FEC-Based File Transfer in Communication Networks

John N. Daigle and Nail Akar

wcdaigler@olemiss.edu, akar@ee.bilkent.edu.tr

[1]Department of Electrical Engineering
University of Mississippi

[2]Department of Electrical Engineering
Bilkent University

Overview

1. Introduction
2. Luby transform FEC
3. Luby transform performance
4. Luby transform-based file transfer
5. Raptor code-based file transfer
6. Conclusions

Objective

To explore the potential for Raptor code-based file transfer protocols in communication networks.

- Review basic ideas of Luby transform FEC
- Show examples of object transfer characteristics
- Discuss scenarios for Luby transform-based file transfer
- Review of Raptor encoding-based file transfer
- Summarize status of standards

Luby transform encoding

- Given
  - An object composed of set of k source symbols, say, $\mathcal{S} = \{s_0, s_1, \cdots, s_{k-1}\}$
  - A degree distribution, $F_{\tilde{d}}(d)$ with support $\mathcal{D} = \{1, 2, \cdots, k\}$
- For $i = 1, 2, \ldots,$ form the ith encoding symbol as follows:
  - Select d_i independently from the degree distribution, $F_{\tilde{d}}(d)$
  - Select d_i source symbols, say $\mathcal{S}_i$, at random and independently, from $\mathcal{S}$
  - Form E_i as the XOR of the elements of $\mathcal{S}_i$.
- Send each encoded symbol together with its composition to the destination
- Either
  - Send a given number of symbols, L, and try to decode, or
  - Keep forming and sending symbols until all of the source symbols are decoded

Luby transform decoding

- The set of source symbols can be written as a vector: $S = [s_0, s_1, \cdots, s_{k-1}]$.
- Each encoding symbol is a binary linear combination of the source symbols, $E_i = b_i S$, where b_i is known by the receiver.
- The set of received symbols forms a set of linear equations

$$\begin{bmatrix} E_{j_1} \\ E_{j_2} \\ \vdots \\ E_{j_n} \end{bmatrix} = \begin{bmatrix} b_{j_1} \\ b_{j_2} \\ \vdots \\ b_{j_n} \end{bmatrix} S = BS$$

- the E_{j_ℓ} and b_{j_ℓ} are known.
- Any set of k linearly independent rows of B defines S

Luby transform decoding (continued)

- Encoding symbols are generated randomly implies a random number of encoding symbols are needed to obtain k linearly independent equations.
- Received symbols containing errors are discarded
- Probabilistically, and any set of n received encoding symbols is as good as any other
- Decoding alternatives
  - Peeling
  - Gaussian elimination
- Peeling requires more symbols and less computing

Luby transform decoding (concluded)

- Sketch of decoding through on-the-fly peeling
  1. Collect received encoding symbols until a symbol of degree 1 is received
  2. Recover the source symbol S_{r_1} from E_{r_1}.
  3. Process recovered symbol
     - A For each received encoding symbols containing recovered symbol, replace received encoding symbols with XOR of symbol with recovered symbol
     - B Mark S_{r_1} processed
     - C Search reduced set for symbol of degree 1
     - D If found, go to 3
  4. Until done,
     - A Read in additional symbol
     - B Remove processed symbols from received symbol
     - C If resulting symbol, has degree 1, recover and go to 3

Luby transform performance example 1

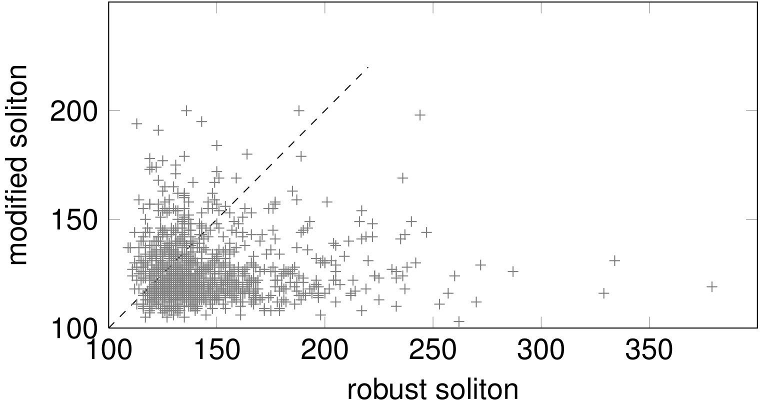


Figure: Scatter plot of the number of encoding symbols required for decoding required for two degree distributions

Luby transform performance example 2

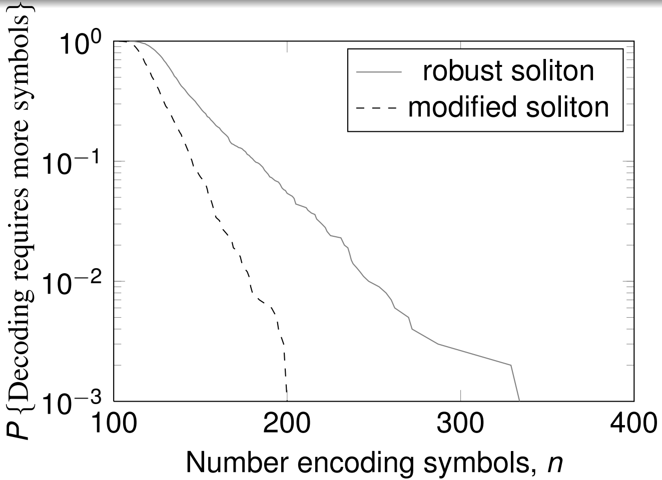


Figure: Survivor function the number of encoding symbols required

Luby transform performance example 3

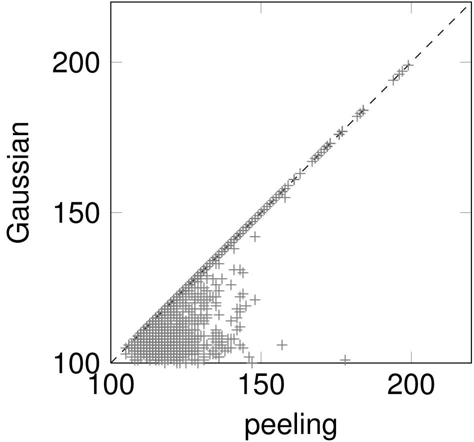


Figure: Scatter plot of the number of encoding symbols required

Summary of numerical results

- Number encoding symbols needed for decoding depends on degree distribution
  - Developing good degree distributions is very worthwhile
- Number of encoding symbols needed for decoding can be large
- Gaussian elimination requires fewer symbols, but still can require a larger number
- Not shown: overhead decreases with the number of input symbols

Luby transform-based file transfer

- The file is partitioned into k segments
- The segments are used to construct source symbols
- The source symbols are processed to form the encoding symbols
- The encoding symbols are encapsulated and sent to destination
- Decoding is done in the standard way of Luby transform decoding

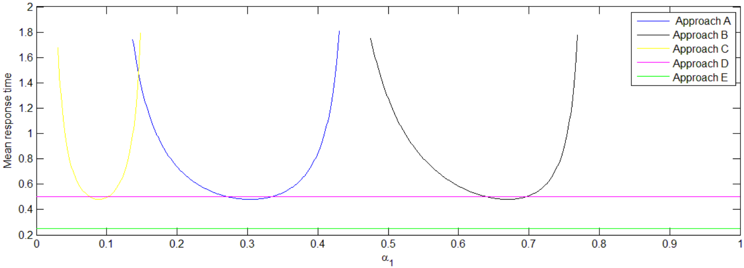
One-to-many file transfer paradigm

- Source organizes the file transfer as though transmitting to a single destination
- Each destination collects encoding symbols and decodes independently
- Packet losses to each of the destinations may be different
- Routers can drop packets without adversely affecting decoding
- New destinations can be added to the destination list at any time during file transfer
- Multicast can be used to deliver packets
- Each destination can independently report reception of file
- Transmitting server can just stop transmitting when all destination have received the file

Many-to-one file transfer paradigm

- Source file may be available at multiple servers
- Each source prepares the file for transfer in the identical way
- A server controller receives the file request and delivers requests to a selected subset of the servers
- Each server generates encoding symbols independently drawing degrees from the same distribution
- Each destination collects encoding symbols and decodes independently
- Each receiver independently reports completion of file transfer to the server controller
- Many-to-many can be implemented with each server sending to multiple possibly different destingations

Load balancing potential in many-to-one



- A Jobs are randomly directed to server i with probability α_i
- B α_1 proportion of shorter jobs directed to server 1
- C α_1 proportion of longer jobs directed to server 1
- D Jobs are assigned to each server with probability proportional to r_i.
- E Jobs are assigned to both servers using Luby encoding

- Two phase process
  - Erasure coding of input symbols to form intermediate symbols
  - Luby transform-based transfer of the intermediate symbols
- Intermediate symbols are the same size as input symbols
- Number of intermediate symbols is $n \geq k$
- Reception of any k of the n intermediate symbols may be sufficient to recover the k input symbols
  - Depends on the specifics of the erasure code
- Basically changes the problem from decoding the k specific input symbol to decoding any k out of n intermediate symbols.

Why erasure coding works

- A codeword in a linear code is linear combination of the basis vectors for a vector subspace
  - $c = mG$
  - m is a k-vector
  - G is a matrix of k n-vectors that span k-dimensional subspace of vectors of length n
  - c is a codeword
- If the columns of G are chosen such that any k columns are linearly independent (for example, Reed-Solomon codes), then if any k components of c are known, they can be used to solve for m and consequently c.
- In the general case, slightly more than k values may be needed for decoding.

Example Raptor coding in the standards

1. RFC 5052, “Forward Error Correction (FEC) Building Block,” August 2007.
    1. Describes the framework for developing an RFC for an FEC-based content delivery protocol
    2. Sets aside 256 FEC Encoding IDs, 128 each for *fully-specified* and *under-specified*
    3. Obsoletes RFC 3452, December 2002.
2. RFC 5053, “Raptor Forward Error Correction Scheme for Object Delivery,” September 2007.
    1. Describes of protocol for Raptor-based file transfer.
3. RFC 6330, “RaptorQ Forward Error Correction Scheme for Object Delivery,” August 2011.
    1. Describes RaptorQ-based file transfer.
    2. Extends the idea of Raptor coding to terabyte range files sizes

Conclusions

- Raptor codes provide a solid theoretical basis for file transfer in communication networks
- Conceptually, reliable broadcast, multicast, any cast are straightforward using Raptor codes
- RFCs are available for implementing content delivery protocols based on Raptor and RaptorQ codes
- Lots of activity in standards arena
- No manuscripts providing detailed comparisons to other content delivery protocols found
- Reports of experience using Raptor codes for content delivery were not found in searches of the net