Peer-to-Peer Networks

12 Fast Download, Part II

Arne Vater
Technical Faculty
Computer Networks and Telematics
University of Freiburg

Forward Error Correction

- uses plain blocks for distribution
- plus *k* linearly independent code blocks
  - Reed-Solomon code
  - proposed in *"Network coding for large scale content distribution“,* [2005]

Forward Error Correction

- FEC(k) has read/write cost of O(min{$k \cdot n$, n^2})
  - example decoding matrix with 8 blocks and 3 FEC blocks:

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Forward Error Correction

- bring all plain blocks to the right

$$
\begin{pmatrix}
\alpha_1 & \alpha_4 & \alpha_6 & \alpha_7 & \alpha_5 & \alpha_2 & \alpha_3 & \alpha_8 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\beta_1 & \beta_4 & \beta_6 & \beta_7 & \beta_5 & \beta_2 & \beta_3 & \beta_8 \\
\gamma_1 & \gamma_4 & \gamma_6 & \gamma_7 & \gamma_5 & \gamma_2 & \gamma_3 & \gamma_8 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
$$

Forward Error Correction

- bring all code blocks to the top

$$\begin{pmatrix} \alpha_1 & \alpha_4 & \alpha_6 & \alpha_7 & \alpha_5 & \alpha_2 & \alpha_3 & \alpha_8 \\ \beta_1 & \beta_4 & \beta_6 & \beta_7 & \beta_5 & \beta_2 & \beta_3 & \beta_8 \\ \gamma_1 & \gamma_4 & \gamma_6 & \gamma_7 & \gamma_5 & \gamma_2 & \gamma_3 & \gamma_8 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Forward Error Correction

- remove all columns and rows with uncoded blocks
  - requires O($k \cdot (n - k)$) read/write accesses
- and decode the remaining code blocks

$$\begin{pmatrix} \alpha_1 & \alpha_4 & \alpha_6 \\ \beta_1 & \beta_4 & \beta_6 \\ \gamma_1 & \gamma_4 & \gamma_6 \end{pmatrix}^{-1} \times \begin{pmatrix} b_1 \\ b_4 \\ b_5 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_4 \\ x_6 \end{pmatrix}$$

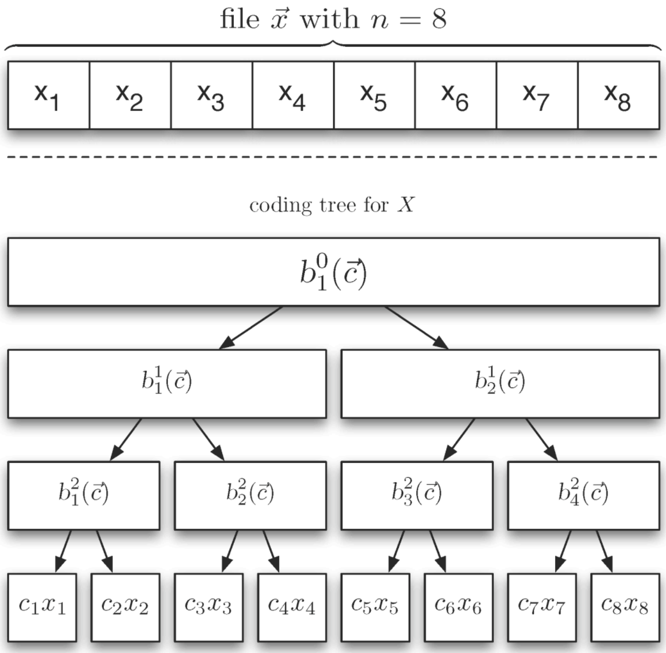
- this adds O(k · n) read/write accesses

Forward Error Correction

- FEC(0) equals BitTorrent
- performance hierarchy
  - FEC(k + 1) > FEC(k)
- FEC(k) < Network Coding

Treecoding

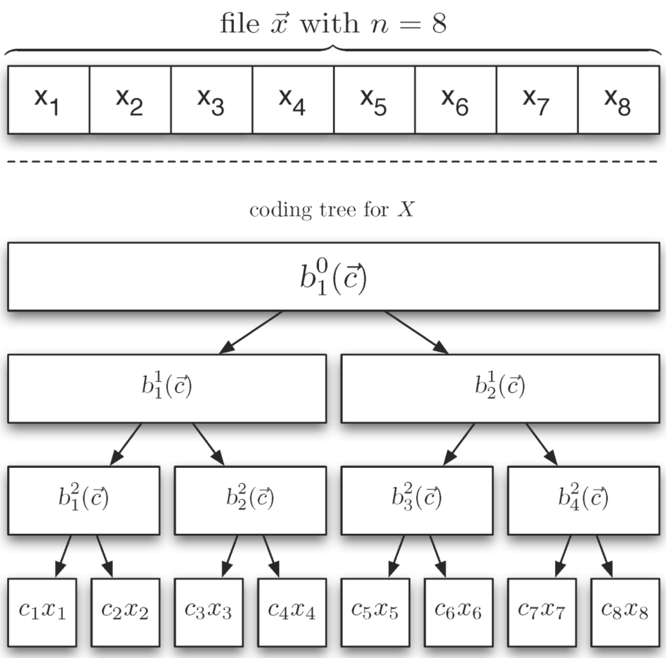
- SPAA 2009, SPAA 2010
- tree structure
  - fixed linear coefficients for all blocks x_i
  - Xor of two nodes creates parent node



$$
\begin{aligned}
b_i^{\log n}(c) &= c_i x_i && \text{for } i \in \{1, \ldots, n\} \\
b_i^{j-1}(c) &= b_{2i-1}^j(c) + b_{2i}^j(c) && \text{for } j \in \{1, \ldots, \log n\}, \\
& && \quad i \in \{1, \ldots, 2^{j-1}\}
\end{aligned}
$$

Treecoding

- *k* different trees
  - with linearly independent linear coefficients
- root nodes are equivalent to network coding blocks
- leaves are equivalent to uncoded blocks
- any code block can be decoded by Xor from
  - either its two children blocks
  - or its parent block and its sibling block
  - requires constant read/write complexity



Treecoding

- Downloading from one tree
  - start with root block
  - continue with any child
    - and decode the other one by Xor
- Downloading from several trees
  - parallel download as from one tree
  - if in any subtree with *m* nodes there are *m* blocks available in all downloading trees
    - and Xor decoding is not possible
    - then use network coding to decode that subtree

Treecoding

- Read/Write Complexity (average)
  - $O(n)$ for $k = 1$
  - $O(\min\{ kn \cdot \log^2 n, n^2\})$ for any k

- Performance hierarchy
  - Treecoding(k + 1) > Treecoding(k)
- Treecoding(k) ≥ FEC(k)

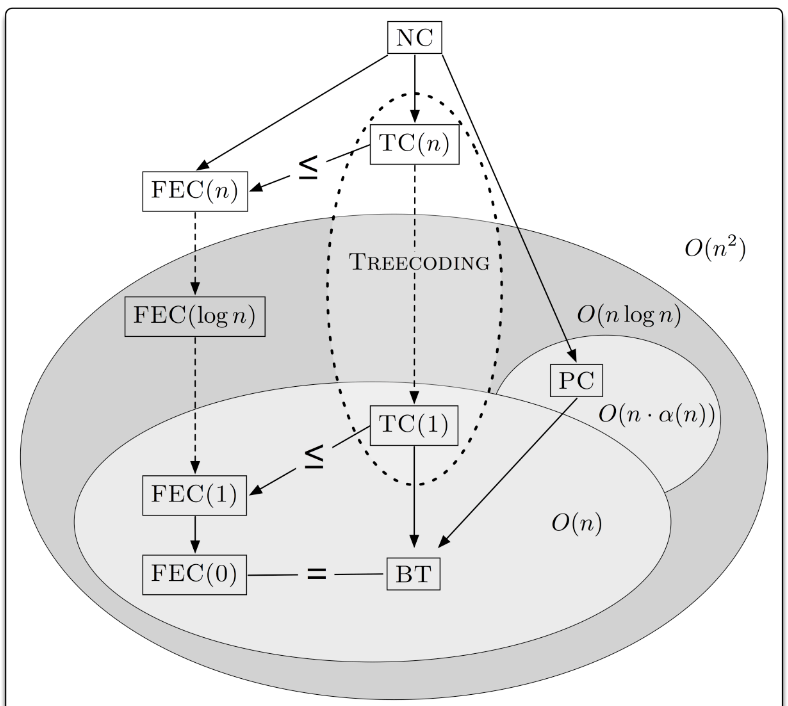
Comparison

- R/W Cost (average)

BitTorrent	Paircoding	FEC(k)	Treecoding	Network Coding
O(n)	O($n \cdot \alpha(n)$)	O($k \cdot n$)	O($kn \cdot \log^2 n$)	O(n^2)

$\alpha(n)$ is the inverse Ackerman function

- Performance



Peer-to-Peer Networks

12 Fast Download, Part II

Arne Vater

Technical Faculty

Computer Networks and Telematics

University of Freiburg