Raptor Codes for P2P Streaming

Philipp Eittenberger\textsuperscript{1}, Todor Mladenov\textsuperscript{2}, Udo Krieger\textsuperscript{1}

\textsuperscript{1}Faculty of Information Systems and Applied Computer Science
Otto-Friedrich University Bamberg, Germany

\textsuperscript{2}Department of Information and Communications, Gwangju Institute of Science and Technology, Gwangju, South Korea

PDP’12, 15. February 2012
1. Introduction

2. Fountain Codes

3. Evaluation
P2P Video Streaming

Chunk Scheduling

\[ p_i = \text{Peer}_i \]

\[ j = \text{Chunk}_j \]

\[ n = \text{Upload Capacity} \]
Assumptions

- Global knowledge of bandwidth distribution is available.
- Last mile is always the bandwidth bottleneck.
\[ p_i = Peer_i \]
\[ j = \text{Chunk}_j \]
\[ n = \text{Upload Capacity} \]
Optimal scheduling algorithm is a NP-complete problem.
P2P Video Streaming
Fountain Code Enabled P2P Streaming

Proposed by Wu and Li [WuLi, 2005]

Benefit
Order of the pieces not important.
No content reconciliation needed.
P2P Video Streaming
Fountain Code Enabled P2P Streaming

\[ p_i = Peer_i \]
\[ j = Chunk_j \]
\[ n = Upload Capacity \]

Proposed by Wu and Li [WuLi, 2005]

**Benefit**

Order of the pieces not important.

- No content reconciliation needed.
Fountain Codes

History

- Many implementations are based on variations of low-density parity-check (LDPC) codes (introduced by Gallager [Gallager, 1963])
- First construction of a efficient fountain code in 1998 (by Michael Luby; published 2002 [Luby, 2002])
- Raptor codes are an improvement over Luby transform codes (invented in late 2000 by Amin Shokrollahi [Shokrollahi, 2006])
- Raptor codes are the first practical class of a fountain code with near optimal error correction functionality
For a given vector \((x_1, \ldots, x_k)\) of source symbols, a fountain encoder produces a potentially limitless stream of encoded symbols \(y_1, y_2, \ldots\).

A symbol refers to a bit or a sequence of bits.

Fountain codes are governed by a probability distribution \(\mathcal{D}\) on the vector space \(\mathbb{F}_2^k\).
**Encoding procedure for generating encoded symbol** $y_i$: 

1. Sample $D$ to obtain a vector $(a_1, ..., a_k) \in \mathbb{F}_2^k$.
2. Calculate $y_j = \sum_i a_i x_i$.

- To decode the data, the receiver needs to be able to determine the corresponding vector $(a_1, ..., a_k)$.
- Therefore, an unique ID for might be used to serve as the seed for a PRNG.
Belief-Propagation Decoding

Repeats the following steps until failure in step 1 or the decoder stops successfully in step 4:

1. Find encoded symbol $y_i$ of degree 1. If there is no encoded symbol of degree 1, decoding fails.
2. Decode $x_j = y_i$.
3. Let $i_1, \ldots, i_l$ denote the indices of encoded symbols connected to $x_j$.
   Set $y_{i_s} = y_{i_s} + x_j$ for $s = 1, \ldots, l$, and remove $x_j$ and all edges emanating from it.
4. Goto step 1, if there are unrecovered source symbols, else stop.
Belief-Propagation Decoding Problems

- There might not be any encoded symbols of degree one at some intermediate step of the decoding.
- Too many encoded symbols of degree one at some intermediate step, leading to many redundant encoded symbols and thus, to a large overhead.
- With high probability there is a fraction of source symbols that do not contribute to the values of any of the encoded symbols.
- These source symbols can never be recovered.
**Raptor Codes**

- Idea: Use a high rate code to precode the source symbols (= *intermediate symbols*).
- A LT code is applied to the intermediate symbols. There is still a small fraction of intermediate symbols, which can not be recovered.
- But they can be recovered by using an appropriate erasure decoding algorithm.
Raptor Codes

- Achieve linear time encoding and decoding performance.
- Have better overhead-failure curves than LT codes in practice.
- Decoding performance extremely close to the Shannon bounds.
- Linear block code
- Can be represented by its generator matrices.
Raptor Code R10

- Designed for encoding and decoding speed and a reasonable overhead-failure curve.
- Already adopted into a large number of different standards, e.g.:
  - 3GPP Multimedia Broadcast Multicast Service
  - IETF RFC 5053
  - DVB-IPTV
  - ...
- Source blocks of up to 8,192 source symbols and up to 65,536 encoded symbols.
Raptor Code R10: Implementation

\[ e_{N \times 1} = G_{LT(1..N)} A_{L \times L}^{-1} d'_{L \times 1} = G_{LT(N..1)} c_{L \times 1} \]

\[ t'_{K' \times 1} = G_{LT(1..K')} A_{M \times L}^{-1(T)} e'_{M \times 1} = G_{LT(1..K')} c_{L \times 1} \]

[Mladenov et al., 2011]
Outline

1. Introduction
2. Fountain Codes
3. Evaluation
Data Encoding & Encapsulation

Parameters to determine:
- Block size & symbol size.
- Number of repair symbols $\varepsilon$ needed for a successful decoding (the overhead rate of the code).

Video Data Generation → Video Data Encoding → Transmission
Data Encoding & Encapsulation

Parameters to determine

- Block size & symbol size.
- Number of repair symbols $\varepsilon$ needed for a successful decoding (= the overhead rate of the code).
Raptor implementation in plain Ansi-C (no assembly, no GPU instructions).

One CPU Thread (i7 2.8 GHz) for all encoding/decoding throughput measurements.

No corruption of encoded symbol (∼ handled by lower layers).

Unique range of encoding IDs per peer.

In total more than 1,000,000,000 tests were performed.
**Introduction**

Fountain Codes

**Evaluation**

**Encoding Throughput**

![3D Graph]

Throughput (in Mbit/s)

Symbol Size (in Byte)

Block Size (in Symbols)

Figure: Encoding Throughput
Decoding Throughput

Throughput (in Mbit/s)
Symbol Size (in Bytes)
Block Size (in Symbols)
Decoding Success

- $P(\text{Decoding Success})$ vs Number of Repair Symbols

- Graphs show the relationship between $P(\text{Decoding Success})$ and the number of repair symbols for different values of $k$.

- $k = 10$, $k = 100$, $k = 1000$, and $k = 2000$.

- Each graph represents a different $k$ value, with the x-axis indicating the number of repair symbols and the y-axis showing the probability of decoding success.

- The graphs demonstrate how the probability of decoding success increases with the number of repair symbols for each $k$ value.

- The plots illustrate the effectiveness of fountain codes in improving decoding success as the number of repair symbols increases.
Minimum amount of repair symbols necessary to achieve a decoding success of 99.9 %:

<table>
<thead>
<tr>
<th>k</th>
<th>10</th>
<th>32</th>
<th>64</th>
<th>100</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td># repair symbols</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>ε</td>
<td>110%</td>
<td>34.37 %</td>
<td>17.18 %</td>
<td>11 %</td>
<td>9.37 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>256</th>
<th>512</th>
<th>1000</th>
<th>1024</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td># repair symbols</td>
<td>13</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>ε</td>
<td>5.07%</td>
<td>2.92 %</td>
<td>1.6 %</td>
<td>1.56 %</td>
<td>0.85 %</td>
</tr>
</tbody>
</table>
Overhead Rate

Figure: Overhead Rate
Conclusion

Are Raptor Codes Suitable for P2P Streaming?

Necessary Prerequisites
- Necessary throughput rates can be achieved ✓
- Overhead rate is negligible for larger block sizes ✓
- Robustness (?)

Cons
- Introduce extra delay
- Additional computational complexity

Pros
- No content reconciliation (avoid chunk scheduling)
- Better/Easier utilization of “slow“ peers
- Lean On/Off-Push protocol (less protocol overhead)
Questions?

Philipp Eittenberger
philipp.eittenberger@uni-bamberg.de
Low-Density Parity-Check Codes, 1963.

[Luby, 2002] Michael Luby,

[Shokrollahi, 2006] Amin Shokrollahi,

[WuLi, 2005] C. Wu and B. Li,
[Mladenov et al., 2011]
T. Mladenov and S. Nooshabadi and K. Kim,