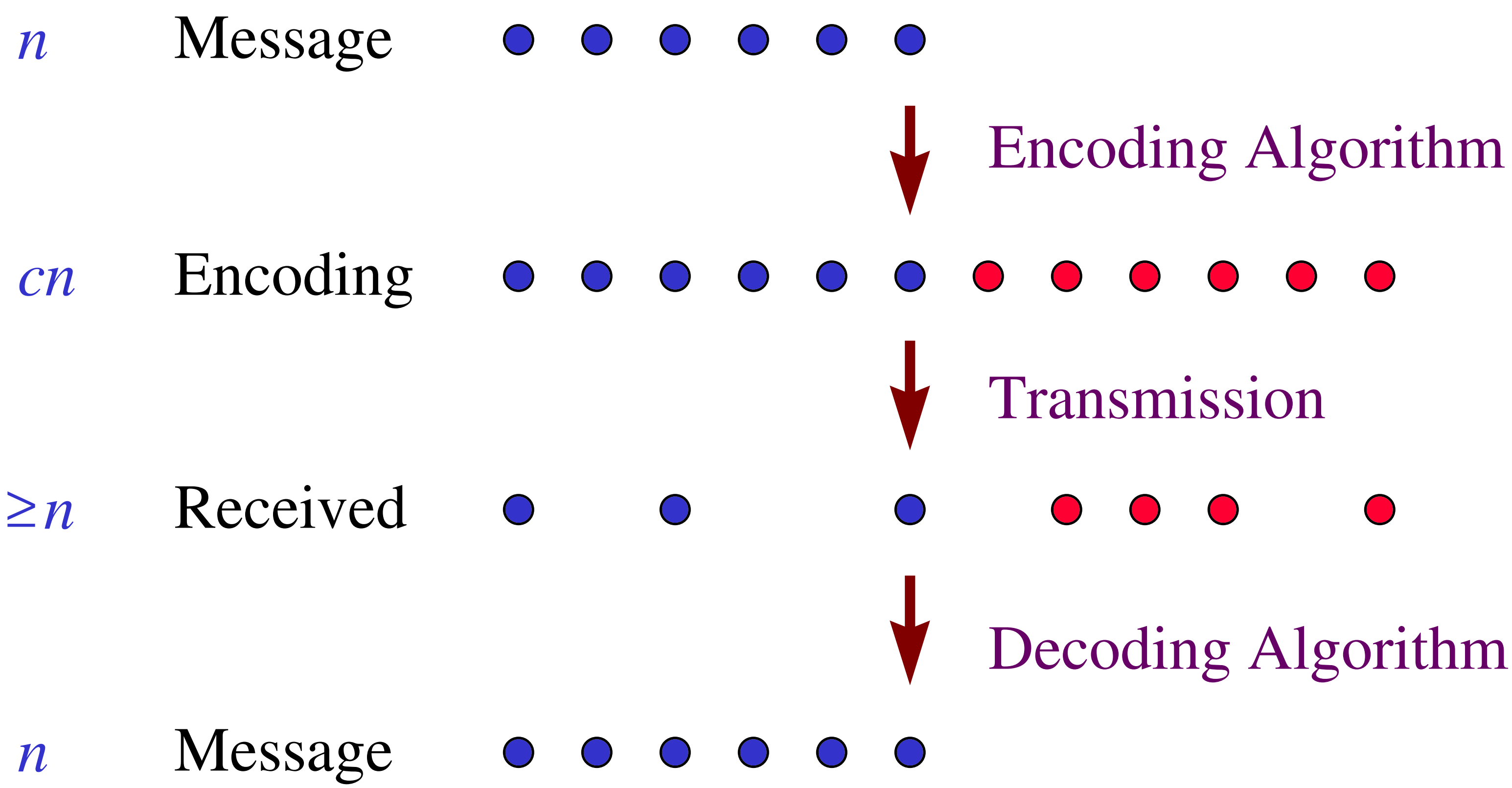
Practical Loss-Resilient Codes

Michael Luby
Michael Mitzenmacher
Amin Shokrollahi
Daniel Spielman
Volker Stemann

What are they useful for?

- Applications
  - Internet video/audio transmission
  - Wireless transmission
  - Multicast
- Problems
  - Data corruption
  - Packet loss
  - Congestion (real-time transmission)
- Error-correcting codes
  - Detects/corrects data corruption
- Loss-resilient code
  - Protects against losses
  - Forward Error Correction

What are they?



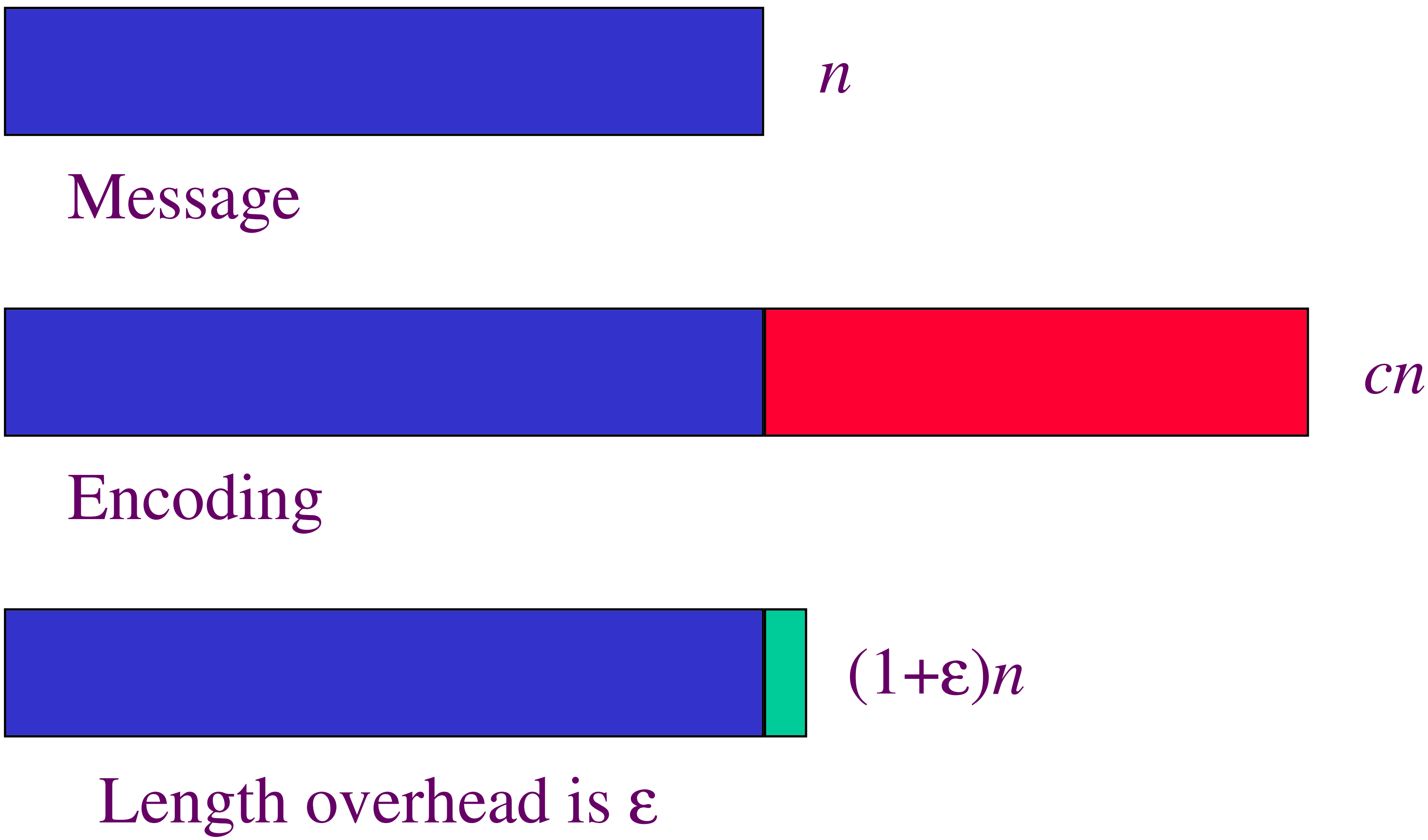
Performance Measures

- Time Overhead
  - The time to encode and decode expressed as a multiple of the encoding length.
- Length Overhead
  - The amount of additional encoding needed beyond the message length to decode expressed as a fraction of the message length.

Length Overhead

- Optimal
  - Can decode from any n words of encoding.
  - Length overhead is zero.
- Relaxation
  - Decode from any $(1+\varepsilon)\, n$ words of encoding
  - Length overhead is ε.

Parameters of the Code



Previous Work

- Length overhead is zero.
  - Standard
    - Time overhead is number of redundant packets.
  - Fast Fourier-based
    - Time overhead is $\ln^2 n$ field operations.
- Length overhead is ε.
  - Random mixed-length linear equations
    - Time overhead is $\ln(1/\varepsilon)/\varepsilon$.

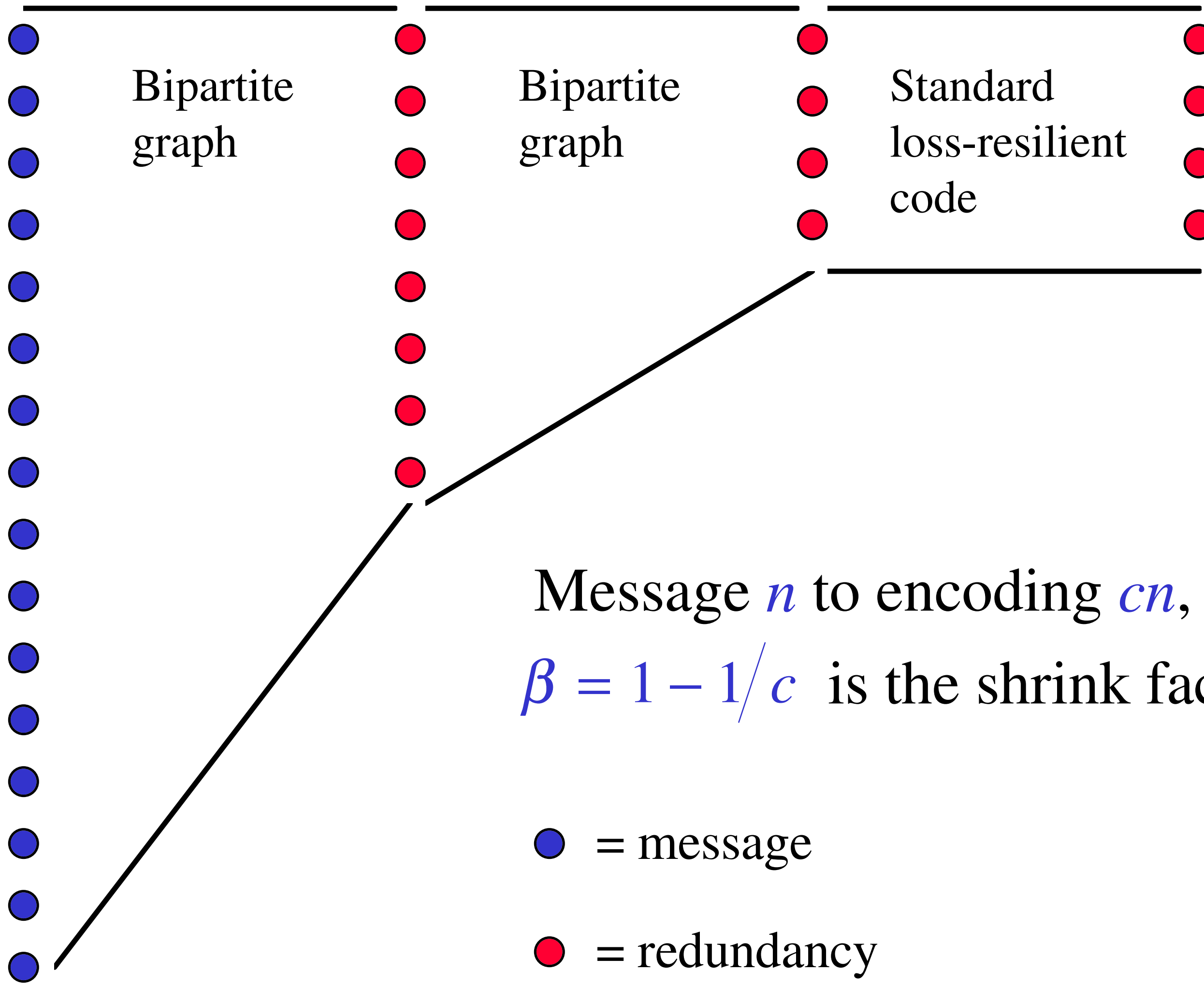
Our Performance

- Length overhead is ε.
- Time overhead is $\ln(1/\varepsilon)$.
- Simple, fast, and practical.

Transmission Model

- Which packets received
  – Can depend on when sent, route taken.
  – Does not depend on payload.
- Algorithm
  – Compute entire encoding.
  – Place encoding into packets in random order.
- Implication
  – Random portion of encoding is received.
  – Same amount from each level.

Encoding Structure



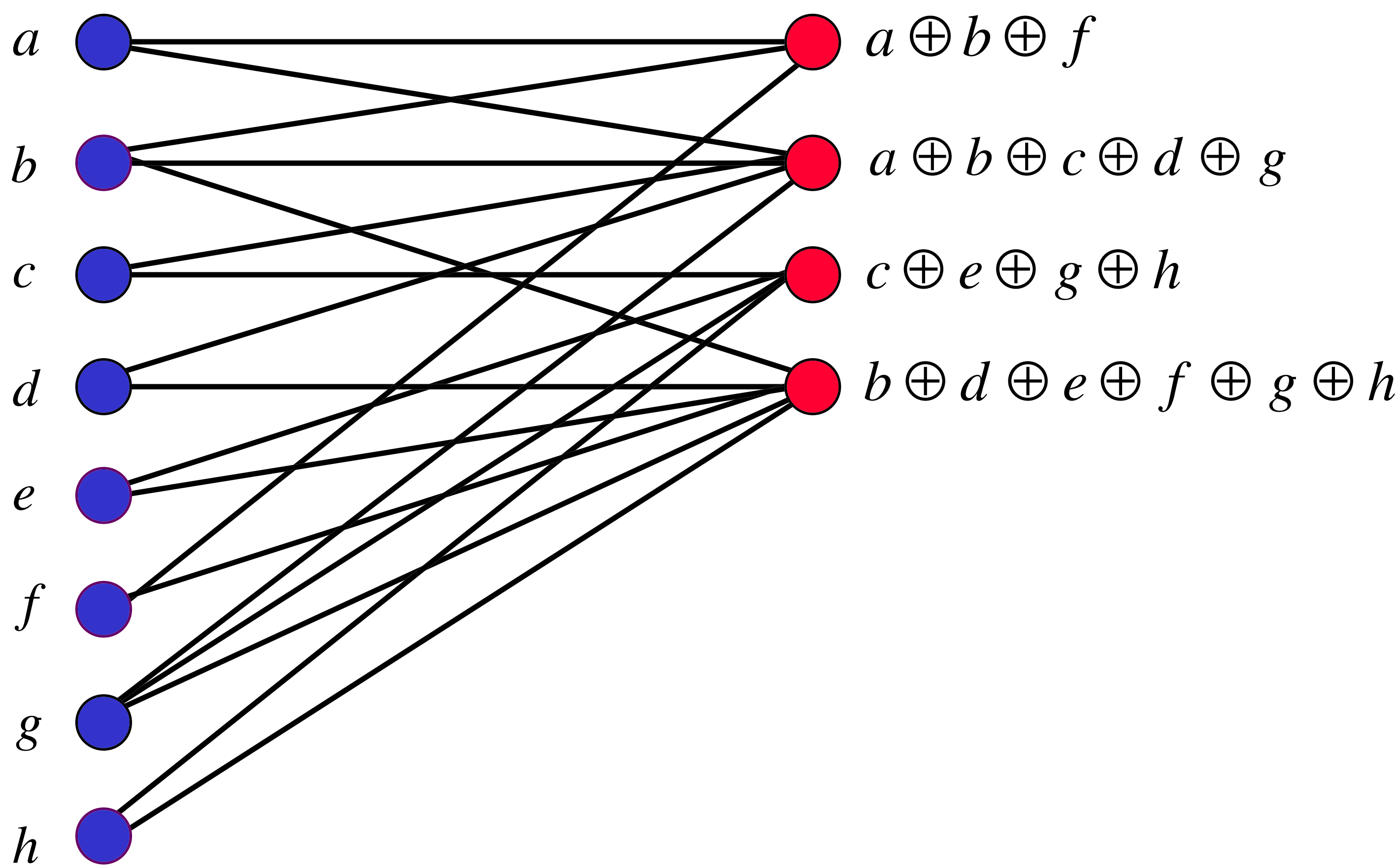
Message n to encoding cn, c=2

$\beta = 1 - 1/c$ is the shrink factor

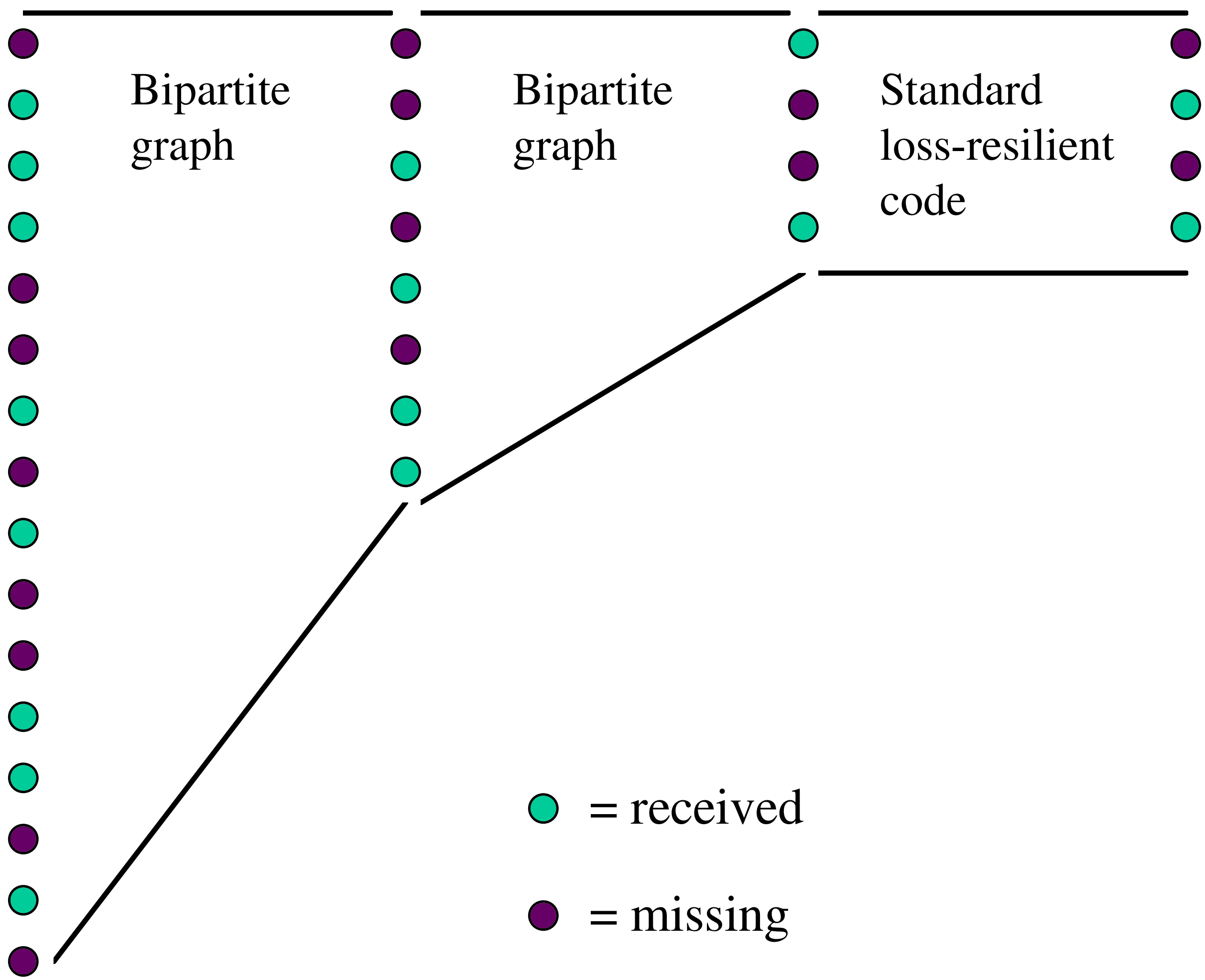
● = message

● = redundancy

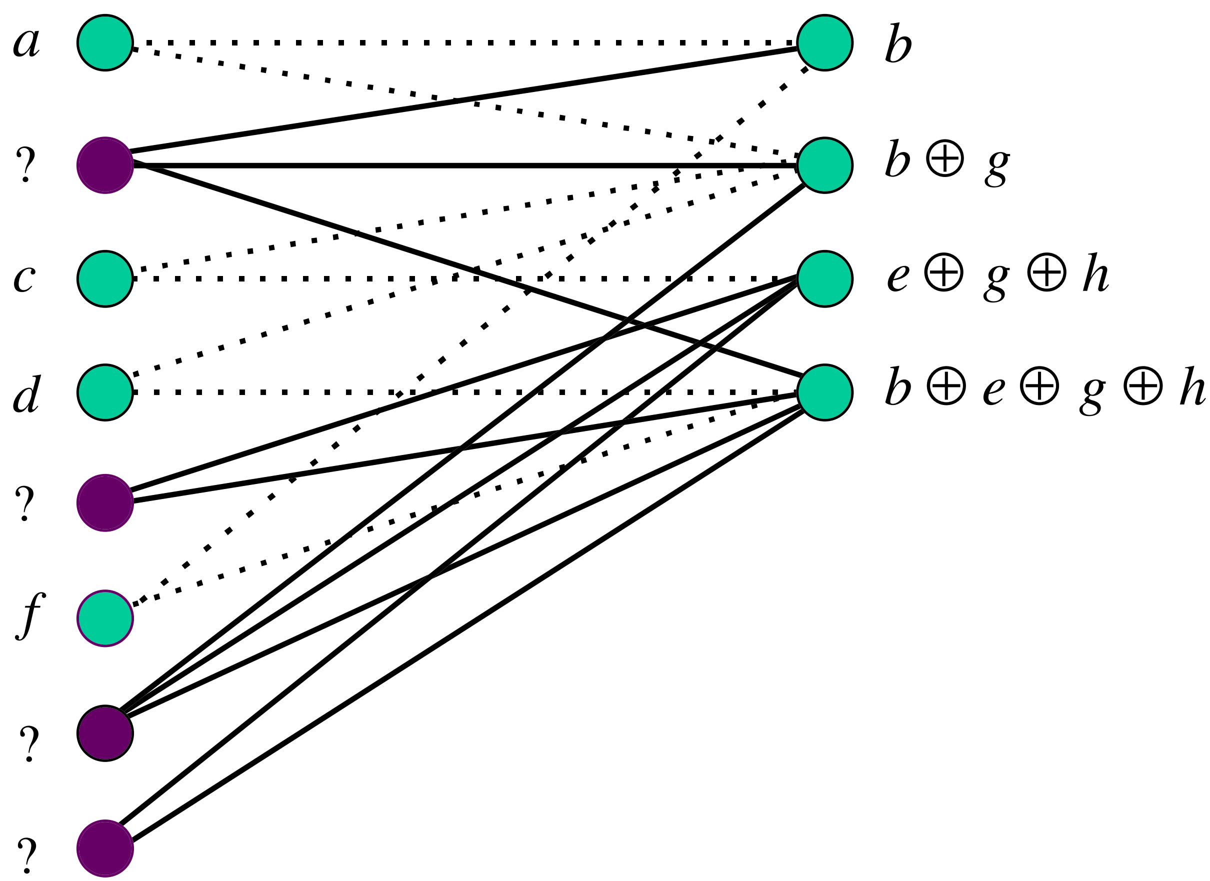
Encoding Process



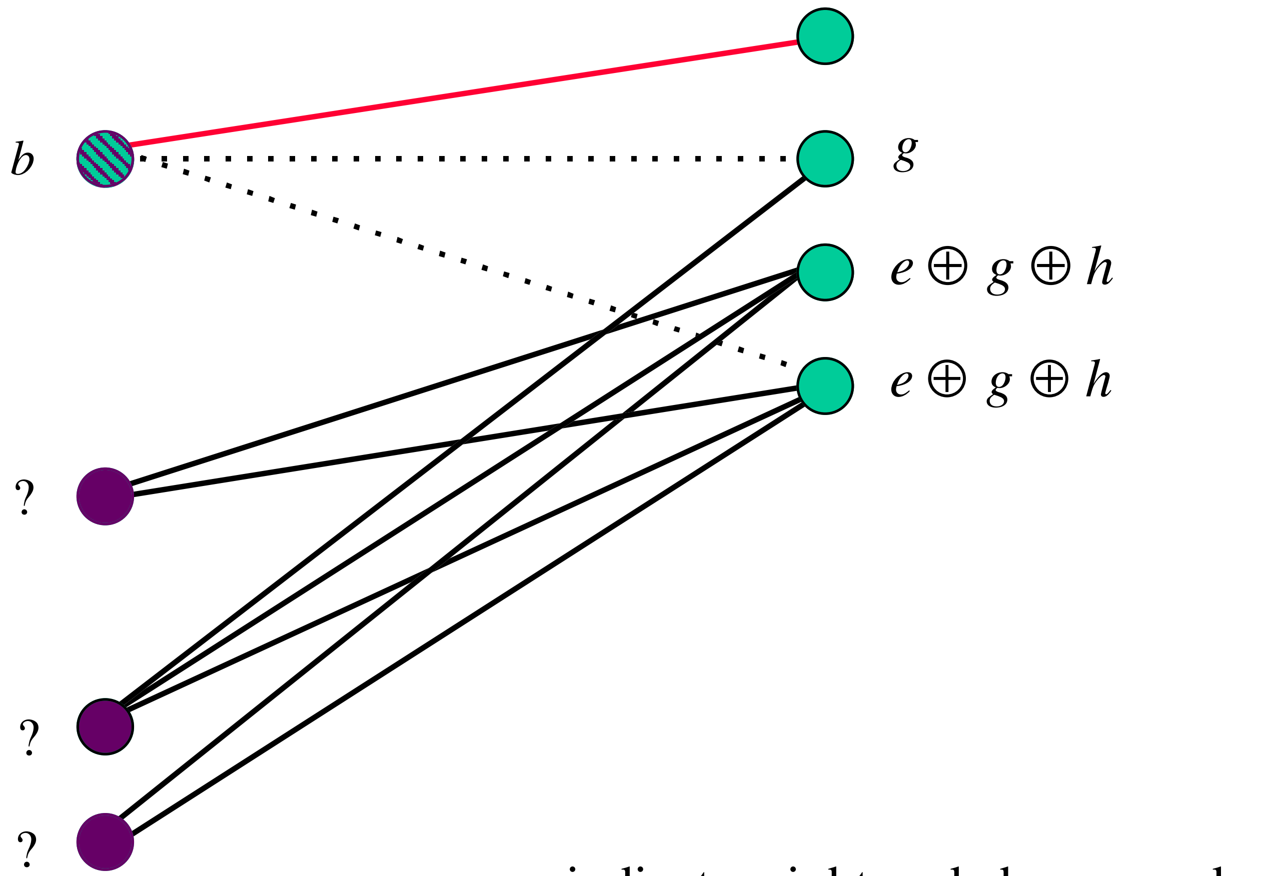
Decoding Structure



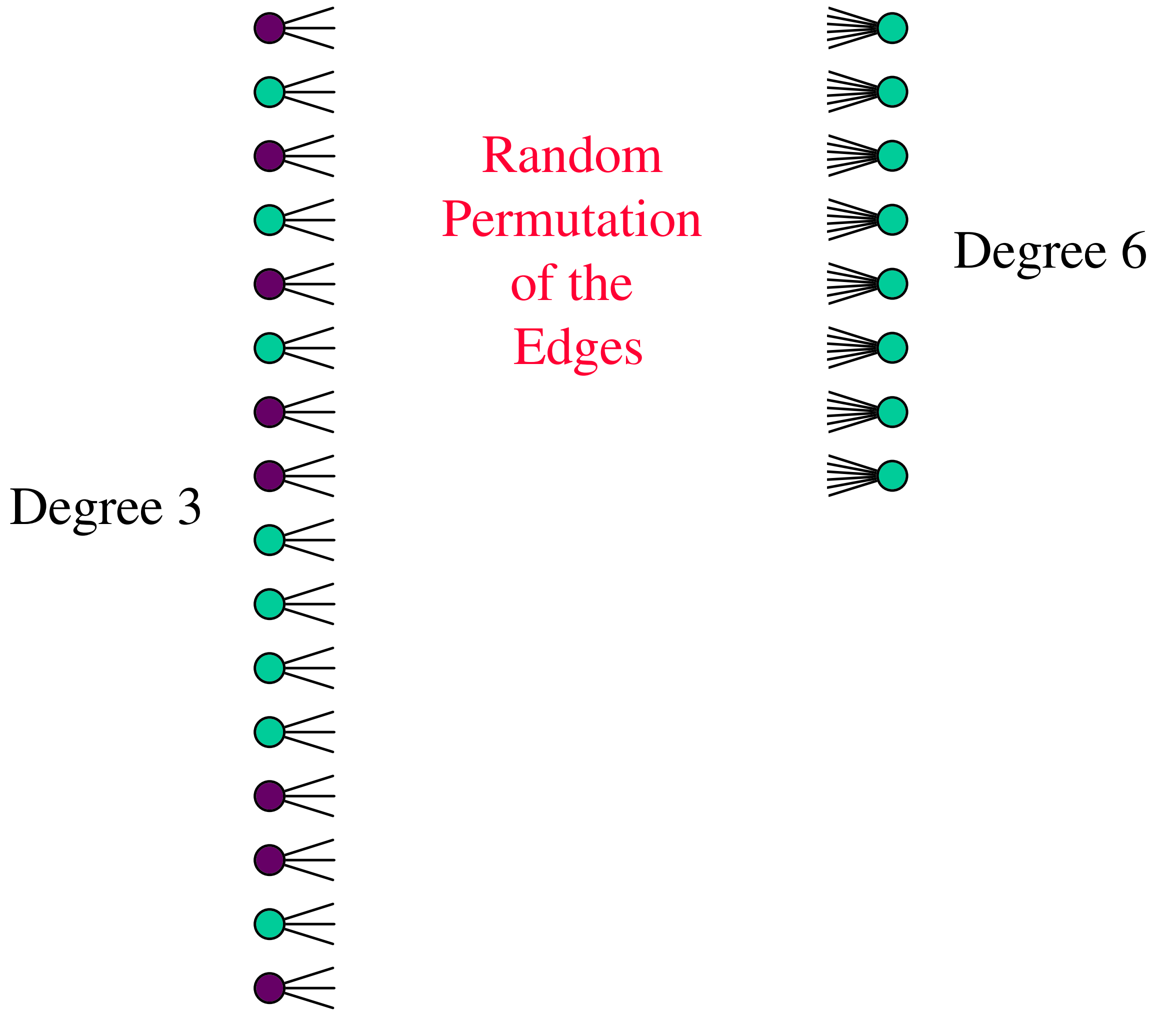
Decoding Process: Direct Recovery



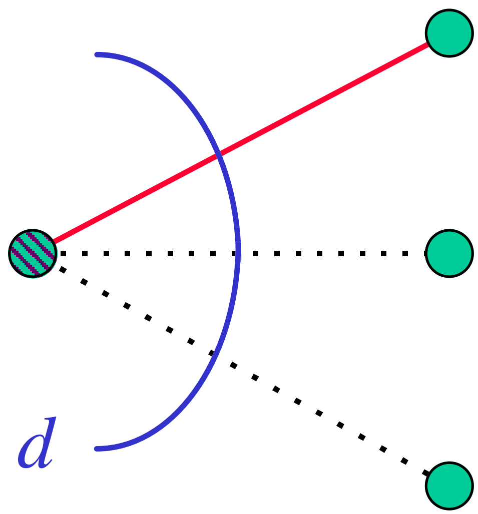
Decoding Process: Substitution Recovery



Regular Graphs



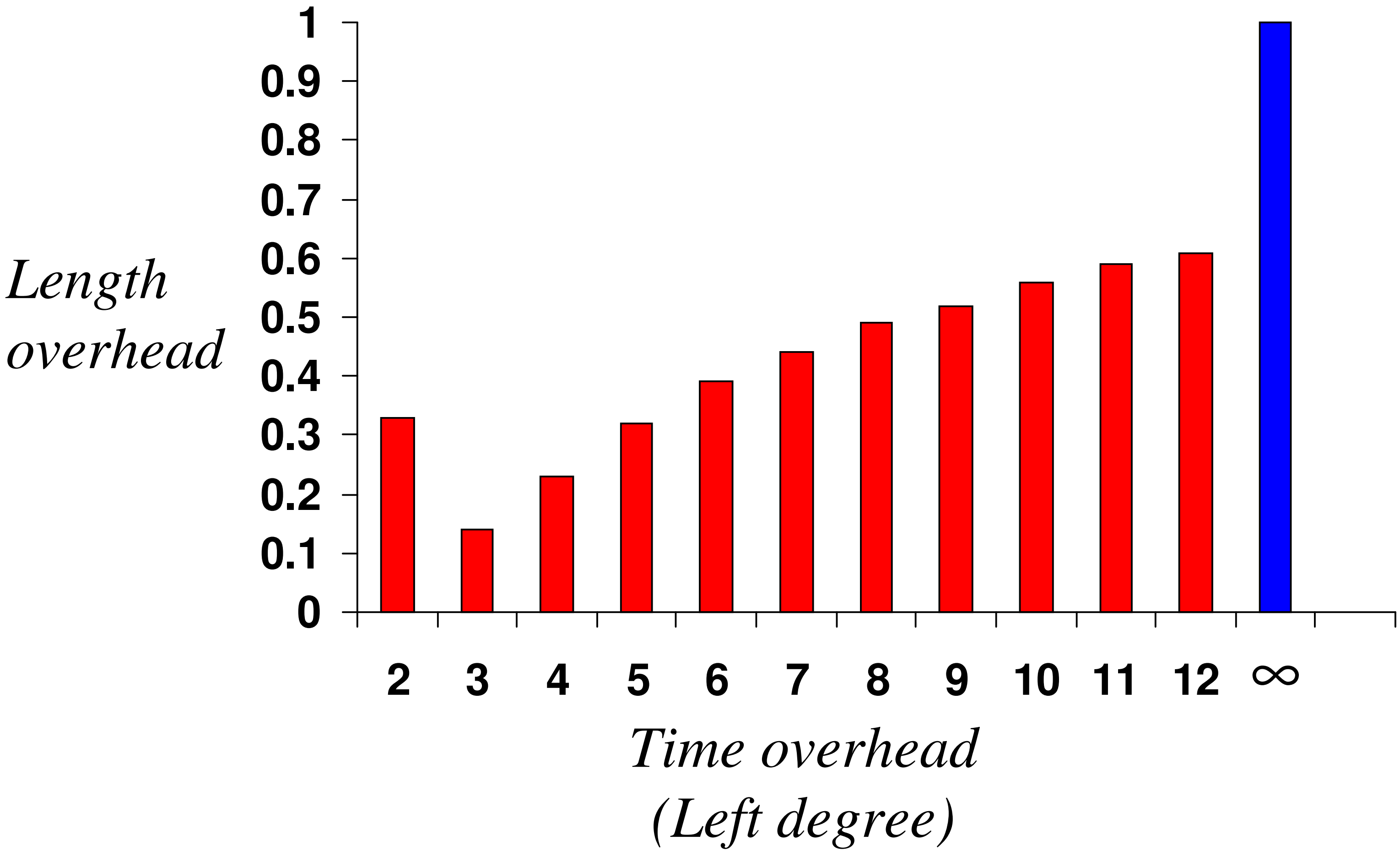
Why regular graphs are bad



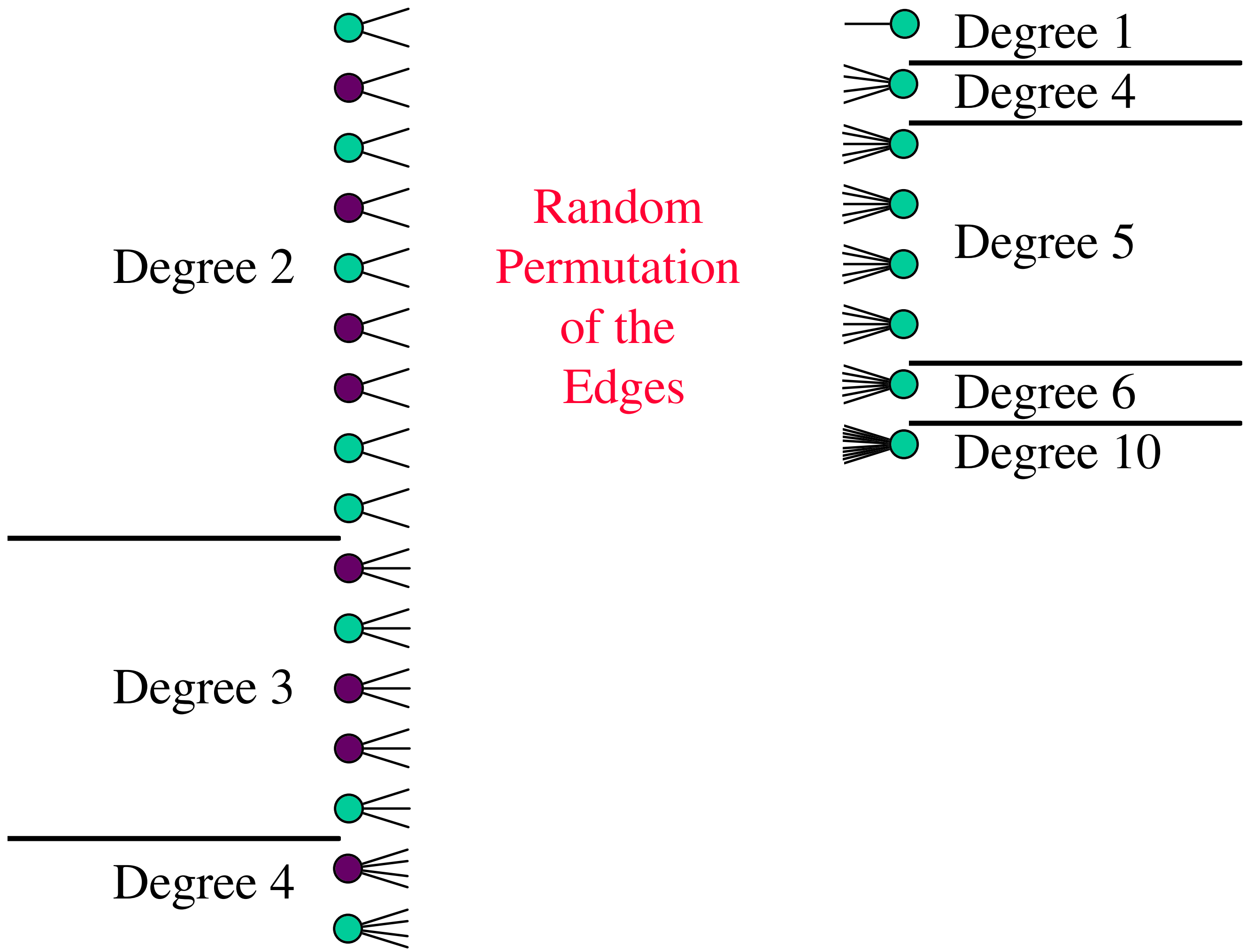
Right degree $2d$
implies
Pr[right degree =1] $= 1/2^{2d-1}$

Left node has on average
$d/2^{2d-1}$ neighbors of degree one.

Regular Graph Performance



Irregular Graphs



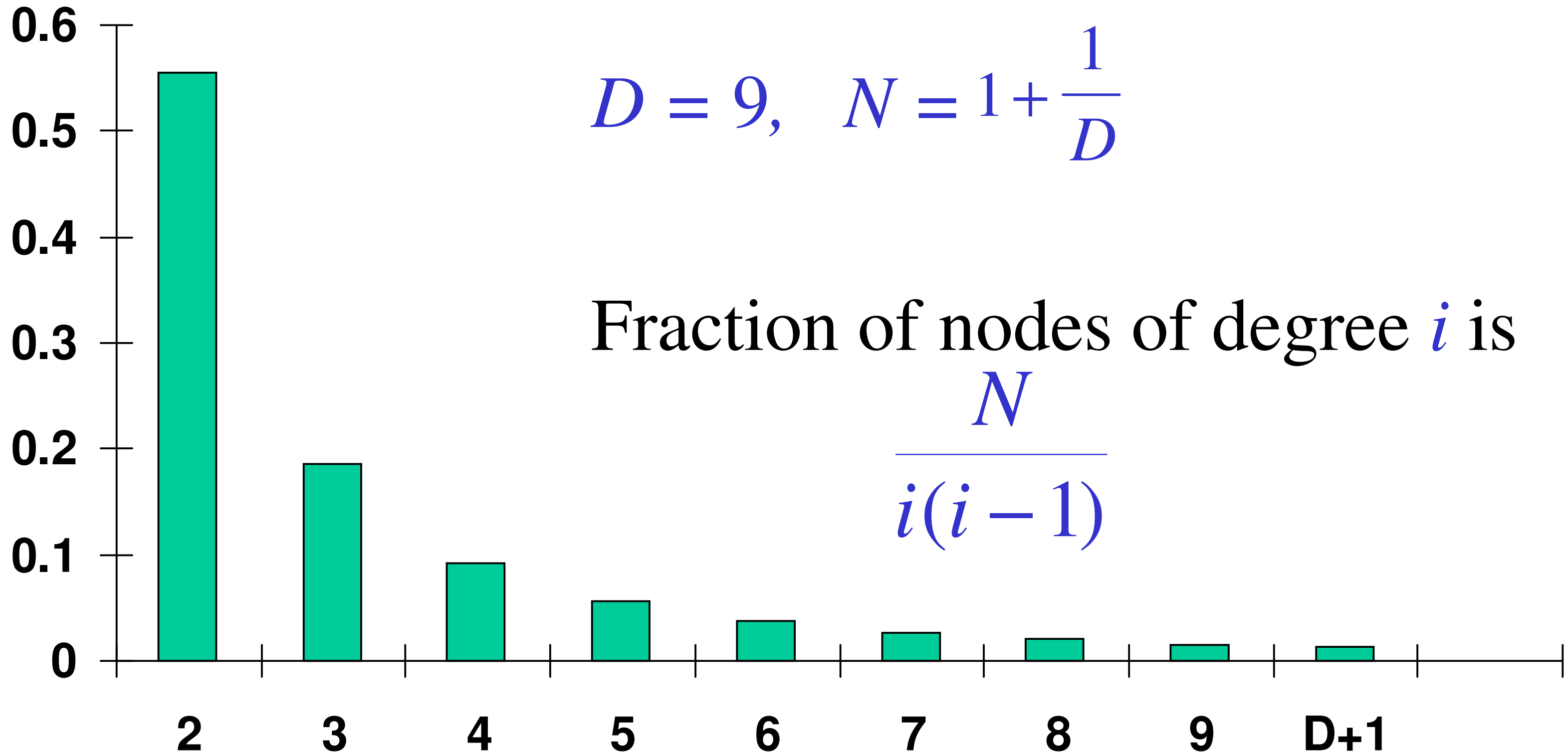
Random Graph Design

- Degree sequence is crucial!
- Decoding described as a process that depends on the degree sequence.
- Express the process using differential equations.
- Solve the differential equations.
- Based on the solution, design a degree sequence with low average degree and a small length overhead.

A Good Degree Sequence Design

- Positive integer D is a design parameter
  - $D+1$ is the maximum left degree.
  - $\ln(D)$ is the average left degree.
- Tradeoff of time and length overheads as a function of D.

Good Left Degree Sequence: Truncated Heavy Tail



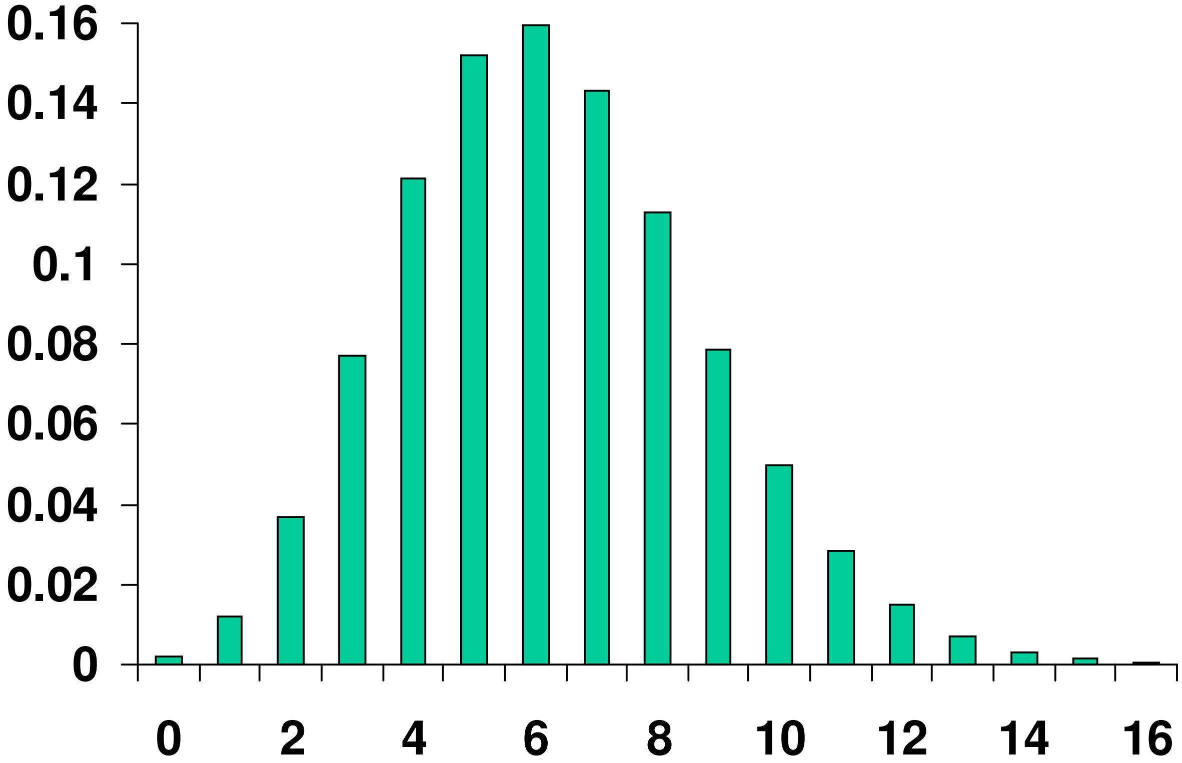
$D = 9, \quad N = 1 + \frac{1}{D}$

Fraction of nodes of degree i is

$$\frac{N}{i(i-1)}$$

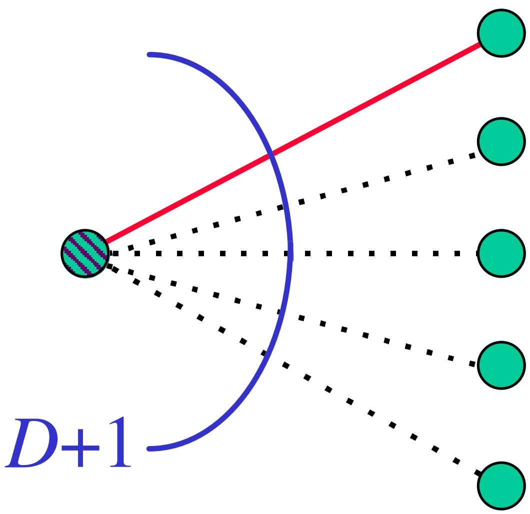
Average node degree is $N \cdot H(D) \approx \ln(D)$

Good Right Degree Sequence: Poisson



Average node degree is $N \cdot H(D)/\beta \approx \ln(D)/\beta$

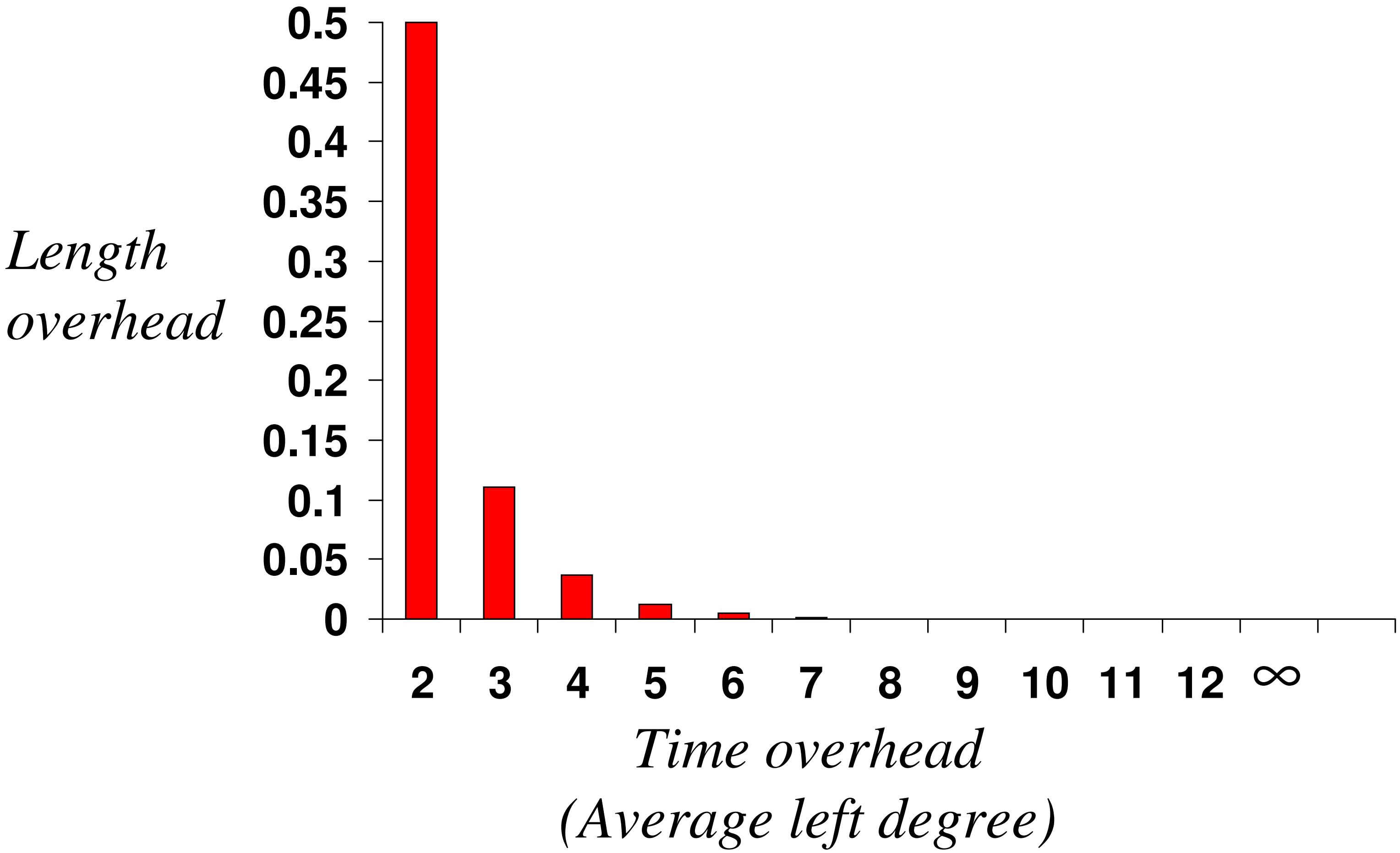
Why irregular graphs are good



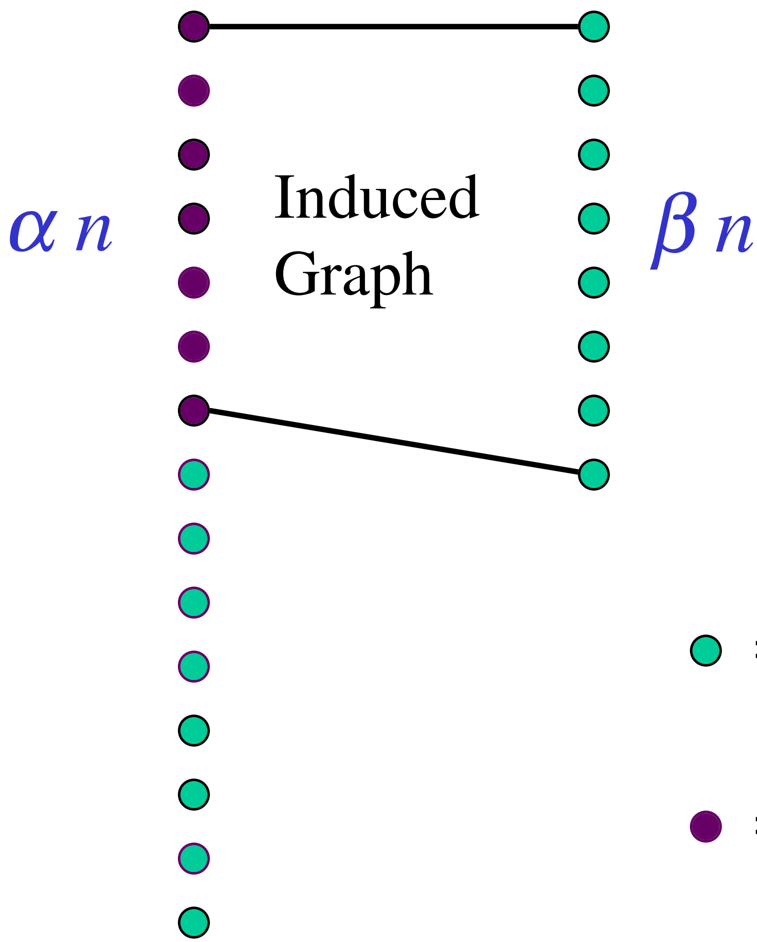
Average right degree $2\ln(D)$
implies
Pr[right degree =1] ~ $1/(D+1)$

Left node of max degree has on average
one neighbor of degree one.

Irregular Graph Performance



Decoding Process Analysis



= Recovered

= Missing/not yet recovered

Differential Equations

- $t = 1,\ldots,\alpha n$ no. of decoded left nodes.
- $E(t)$ no. of edges at time t.
- $A(t)$ average left edge degree.
- $L_i(t)$ no. of edges of degree i on left.
- $R_i(t)$ no. of edges of degree i on right.

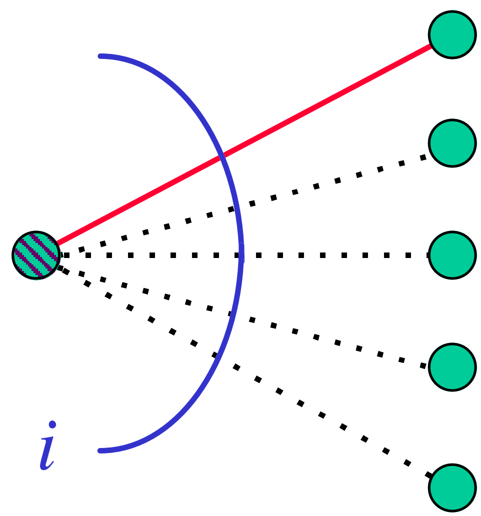
Successful Decoding

The process completes the decoding sucessfully if $R_1(t) > 0$ for all

$$t \in \{1, \ldots, \alpha n\}$$

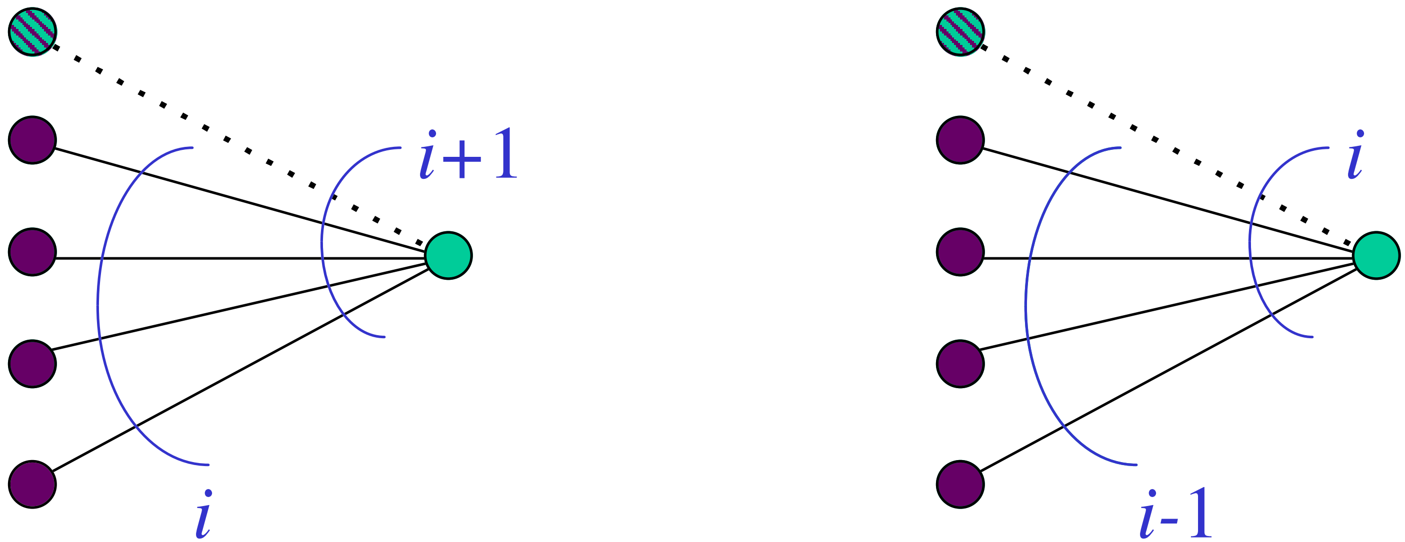
How can we describe this criterion explicitly?

Left Side Evolution



$$L_i(t+1) - L_i(t) = -\frac{iL_i(t)}{E(t)}$$

Right Side Evolution



$i > 1, \quad i = 1$

$$R_i(t+1) - R_i(t) = \left(\frac{iR_{i+1}(t)}{E(t)} - \frac{iR_i(t)}{E(t)} \right) (A(t) - 1) \quad -1$$

Degree Sequence Functions

- Left Side
  - λ_i fraction of edges of degree i on the left in the *original* graph.
  - $\lambda(x) := \sum \lambda_i x^{i-1}$.
- Right Side
  - ρ_i fraction of edges of degree i on the right in the *original* graph.
  - $\rho(x) := \sum \rho_i x^{i-1}$.

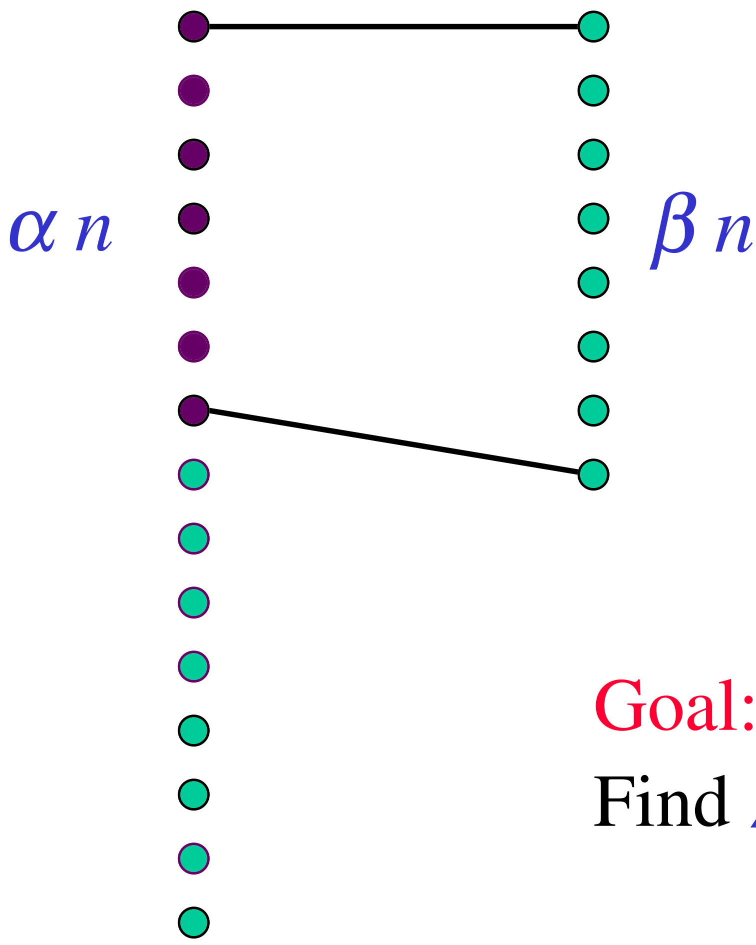
Explicit Condition

Decoding process can recover
a missing fraction α nodes on the left
with high probability iff:

$$\rho\left(1-\alpha\lambda(1-x)\right) > x$$

for all x in $[0,1)$.

Designing the Graph



Condition:

$$\rho\left(1-\alpha\lambda(1-x)\right) > x$$

Goal:

Find λ and ρ so that $\alpha \approx \beta$

Length Overhead Analysis

The value $\alpha = \dfrac{\beta}{1+1/D}$

satisfies the condition for all x in $[0,1)$
for the λ and ρ associated with
the Good Truncated Heavy Tail/Poisson
degree sequences described previously.

Time-Length Overhead Tradeoffs

- Time overhead is $\ln(D)$.
- Length overhead is $\dfrac{c-1}{D+1}$.

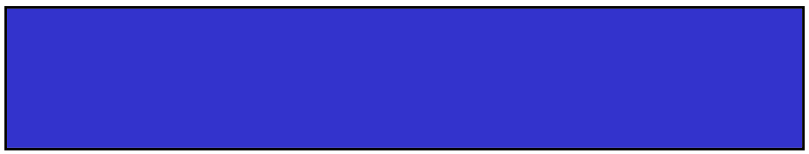
From Expected to High Probability

- Differential equations
  - model expected behavior of process.
- Kurtz's Theorem
  - Law of large numbers for Markov process.
  - Provides Chernoff-like bounds.
- End Game
  - Small sets of left nodes are expanders.

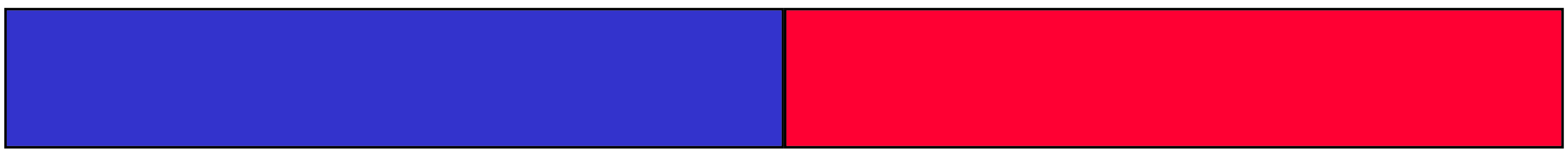
Implementation Issues

- Smaller graphs perform slightly worse than larger graphs.
- A lot of time is spent in graph setup and pointer chasing.
- Additional structure required at the first level for degree two nodes on the left.
- Simpler methods than standard loss-resilient codes at the last layer work.

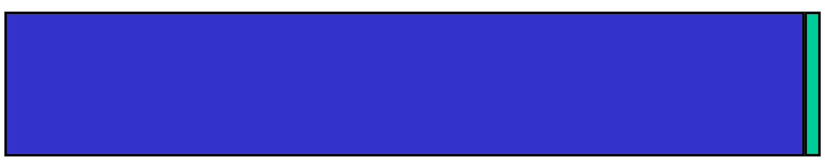
Irregular Example



640,000 message packets



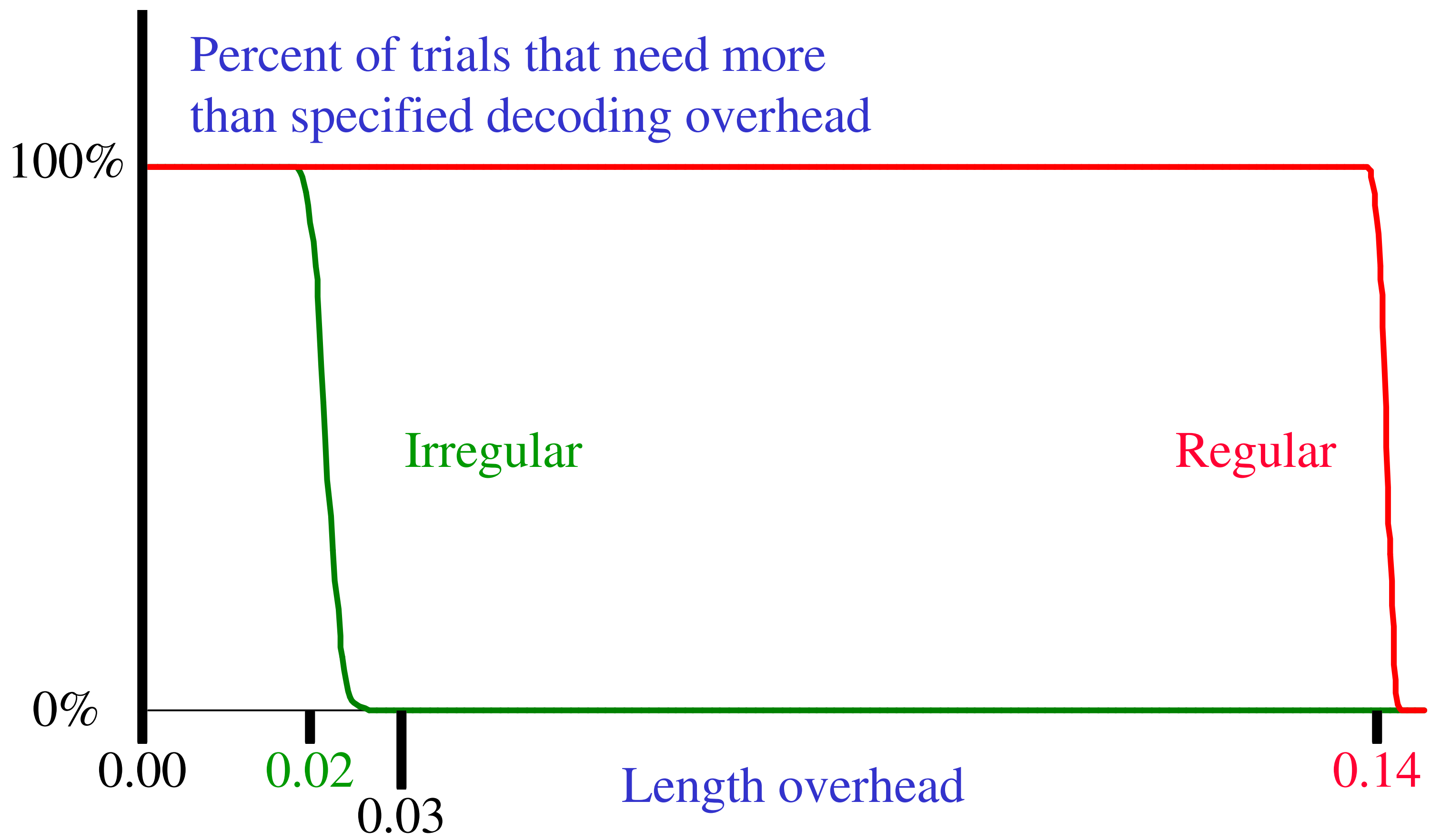
1,280,000 encoding packets



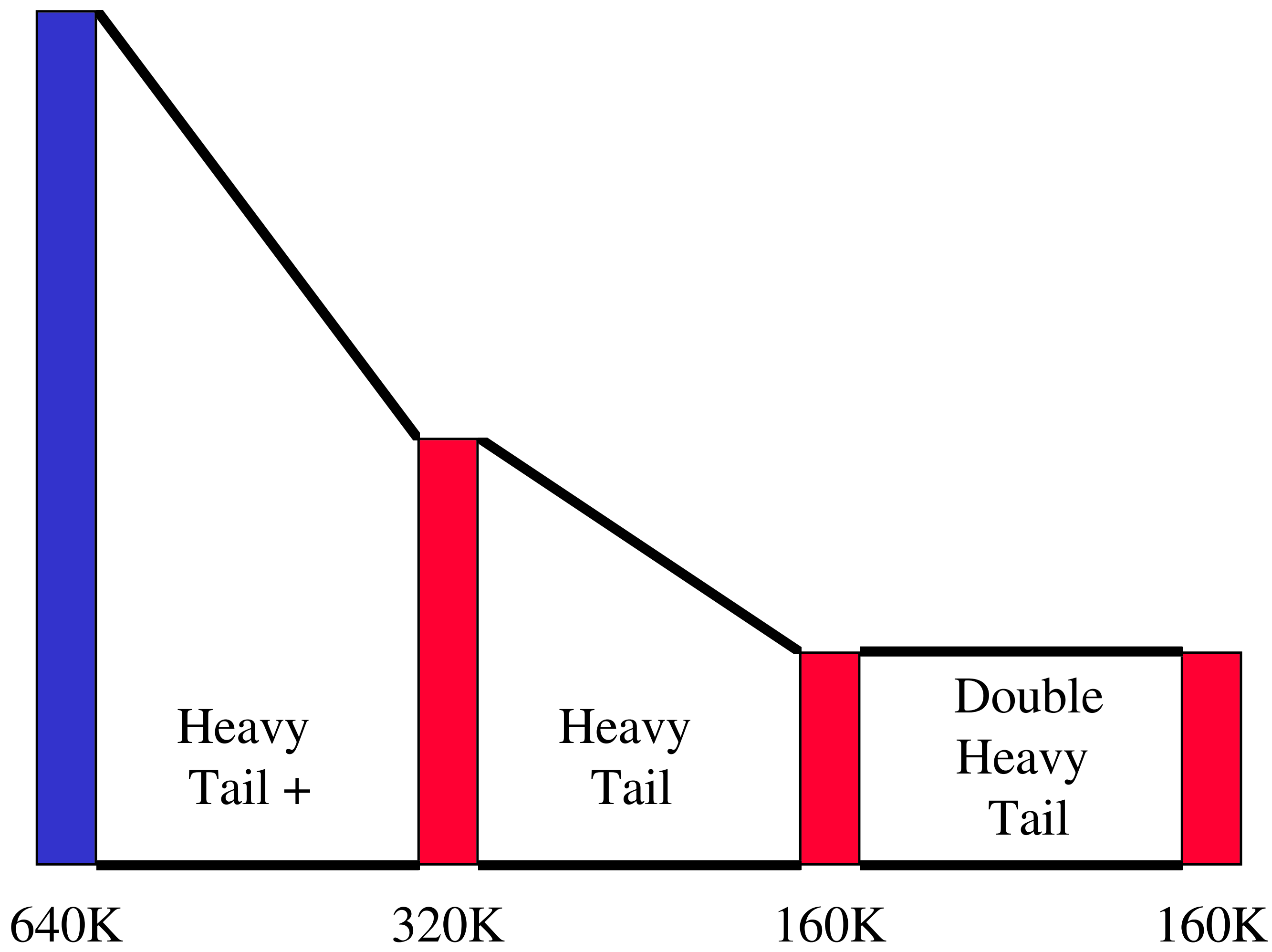
0.0213 ± 0.00115 Length overhead (based on 1000 trials)

Time Overhead is 8

Length Overhead Statistics



Cascade for Example



Good Right Degree Sequences

- Linear Programming Approach
  - Given a left degree sequence, find good matching right degree sequence.
- Example
  - 1/5 edges with left degree: 3, 5, 9, 17, 33.
  - Rate 2/3 code.
  - Time overhead is 6.82.
  - Length overhead is 0.013.

Open Questions

- Deterministic choice for graph?
- Linear time with zero length overhead?
- How will it be used?
- Are there other applications of the general methodology developed in this work?