Optimizing the Degree Distribution of LT Codes

Master's Thesis

Optimizing the Degree Distribution of LT Codes

- Work done in Networking Laboratory for PAN-NET-project (Performance Analysis of New Networking Technologies), funded by Tekes, Nokia and Rommon (now part of F-Secure)
- Professorship: S-38 Teletraffic theory
- Supervisor: Prof. Jorma Virtamo
- Instructor: Esa Hyytiä, D.Sc. (Tech.) (At NTNU, Norway)

Outline

- Background
- FEC codes for erasure channels
- LT Codes
- Degree Distributions
- Optimizing the degree distribution
- Applications for erasure codes
- Implementation Issues
- Test results

Background

- Coding theory is used to either remove redundancy from messages (source coding) or to add redundancy in order to make error correction possible (error-correcting codes)
- We focus on the error-correcting codes
- Different schemes ARQ, FEC, muting
- For example, TCP uses retransmissions to deal with missing packets ("high level ARQ")
- Retransmissions are inefficient for example in multicast situations
- We focus on erasure channels where data is either transferred correctly, in case of errors packets are dropped (cf. Internet)

Erasure codes

- File divided into n blocks is distributed.
- Erasure codes can be used to encode the original data, resulting in $n + m$ blocks
- These blocks are then distributed, any $n' \geq n$ blocks are sufficient to recover the original file (**overhead factor** $f = n'/n \geq 1$)
- With different codes there usually is a trade-off between computational efficiency and f
- E.g. with Reed-Solomon codes, $f = 1$, but computation is inefficient.

The fountain coding principle

1. A server distributes a file consisting of n **blocks** of length l
2. Server encodes blocks into **packets**, and distributes these:
   - 2^n different packets
   - Practically infinite number of these packets can be generated
   - Number of blocks encoded into one packet is the **degree** of packet
3. In ideal case, client needs to collect any n packets in order to decode the original content

   $\rightarrow$ analogy to filling a bucket under a fountain spraying water drops

- No retransmission of a specific packet is needed
- This is the ideal case, in practice erasure codes *approximate* this.

Application examples

- Reliable multicast
  - IETF: Realiable multicast working group
  - 3GPP: Multimedia Broadcast/Multicast Service (MBMS)
- Peer-to-Peer systems
  - E.g. for solving the last block problem
- Distributed storage

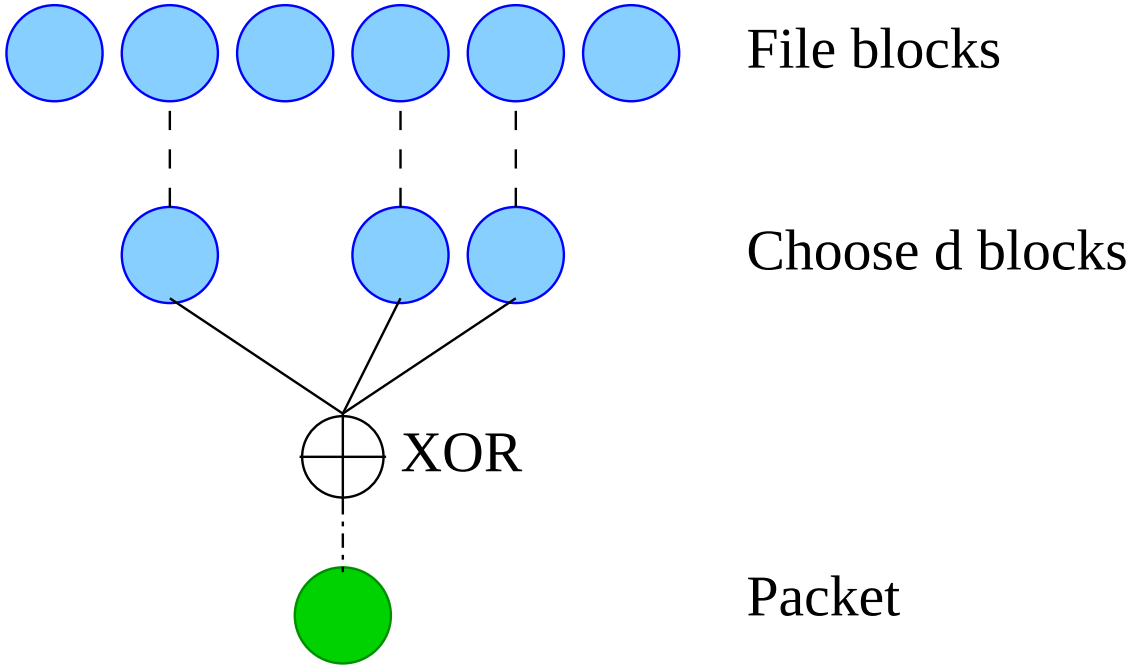
Implementation issues

- How to divide the data into blocks?
- Efficiency
  - Encoding and decoding takes time
- Overhead
  - Overhead factor $f > 1$ for efficient coding methods
  - This results in f times more data to be sent!
- Security
- Patents
  - LT / Raptor codes are patented

LT codes

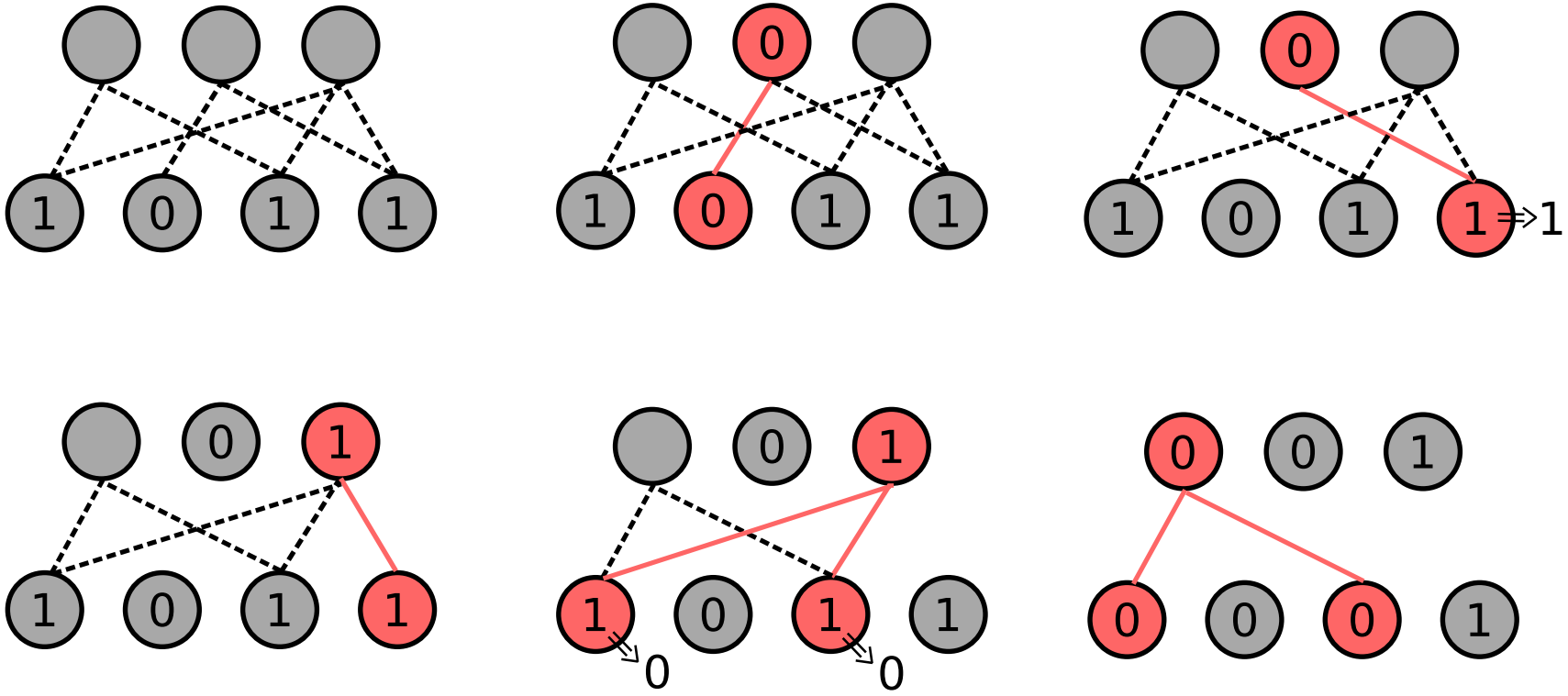
- Efficient codes developed by M. Luby, company Digital Fountain
- Optimal codes for infinite n in terms of overhead
- A **degree distribution** defines the efficiency
- Extension to LT codes with linear time encoding and decoding: **Raptor** codes
- One form of low-density parity-check (LDPC) codes

LT encoding



1. Choose packet degree d from **degree distribution** $\rho(d)$
2. Choose uniformly at random d blocks
3. XOR the selected blocks bitwise

LT decoding

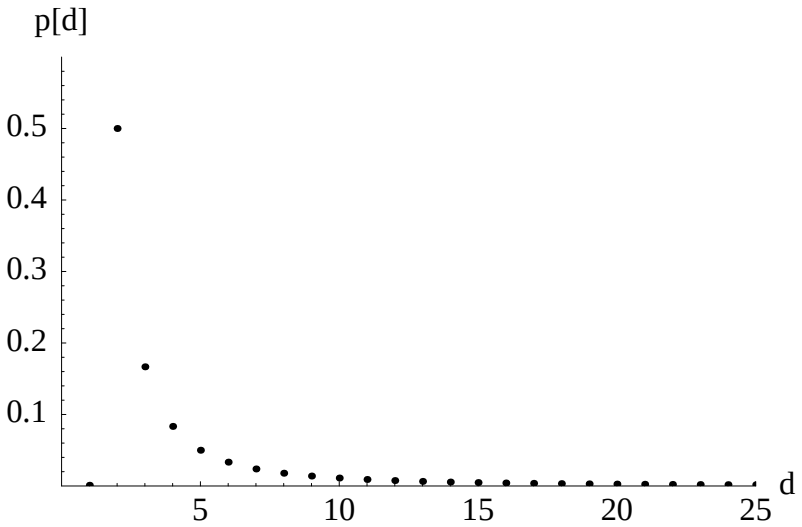


Original file blocks (input symbols) above, encoded packets below.

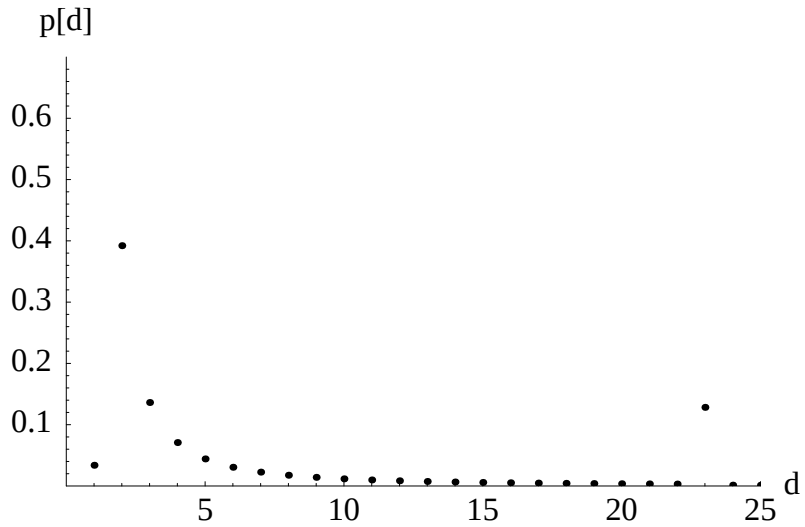
1. Degree one packet starts the decoding.
2. If the packet degree is one, then the neighbor block = packet.
3. XOR the value of the recently recovered block with its neighbors, remove edges.
4. Continue iteratively until done or no more degree one packets.

Degree distributions

- With a good degree distribution the overhead factor is as small as possible
- In expectation, ideal soliton distribution works best.
- From the ideal soliton distribution Luby derived the robust soliton distribution. It is possible to tune this distribution so that for $N \approx 10000$, the overhead is about 5%



Example of ideal soliton distribution



Example of robust soliton distribution, note the spike

Optimization of the degree distribution

- **Idea**: to estimate the average number of packets using **importance sampling**
- Review of IS:

$$E[h(\mathbf{X})] = \int h(\mathbf{x})p(\mathbf{x})\ d\mathbf{x}. = \int h(\mathbf{x})\frac{p(\mathbf{x})}{g(\mathbf{x})}g(\mathbf{x})\ d\mathbf{x},$$

where $p(\mathbf{x})$ and $g(\mathbf{x})$ are probability distributions. An estimate for expectation $\hat{h}$ can be calculated by drawing samples $\tilde{\mathbf{X}}^{(i)}$ from $g(\mathbf{x})$:

$$\hat{h} = \frac{1}{K}\sum_{i=1}^{K} h(\mathbf{X}^{(i)}) = \sum_{i} w(\tilde{\mathbf{X}}^{(i)})h(\tilde{\mathbf{X}}^{(i)}),$$

where we use the *importance ratios*

$$w(\mathbf{x}) = \frac{p(\mathbf{x})}{g(\mathbf{x})}$$

Importance sampling based optimization for LT codes

- Importance sampling allows the calculation of an expectation using a different distribution than the original one
- We propose an optimization method for LT codes using this fact, the estimate for average number of packets needed for decoding is:

$$\widehat{R}\left(\mathbf{q}\right) = \frac{1}{m}\sum_{k=1}^{m} R_k \prod_i \left(\frac{q_i}{p_i}\right)^{n_i^{(k)}},$$

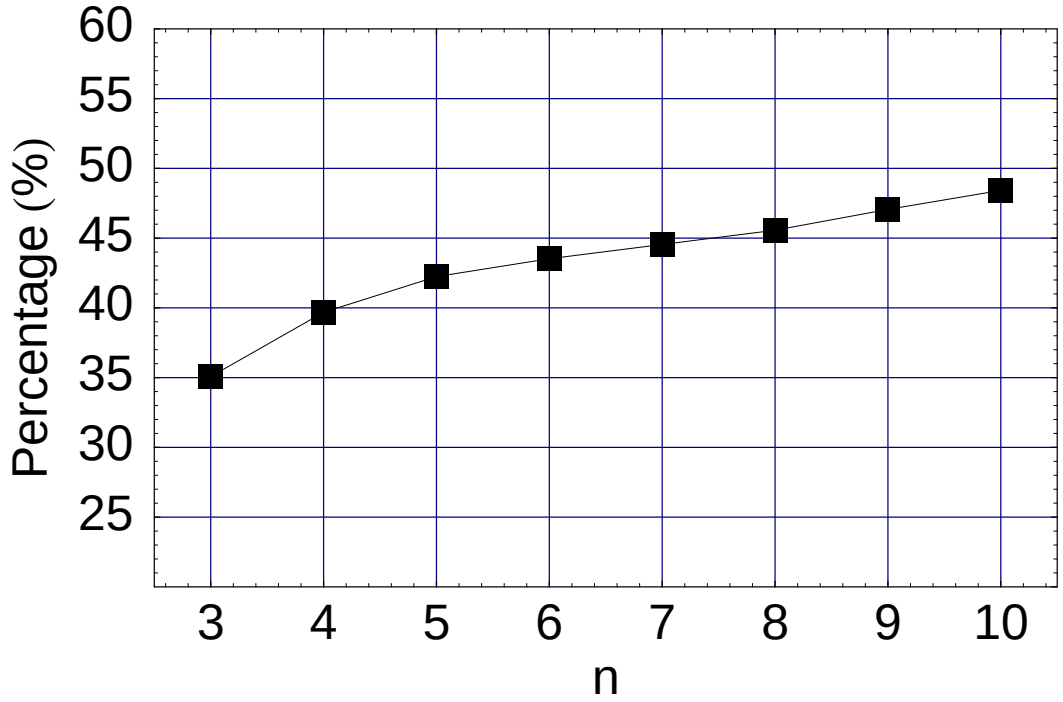
  where $n_i^{(k)}$ denotes the number of packets of degree i, $\mathbf{q}$ and $\mathbf{p}$ are degree distributions, R_k is the number of packets needed for decoding
- This means that we can generate samples of LT process with one probability distribution and use the presented equation to calculate the expectation with a different degree distribution

Importance sampling based optimization for LT codes

- We implemented an iterative algorithm, which optimizes the estimate by the method of steepest descent (i.e. gradient method)
- The accuracy of the estimate can be controlled trough the gradient, i.e., samples are generated until the variance of the gradient is small enough
- The algorithm takes some degree distribution as input and outputs a better one, if possible
- Present implementation has been tested for $n \leq 100$, in order to keep the simulation and optimization times reasonable

Results

- We optimized point distributions (i.e. parameters are the probabilities for each degree) for $n = 3, \ldots 10$.
- Plot of average overhead percentages from 100000 simulations with the optimized distributions:

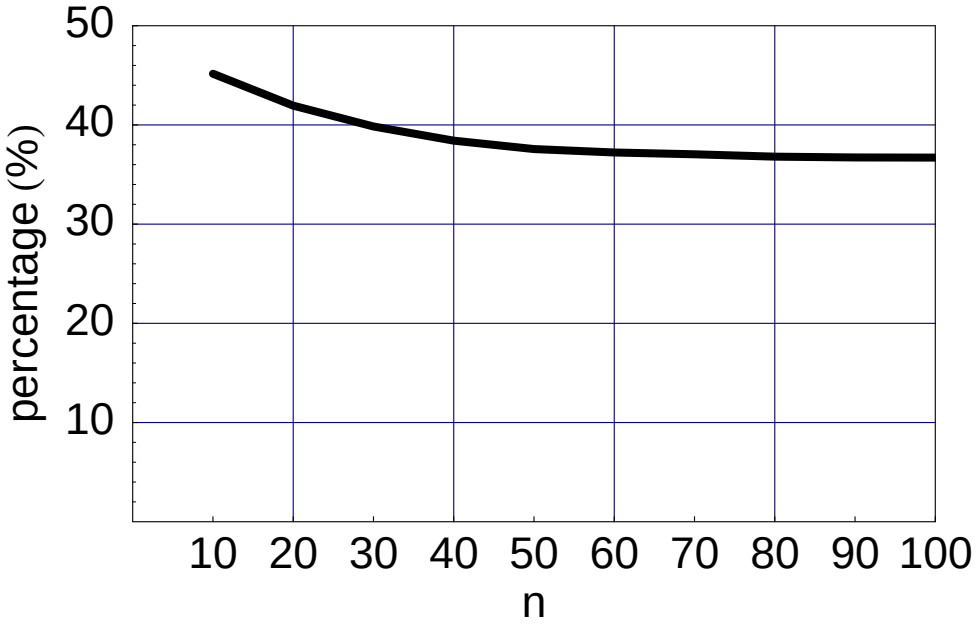


Results

- We tested geometric forms with one and three parameters, optimizing the parameters

$$p_i = e^{-\eta i} \text{ and } {}_i = e^{-\eta_1 i} + \eta_2 e^{-\eta_3 i},$$

- Overhead percentages again from 100000 simulations with optimized distributions (roughly the same for both forms!)

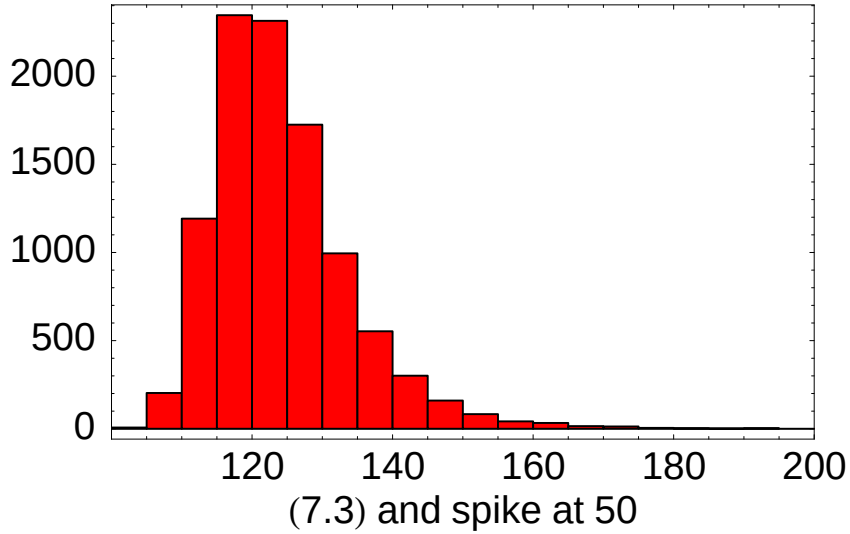


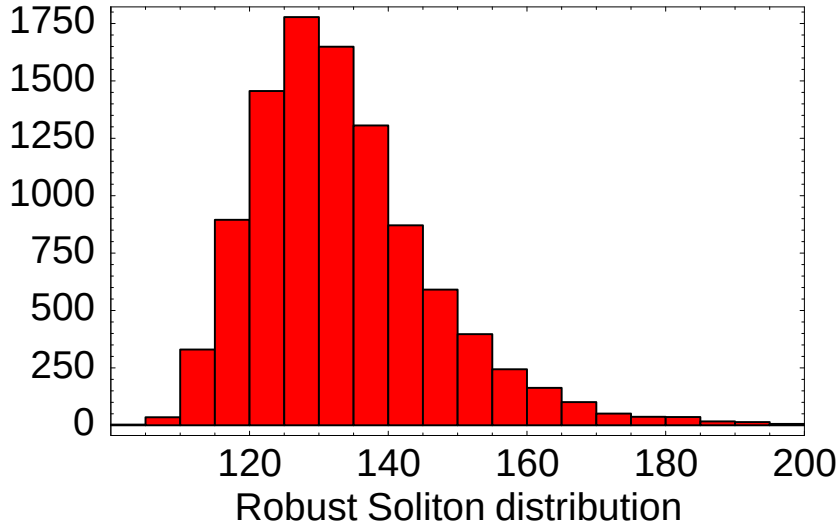
Results with $n = 100$

- We tried to create a more efficient form from soliton distributions
- Optimized parameters are the probabilities for degrees one and two, and additionally for degree 50. Otherwise the distribution is soliton distribution

Distribution	Avg	Std
Degrees one and two optimized	125	13
Degrees one and two + spike at 50	124	10
Soliton	170	70
Robust soliton	130	13

Histograms of simulations with $n = 100$





Numbers of packets needed with the best form we found and with robust soliton distribution.

Tests with $n = 1000$

- We tested the optimized distributions for $n = 100$ with simulations of the LT process when $n = 1000$

Distribution	Avg	Std
Degrees one and two optimized	1121	37
Degrees one and two + spike at 100	1130	84
Robust soliton	1124	57

- To our surprise, the optimized results for $n = 100$ perform well.

Conclusions

- Using software FEC is interesting method
- Nevertheless, applications should be carefully designed to avoid inefficiencies
- Our proposed optimization strategy for LT codes works
- We were able to generate better results than with the robust soliton distribution for $n = 100$ and even for $n = 1000$
- Open questions:
  - What is the best form of degree distribution?
  - Is the spike really needed?