A new deterministic source coding method in peer-to-peer systems

Attila Balaton Tamás Lukovszki Ádám Agócs
Eötvös Loránd University
Faculty of Informatics
Budapest, Hungary
{balcsi4,lukovszki,adoszka}@inf.elte.hu

***Abstract*—We propose a novel deterministic method for source coding in peer-to-peer networks. The main advantage of our method compared to randomized methods is that the coding will be invertible with probability one. A further advantage is a much lower communication overhead due to special coding vectors. We apply the coding algorithm in a scalable and robust peer-to-peer system. We analyze the deterministic method theoretically and prove upper and lower bounds for the number coded pieces. The theoretical results are backed up by simulations.**

I. Introduction

Network coding has been attracted a huge amount of theoretical and practical research since its first appearance in [3]. Network coding is a technique, where the nodes of a network can take several packets and combine them together for transmission, instead of simply relaying the packets they receive. Consider the network as a directed graph $G = (V, E)$ with capacities on the edges. Ahlswede et al. [3] showed that for a source node to a set of destination nodes, the rate of the multicast can achieve the cut bound (the minimum of the cuts separating the source from a destination) by using network coding. In general, this is not possible, if the data is regarded as "fluid", which can simply be routed or replicated. Li et al. [12] proved that linear coding is enough to achieve the upper bound in multicast problems.

In this work we deal with a special case of network coding, where just the server is allowed to combine the pieces of the file, which has to be distributed in the network. This is a form of source coding. Locher et al. [13] has shown that in peer-to-peer systems, such as BitTorrent [5], this kind of source coding is very useful to increase the diversity of pieces in the network. This accelerates tit-for-tat exchanges between the peers and it is a good solution to the problem of rare pieces, because every node is able to restore the original file from any d linearly independently coded pieces, where d is the number of the original pieces of the file. In [13] a special random linear coding scheme is used. Random linear network coding [11], [8], [6] is a powerful and popular method for disseminating information in networks. In this scheme the coding coefficients are chosen randomly from a finite field. In this method it is possible that the coding will not be invertible. If the coding vectors are linearly dependent, the original content is not restorable from the coded blocks. Jaggi et al. [9] have shown that if the field size is at least $|E|/\delta$, the encoding will be invertible with probability $1 - \delta$. Locher et al. [13] use random 0-1 coding vectors in their source coding scheme for increasing the diversity of pieces. We introduce a deterministic method, which guarantees that, with probability 1, the original file can be restored after downloading of any d different coded pieces. A further advantage of our deterministic methode is that each coding vector can be represented by a single coefficient. This results in a significantly lower communication overhead than random methods.

A. Our results

We propose a novel deterministic source coding method for increasing the diversity of pieces of a file in peer-to-peer networks. Due to the increased piece diversity, tit-for-that piece exchange can be accelerated and the problem of rare pieces can be ignored in BitTorrent like peer-to-peer systems. Our deterministic coding guarantees that the original file can be restored from the coded blocks with probability one. We use special coding vectors, that result in a coding matrix which corresponds to a Vandermonde matrix over a finite field. Another advantage of our method is a significant reduction of the communication overhead. For the decoding of the blocks, the coding vectors also must be known at the receivers. They are necessary to compute the inverse of the coding matrix. Therefore, these vectors also must be transmitted, which causes communication overhead. Our deterministic coding method uses special coding vectors, that allow to compute a coding vector from only one coefficient. It means that instead of the whole coding vector, only one element of the finite field must be transmitted with each coded piece.

B. Outline of the paper

In Section II we describe the network model and the deterministic source coding algorithm and prove that any receiver can decode the file after receiving d different coded pieces, where d is the number of original pieces of the file. In Section III we compare our method with the random coding methods regarding the traffic overhead. In Section IV we study the problem, when the client can not be distinguish between two coded pieces before downloading them. In dependence

on the number of different coded pieces in generated by the server we prove upper and lower bounds on the number of coded pieces that must be download for the decoding the file. In Section V we provide simulation results. Section VI summarizes our results.

II. Deterministic source coding

Our deterministic source coding method can be applied to any peer-to-peer system which has a server or there is at least one node which owns the original file at the beginning of the content distribution and the file is divided into pieces. The server generates combinations of the original pieces by applying our coding method. The other nodes in the network download the coded pieces and upload the pieces they received to other nodes. These nodes restore the original file at the time they have enough coded pieces. A frequently used p2p system is BitTorrent. In this system the seeders play role of the server. The first seeder is the client which usually publish the .torrent file. When a new node joins the network it receives a set of neighbors from the tracker and becomes a leecher. The leechers download the coded pieces and restore the original file. Every leecher which restored the original file can leave the network or it becomes a seeder and generates new coded pieces, in order to increase the piece diversity.

A. Coding algorithm

Now we describe our deterministic source coding method, which guaranties that the encoding will be invertible at any receiver with the probability one. Furthermore, compared to random coding methods, the communication overhead can be reduced in the network, due to the special coding vectors.

Suppose the file is split into d pieces. The coding algorithm generates $K > d$ coded pieces. Every piece of the file is represented as an m-dimensional vector: $x_1, x_2, \dots x_d \in GF(p)^m$, where $p \geq K$ is a prime number and $GF(p)$ is a finite field of order p. Let $a_1, a_2, \dots, a_K \in GF(p)$ where $a_i \neq a_j$ if $i \neq j$.These numbers will be called the *coding coefficients*.

The coding algorithm generates K different linear combination from the original vectors:

$$y_i = 1x_1 + a_i x_2 + a_i^2 x_3 + \ldots + a_i^{d-1} x_d$$

where $1 \leq i \leq K$. We show that any d of these K vectors will be independent. Another huge advantage of this method compared to the randomized approach is that we do not have to send the full coding vector with the combination. The coding coefficient a_i will be enough, the other coefficients are the powers of them and can be computed easily at the receiver.

Theorem 2.1: We can restore the original file from arbitrary d different y_i vectors.

Proof: 2.1 The proof based on the following lemma:

Lemma 2.2: Let $a_{i_1}, a_{i_2}, \dots, a_{i_d} \in GF(p)$ where $0 < a_{i_1} < a_{i_2} < \ldots < a_{i_d}$. The following matrix is not singular:

$$A = \begin{pmatrix} 1 & a_{i_1} & a_{i_1}^2 & \dots & a_{i_1}^{d-1} \\ 1 & a_{i_2} & a_{i_2}^2 & \dots & a_{i_3}^{d-1} \\ 1 & a_{i_3} & a_{i_3}^2 & \dots & a_{i_3}^{d-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & a_{i_d} & a_{i_d}^2 & \dots & a_{i_d}^{d-1} \end{pmatrix}$$

Proof: The matrix A is a Vandermonde matrix, its determinant is the following:

$$det(A) = \prod_{j<\ell} (a_{i_\ell} - a_{i_j}) \tag{1}$$

Since every a_{i_j} is different, the determinant of A is not zero, hence A is not singular. ■

Assume that we get d different linear combinations denoted by $y_{i(1)}, y_{i(2)}, \dots, y_{i(d)}$, where

$$y_{i(k)} = 1x_1 + a_{i(k)} x_2 + a_{i(k)}^2 x_3 + \ldots + a_{i(k)}^{d-1} x_d$$

for every $1 \leq k \leq d$, which gives us d different vector equation. Now consider the first coordinates of these equations. This is a linear equation system with the following coefficients:

$$\begin{aligned} y_{i(1)_1} &= 1x_{11} + a_{i(1)} x_{21} + a_{i(1)}^2 x_{31} + \ldots + a_{i(1)}^{d-1} x_{d1} \\ y_{i(2)_1} &= 1x_{11} + a_{i(2)} x_{21} + a_{i(2)}^2 x_{31} + \ldots + a_{i(2)}^{d-1} x_{d1} \\ &\vdots \\ y_{i(d)_1} &= 1x_{11} + a_{i(d)} x_{21} + a_{i(d)}^2 x_{31} + \ldots + a_{i(d)}^{d-1} x_{d1} \end{aligned}$$

which can be written in the following form:

$$\begin{pmatrix} y_{i(1)_1} \\ y_{i(2)_1} \\ y_{i(3)_1} \\ \vdots \\ y_{i(d)_1} \end{pmatrix} = \begin{pmatrix} 1 & a_{i_1} & a_{i_1}^2 & \dots & a_{i_1}^{d-1} \\ 1 & a_{i_2} & a_{i_2}^2 & \dots & a_{i_3}^{d-1} \\ 1 & a_{i_3} & a_{i_3}^2 & \dots & a_{i_3}^{d-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & a_{i_d} & a_{i_d}^2 & \dots & a_{i_d}^{d-1} \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \\ \vdots \\ x_{d1} \end{pmatrix}$$

According to the previous lemma the matrix is invertible. Hence the solution exists and unique. Thus, the first coordinate of every piece is decoded. Applying this method for the remaining $m-1$ coordinates we can decode every piece. ■

Now compare this method with the random techniques. If the encoding coefficients are chosen uniformly at random, the probability that the coding matrix will be invertible is depends on the size of the field. Jaggi et al. [9] showed that if the network is modeled as a directed graph $G = (V, E)$ and the field size is at least $|E|/\delta$, the encoding will be invertible at any given receiver with probability at least $1 - \delta$. This means that a large network requires a large field size.

Our new deterministic source coding method guarantees that d different coded pieces are enough for the decoding, where d is the number of the pieces of the original file. To restore the original file, we need the coding vectors to compute the inverse of the coding matrix. These vectors must be sent for every coded piece, which increase the traffic in the network. Using deterministic source coding, this additional data is just one coefficient for each coded piece, which is significantly less than in case of random coding, where the whole coding vector of d coefficients must be sent. In the next section we analyze traffic overhead resulting from this fact.

III. Traffic overhead

As it mentioned before the deterministic method has another huge advantage compared with random coding regarding the communication overhead. To restore the original file the coding vectors must be known. These vectors must be sent with the coded pieces. This results in traffic overhead.

When the coefficients are arbitrary, the whole vector must be transmitted for the decoding. This means d element of $GF(p)$. If deterministic source coding is used, sending a_i is enough, the other coefficients are the powers of a_i and they can be computed at the receiver. The more pieces the file contains, the more gain against the random method is achieved. Table I and Table II contain results, when p is a 32-bit prime number and 64-bit prime number, respectively. The amount of transmitted data by using random source coding corresponds 100%, the percentages in the tables belongs to the deterministic method.

file size	64K pieces	256K pieces	1M pieces	4M pieces
1M	99.91%	99.99%	100%	-
20M	98.09%	99.88%	99.99%	100%
100M	91.11%	99.40%	99.96%	100%
700M (CD)	59.44%	95.88%	99.73%	99.98%
4,37 GB (DVD)	18.62%	78.99%	98.31%	99.89%

TABLE I
THE AMOUNT OF TRANSMITTED DATA BY DETERMINISTIC CODING COMPARED TO RANDOM CODING, WHEN p IS A 32-BIT PRIME NUMBER

file size	64K pieces	256K pieces	1M pieces	4M pieces
1M	99.81%	99.97%	100%	-
20M	96.24%	99.76%	99.98%	100%
100M	83.67%	98.79%	99.92%	100%
700M (CD)	59.44%	92.11%	99.46%	99.97%
4,37 GB (DVD)	10.28%	64.62%	96.68%	99.79%

TABLE II
THE AMOUNT OF TRANSMITTED DATA BY DETERMINISTIC CODING COMPARED TO RANDOM CODING, WHEN p IS A 64-BIT PRIME NUMBER

We remark that the communication overhead of random coding can be reduced by using special coding vectors. In [13] the random coding vectors contain 0-1 coordinates. This means that the coding vectors are bitmaps of size d bits. Since usually $d > 64$, our method generate lower overhead. If $d \leq 64$, the overhead is negligible in both methods.

To restore the original file a peer needs d different coded pieces in case of our deterministic coding and d linearly independently coded pieces in case of random coding. This means that a peer have to know which coded pieces are available at its neighbors. This can lead to a significant communication overhead. For example, consider the BitTorrent protocol. In this protocol the neighbors of a newly joining peer send them a `bitfield` message immediately after the handshaking, before any other messages are sent. The bits of the bitfield represent the pieces available at the peer. After that, when a peer successfully downloaded and verified a piece, it sends the index of this piece to its neighbors in a `have` message. Compare the size of this message in three cases: without coding, using random coding and using our deterministic coding. If the BitTorrent network does not use any coding, each of the original pieces of the file can be identified by a bit in the `bitfield` message and by an index in a `have` message. In the case of random coding, a coded piece can be identified by the coding vector which is d dimensional and usually every coordinates come from $GF(p)$. Even if we only use 0-1 coefficients, we need d bits for one coded piece. By using our deterministic source coding, every coded piece can be identified by the second coordinate of the coding vector which is an element of $GF(p)$. Therefore, our deterministic coding can easily be made compatible with the standard BitTorrent protocol by sending this coordinate for a newly downloaded coded piece to the neighbors in a `have` message.

IV. Upper and lower bounds on the number of downloaded pieces

In this section we show that even the exchange of the identifier of the piece is unnecessary if the diversity of pieces is large enough. We consider the case when the client does not have information about the pieces its neighbors. A node download d arbitrary pieces and check if these pieces are different after all d pieces are downloaded. The peer is able to restore the original file from any d different pieces. Hence the main question is the number of coded pieces must be downloaded to get d different coded pieces.

We denote by K the number of different coded pieces. In our analysis we assume, that the coded pieces are distributed uniformly in the network, i.e. that the probability of choosing a certain coded piece uniformly at random is $1/K$. The first theorem states that if $K = \omega(d^2)$ it is enough to download d pieces. Then the probability that these will be different is asymptotically 1 if $d \to \infty$.

Theorem 4.1:
(i) Let $K = \omega(d^2)$ and download exactly d coded pieces. The probability that the original file can be restored from these pieces is asymptotically 1 when $d \to \infty$.
(ii) If $K = o(d^2)$ then this probability is asymptotically 0 when $d \to \infty$.

Proof: We choose d coded pieces among K, independently, uniformly at random. Denote $P(d)$ the probability that we choose d different pieces:

$$P(d) = \frac{\binom{K}{d} d!}{K^d} \tag{2}$$

In (2) the numerator $\binom{K}{d} d!$ is the number of favorable cases, i.e. choosing d different coded pieces among K. The denominator is the number of all cases.

$$P(d) = \frac{\binom{K}{d} d!}{K^d} = \frac{\frac{K!}{(K-d)!}}{K^d} = \frac{K(K-1)\dots(K-d+1)}{K^d}$$

For an upper bound on $P(d)$, consider the inequality between the arithmetic and geometrical means for $K, (K-1), \dots, (K-d+1)$:

$$\sqrt[d]{\prod_{i=0}^{d-1}(K-i)} < \frac{\sum_{i=0}^{d-1}(K-i)}{d} = K - \frac{d-1}{2} \tag{3}$$

Using (3) we got the following upper bound for $P(d)$:

$$P(d) < \frac{(K - \frac{d-1}{2})^d}{K^d} = \left(1 - \frac{d-1}{2K}\right)^d \tag{4}$$

which is asymptotically $e^{-\frac{(d-1)d}{2K}}$.

Now we show a lower bound on $P(d)$. Instead of the the inequality between the arithmetic and geometrical means, underestimate the product with the lowest member. Thus we got

$$P(d) = \frac{\prod_{i=0}^{d-1}(K-i)}{K^d} > \frac{(K-d+1)^d}{K^d} = \left(1 - \frac{d-1}{K}\right)^d \tag{5}$$

which is asymptotically $e^{-\frac{(d-1)d}{K}}$.

Summarizing, we have

$$e^{-\frac{(d-1)d}{K}} < P(d) < e^{-\frac{(d-1)d}{2K}} \tag{6}$$

Now using the $K = \omega(d^2)$ and the lower bound on $P(d)$ the proof of (i) is done:

$$\lim_{d\to\infty} P(d) \geq \lim_{d\to\infty} e^{-\frac{(d-1)d}{\omega(d^2)}} = \lim e^{-o(1)} = 1 \tag{7}$$

The proof of (ii) is obtained by the upper bound $P(d) < e^{-\frac{(d-1)d}{2K}}$. Then the right side of the inequality goes to 0 if $K = o(d^2)$. ∎

Theorem 4.1 (i) states that downloading arbitrary d pieces are enough if $K = \omega(d^2)$. Theorem 4.1 (ii) says that if $K = o(d^2)$ then d pieces will not be enough with probability which goes to 1 if d increases. In fact a stronger claim also can be proved. If $K = o(d^2)$ then slightly more pieces than d still not enough. If the number of downloaded coded pieces is $d+s$, where s is a constant which is independent from d, then the probability that the decoding is possible still goes to 0 if $d \to \infty$.

Theorem 4.2: Let $K = o(d^2)$ and assume that we download $d+s$ coded pieces, where s is a given constant. The probability of the original file can be decoded from these $d+s$ pieces is asymptotically 0.

Proof: We will show that for every $0 \leq i \leq s$ the probability of there are $d+i$ different among the downloaded $d+s$ goes to 0. Denote this probability by $P(d+i)$ which is overestimated in the following way:

$$P(d+i) < \frac{\binom{K}{d+i}\frac{(d+s)!}{(s-i)!}(d+i)^{s-i}}{K^{d+s}} \tag{8}$$

For the estimation (8), we choose the $d+i$ different coded pieces, then consider the first occurrences and distribute the remaining $s-i$ coded pieces. This is obviously an upper bound because some cases counted several times, if a piece arrives repeatedly it is counted only once. Continuing the estimation:

$$\frac{\binom{K}{d+i}\frac{(d+s)!}{(s-i)!}(d+i)^{s-i}}{K^{d+s}} = \frac{\prod_{j=0}^{d+i-1}(K-j)(d+s)!(d+i)^{s-i}}{K^{d+i}(d+i)!K^{s-i}(s-i)!}$$

Split the product into two pieces and overestimate both of them. The first part can be overestimated by the following way:

$$\frac{\prod_{j=0}^{d+i-1}(K-j)}{K^{d+i}} < \frac{(K - \frac{d+i-1}{2})^{d+i}}{K^{d+i}} < \left(1 - \frac{d+i-1}{2K}\right)^{d+i}$$

And the second part:

$$\frac{(d+s)^{s-i}(d+i)^{s-i}}{K^{s-i}(s-i)!} < \frac{(d+s)^{2(s-i)}}{K^{s-i}(s-i)!} < \frac{\left(\frac{(d+s)^2}{K}\right)^{s-i}}{(s-i)!}$$

In the estimations we used again the inequality between the arithmetic and geometric means and overestimated $\frac{(d+s)!}{(d+i)!}$ and $(d+i)^{s-i}$ by $(d+s)^{s-i}$. Summarizing the two estimation below we got the following asymptotic result:

$$\begin{aligned}
\lim_{d\to\infty} P(d+i) &= \lim_{d\to\infty}\left(1 - \frac{d+i-1}{2K}\right)^{d+i}\frac{\left(\frac{(d+s)^2}{K}\right)^{s-i}}{(s-i)!} \\
&= e^{-\frac{(d+i-1)(d+i)}{2K}} e^{(s-i)\ln\frac{(d+s)^2}{K}}\frac{1}{(s-i)!} \\
&= \frac{1}{(s-i)!}e^{(s-i)\ln\frac{(d+s)^2}{K} - \frac{(d+i-1)(d+i)}{2K}}
\end{aligned}$$

which goes to 0 if

$$\lim_{d\to\infty}\left((s-i)\ln\frac{(d+s)^2}{K} - \frac{(d+i-1)(d+i)}{2K}\right) = -\infty. \tag{9}$$

Since $K = o(d^2)$,

$$\lim_{d\to\infty}\left(\frac{(d+i-1)(d+i)}{2K}\right) = +\infty \tag{10}$$

Using Equation (9) and Equation (10) the following condition will be enough to $\lim_{d\to\infty}(P(d+i))=0$

$$(s-i)\ln\frac{(d+s)^2}{K}<\frac{1}{2}\frac{(d+i-1)(d+i)}{2K}$$

which is equivalent to

$$c\ln\frac{d^2}{K}<\frac{d^2}{K} \tag{11}$$

with an appropriate c constant. It stands from Equation (10), thus we got that $\lim P(d+i)=0$ for every i. Now denote by $P(\geq d)$ the probability of downloading at least d different pieces. According to the previous results:

$$\lim(P(\geq d))=\lim(\sum_{i=0}^{s}P(d+i))=\sum_{i=0}^{s}\lim P(d+i))=0$$

since s is constant, which is independent from d. ■

V. Simulation results

In this section we present some simulation results about deterministic source coding. The simulations analyze the case discussed in the previous section, when the client can not distinguish between the coded pieces before downloading them. In the simulations we analyzed the number of *defect* nodes in a network. A node is called defect if it gets at most $d-1$ different coded pieces. This means that the node is unable to decode the file. Thanks to the large piece diversity, a defect node cannot effect on its neighbors. It means that the nodes can be analyzed independently. Therefore, the simulations can be done in the following way.

Let the coded pieces are different integers between 1 and K. We independently choose d or $d+s$ integers uniformly at random among them for each client which models the downloading of d or $d+s$ coded pieces. If a node get at least d different number it is non-defect otherwise it is defect.

In the first simulation we considered the number of defect nodes when $K=d^2$ and the number of nodes is given. According to Inequality (6) the probability of being defect is between $\frac{1}{e}$ and $\frac{1}{\sqrt{e}}$. Since the nodes are independently defect or not, the following equation holds

$$\frac{n}{e}<E[n]<\frac{n}{\sqrt{e}} \tag{12}$$

where $E[n]$ is the expected value of defect nodes and n is the number of nodes. Fig. 1 shows the results when there are 5000 nodes in the network and the number of pieces vary from 1000 to 10000. The results are presented in Fig. 1.

In the second case the number of original and coded pieces are given. We varied the number of nodes in the network and considered again the number of defect nodes. Using inequality (12) the results must be between two linear curves, which is shown in Fig. 2.

The third simulation demonstrates the impact of the number of coded pieces on the number of defect nodes. As it was shown in the previous section, the number of defect nodes strictly depends on the ratio of original and coded pieces.

In the third simulation we fixed the number of nodes

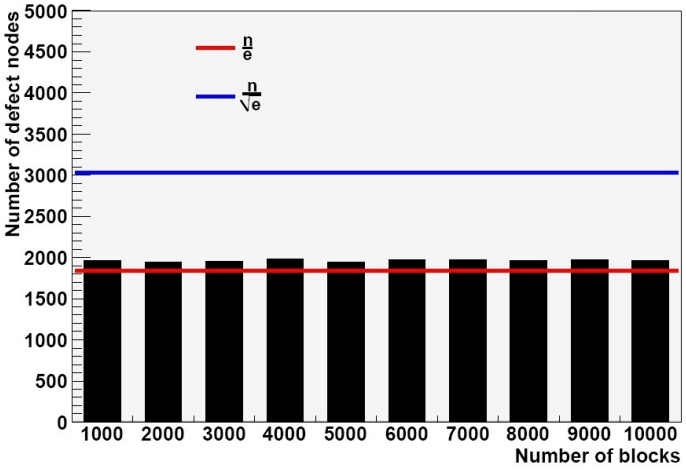

Fig. 1. The number of defect nodes. $n=5000$, $d=1000\ldots10000$

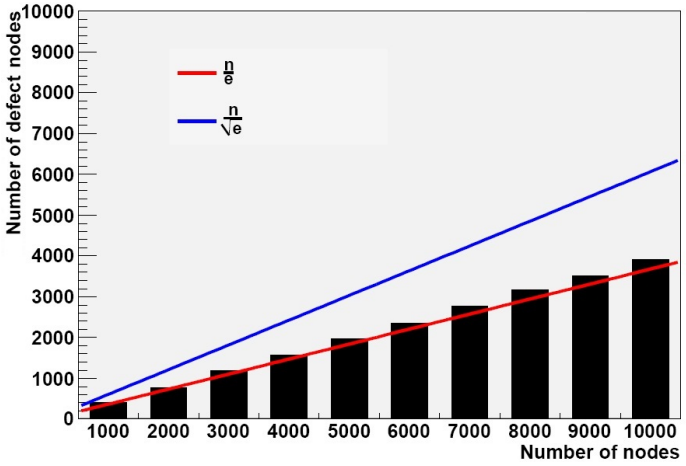

Fig. 2. The number of defect nodes. $d=1000$, $n=1000\ldots10000$

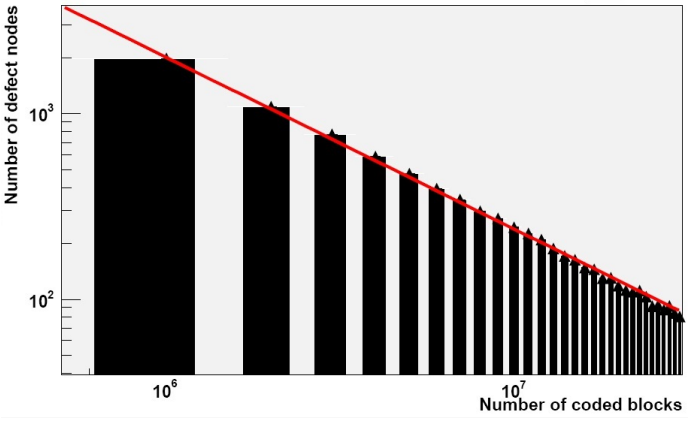

Fig. 3. The number of defect nodes. $n=5000$, $d=1000$, $K=1\ldots30\times10^6$

and the number of original pieces and varied the number of coded pieces. By increasing the number of coded pieces, the number of defect nodes quickly goes to 0. To illustrate this decreasing we present it in logarithmically scaled x-axis and logarithmically scaled y-axis in Fig. 3.

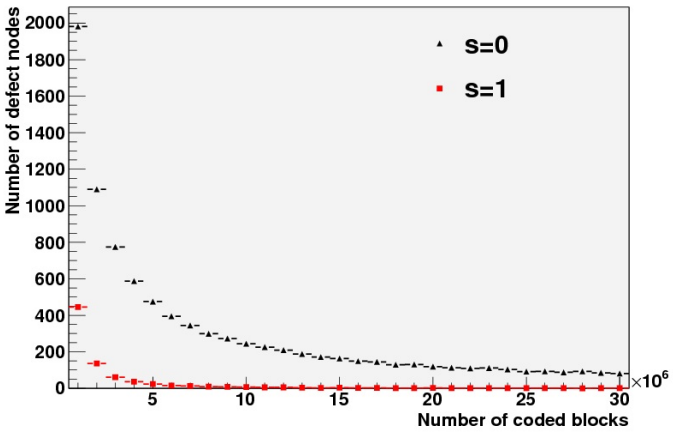

Fig. 4. The number of defect nodes. $n = 5000$, $d = 1000$, $K = 1 \ldots 30 \times 10^6$

In Fig. 4 the number of nodes and pieces are fixed, the number of coded pieces increasing from 1 to 30 million. Two cases tested here with and without an additional downloaded piece. The number of defect nodes quickly goes to 0 by increasing the number of coded pieces. The number of defect nodes decreases faster when $s = 1$. Fig. 5 illustrates this in logarithmically scaled x-axis and logarithmically scaled y-axis.

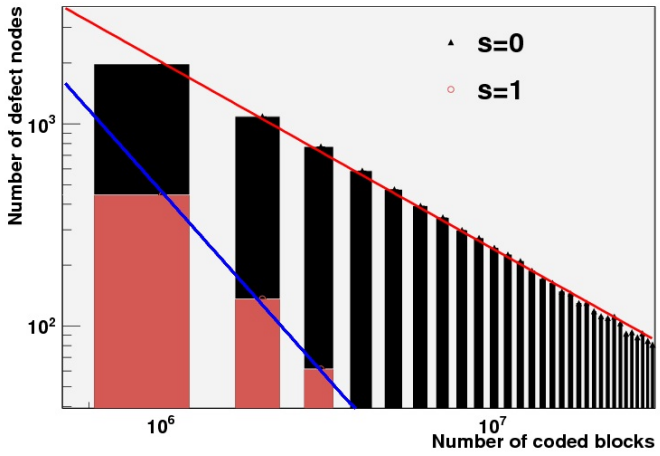

Fig. 5. The number of defect nodes. $n = 5000$, $d = 1000$, $K = 1 \ldots 30 \times 10^6$

VI. Summary

We presented a deterministic source coding method for distributing files in peer-to-peer networks. This method is an alternative to the random source coding. The main advantages of our deterministic coding are (i) that the coding will always be invertible after receiving d different coded pieces and (ii) a significantly lower communication overhead which results from the special coding vectors, that also must be known (and transmitted) for decoding the coded pieces. Instead of the whole vector only one coefficient of the vector must be send with a coded piece. The BitTorrent protocol can be extended relatively easily by our deterministic source coding method.

We also considered a case, where the peers only can check if the coded pieces are different after they download them. Using the simplifying assumption that the coded pieces are distributed uniformly in the network, we have shown that (i) if there are $K = \omega(d)$ different coded pieces, the probability that a peer can reconstruct the original file after downloading d randomly chosen coded pieces is asymptotically 1 when $d \to \infty$. (ii) If $K = o(d)$, then this probability is asymptotically 0 even if a peer downloads a constant number of additional pieces. We have verified the theoretical results by simulations.

Acknowledgement

This project is supported by the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0003.

References

[1]
[2]
[3] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.
[4] Philip A. Chou, Yunnan Wu, and Kamal Jain. Practical network coding. In *Proc. Allerton Conference on Communication, Control, and Computing*, 2003.
[5] Bram Cohen. Incentives build robustness in bittorrent. In *1st Workshop on Economics of Peer-to-Peer Systems*, 2003.
[6] Christos Gkantsidis and Pablo Rodriguez. Network coding for large scale content distribution. In *Proc. IEEE INFOCOM*, pages 2235–2245, 2005.
[7] Nicholas J. A. Harvey, David R. Karger, and Kazuo Murota. Deterministic network coding by matrix completion. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 489–498, 2005.
[8] Tracey Ho, Muriel Medard, Ralf Koetter, David R. Karger, Michelle Effros, and Ben Leong Jun Shi. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.
[9] Sidharth Jaggi, Peter Sanders, Philip A. Chou, Michelle Effros, Sebastian Egner, Kamal Jain, and Ludo Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Transactions on Information Theory*, 51(6):1973–1982, 2005.
[10] Kamal Jain, László Lovász, and Philip A. Chou. Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding. In *Proc. 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 51–59, 2005.
[11] Ralf Koetter. Coding for errors and erasures in random network coding. In *Proc. IEEE International Symposium on Information Theory*, 2005.
[12] Shuo-Yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.
[13] Thomas Locher, Stefan Schmid, and Roger Wattenhofer. Rescuing tit-for-tat with source coding. In *7th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2007.