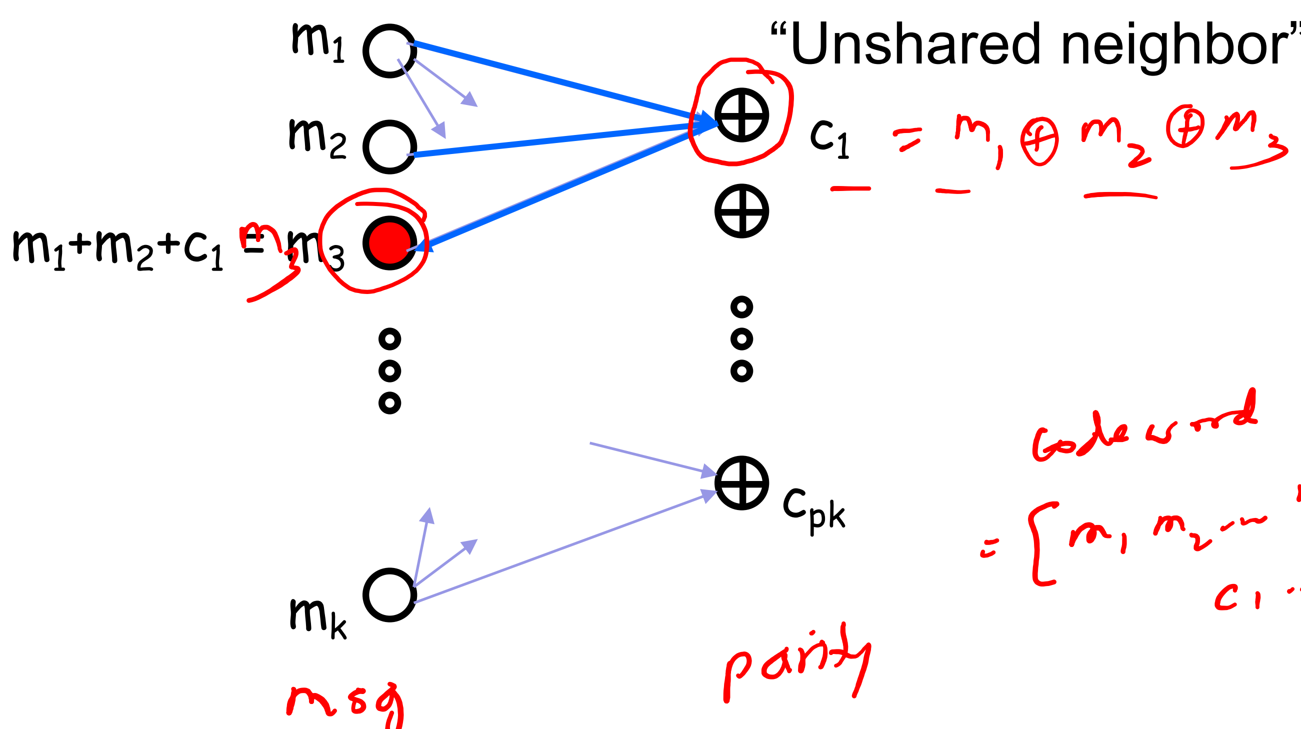
Tornado codes: Decoding



First, assume that all the parity bits are intact

Find a parity bit such that only one of its neighbors is erased (an *unshared neighbor*)

Fix the erased bit, and repeat.



codeword $= [m_1, m_2 \ldots m_k \; c_1 \ldots c_{pk}]$

Tornado codes: Decoding

Intuition:

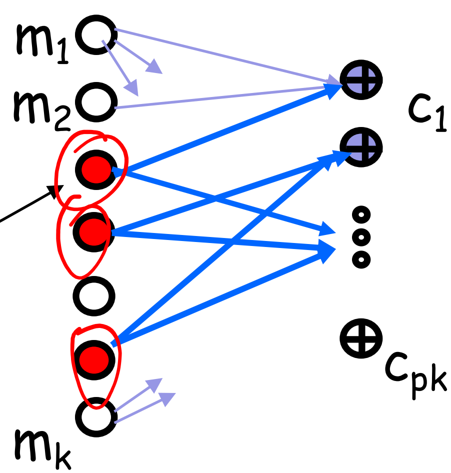
Want to always find such a parity bit with “Unshared neighbor” property.

Consider the set of corrupted message bit and their neighbors.

(Suppose this set is small.)

=> at least one message bit has an unshared neighbor.

Has an unshared neighbor



Tornado codes: Decoding

Can we always find unshared neighbors?

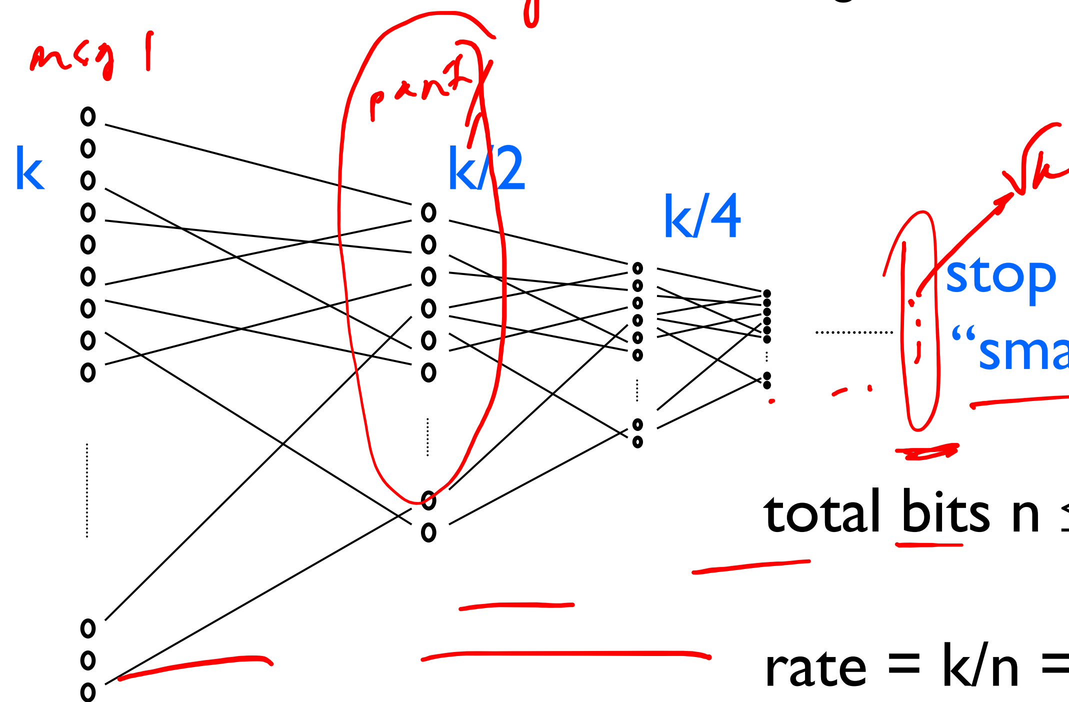
Expander graphs give us this property if expansion > d/2
(similar argument to one above)

Also, [Luby et al] show that if we construct the graph from a specific kind of degree distribution, then we can always find unshared neighbors.

What if parity bits are lost?

Cascading

- Use another bipartite graph to construct another level of parity bits for the parity bits
- Final level is encoded using RS or some other code



stop when $k/2^t$ "small enough"

total bits $n \le k(1 + ½ + ¼ + \ldots)$
$= 2k$

rate = k/n = ½.

(assuming p =1/2)

Tornado codes enc/dec complexity

Encoding time?

- for the first t stages : |E| = d x |V| = O(k)
- for the last stage: poly(last size) = O(k) by design.

Decoding time?

- start from the last stage and move left
- Last stage is O(k) by design
- Rest proportional to |E| = O(k)

So get very fast (linear-time) coding and decoding.

100s-10,000 times faster than RS