Network Coding in P2P-Systems

Christian Ortolf

March 16, 2007

Contents

1 Introduction

Network coding is a new technique to improve datarates in scenarios with multiple peers. The basic idea of networkcoding is to use a finite field (Galois field) to create linear combinations of datablocks for transmission. A receiver can then recalculate the original data when it received enough of these blocks, also the receiver can produce new linear combinations of blocks it received and send these to the next receiver in the system. This scheme specially solves the Coupon Collector's problem.

1.1 History

- 1999 first time used in "Distributed Source Coding for Satellite Communications" [6]
- 2000 first definition of network coding in "Network Information Flow" [1]
- 2005 through Avalanche [4] Networkcoding gets into the media
- since 2000 about 200 papers about Networkcoding.

1.2 Motivation

Usually when we look at a network of nodes we describe it as a directed graph. With each node a peer in our peer-to-peer system. [1] proposes a new "MAX-FLOW MIN-CUT THEOREM" for Dataflow in a Network from a single source to multiple sinks. Which says that a Network is good if each of the sinks reaches the maximum possible flow from the traditional Max-Flow Min-Cut Theorem, beween the single source and single sink.

This bottleneck seems to be theoretic since we are potentially connected with everyone in the internet, which means if our graph has a bad min-cut we could just build a new graph with better min-cuts. But although our network is usually not static in a p2p system we may have less control over contructing a better graph. Also we wouldn't even notice a bad graph, because we have no complete knowledge on how our network looks like.

Also peers may fail, so our graph can change rapidly which makes it even for a central instance like a bittorrent tracker rather impossible to maintain such a max-flow graph. Also another problem remains, data is not a liquid substance that can be distributed as the graph has capacity, meaning not everyone is happy if he just gets x-bytes of data, everyone wants to receive

byte-1 to byte-x each only once. Thats why we make an example in Figure1 that graphs exist where the max-flow can't be reached.

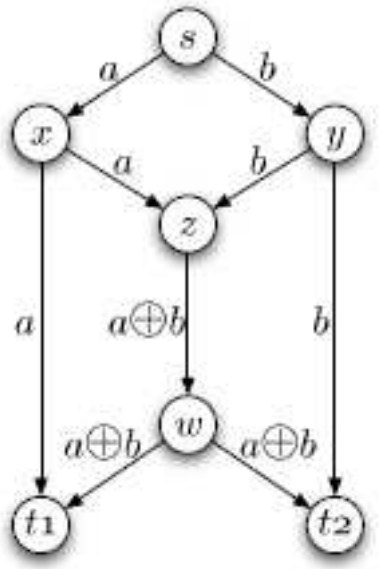

Figure 1: Max-Flow is 2 for both sinks, but can not be reached without coding

Still we used in Figure1 that every node had a complete knowledge of the network .. else the coding node wouldn't have been able to do decide to do coding at the right moment enableing us to receive what we want. And respectively other nodes didn't code or else one of the sink nodes on the bottom may have ended up with two xors of a and b unable to recreate the original data.

And thats where network coding now comes into play. It is an encoding of data that will allways optimally use the max-flow of a graph while not needing any information about how this graph looks like, and even if we can't change the graph it will guarantee us that we will optimally make use of the overlay-graph provided by the p2p network. It does this by sending linear combinations of the data instead of simply xor-ing the bits. By doing this it solves the so called Coupon Collector's problem because it is no longer important which part of the file you get because if you have enough parts you simply solve a system of linear equations to decode the data.

2 Galois field

To be able to do maths we need a field over the data so we can compute linear combinations. Galois fields can be used for this. A Galois field is a field with only a finite number of elements named after Evariste Galois. GF(p^n) would be the naming for a galois field containing p^n elements and can be constructed for any prime number p and any natural Number n.

3 Coupon Collector's problem

The Coupon Collector's problem is a kind of turned around (better known) "Birthday Paradoxon". While on the Birthday Paradoxon is counted how many people are needed to have so two have the Birthday on the same day. In the Coupon Collectors problem we would count how many people are needed to get a birthday on every day of the year. So we repeatedly draw a "Coupon" from a Set , note which we got and then put it back. The expected ammound of needed draws is then $n \log n$. So if we just randomly collect blocks of data we would need $\log n - 1$ additional blocks witch is not acceptable.

4 Coding

4.1 Encoding

We look at a file as a vector of elements from our Galois field. Especailly if we use GF(2^8) as Galois field then one element can be expressed as one byte. Which makes computation in the practice easier.

Now we devide the file in equally sized blocks.(Nr of Blocks = n) And generate a coefficient vector with size n, containing elements from our Galois field. Creating an encoded block, now works by multiplying each element in the first block, with the element in the first entry in the vector, then adding the second Block multiplied with the second entry in the vector and so on. We can create n or more Blocks like this and send them into the network.

4.2 Coding in Between

A node in between can send like the source any linear combination of the Blocks it received until that point of time. The coefficient vector of the block it sends can be easily computed from the coefficient vectors of the blocks he

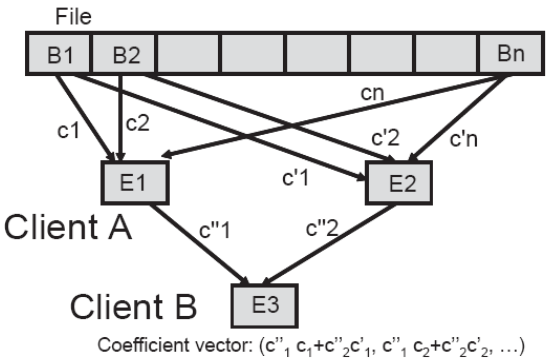

Figure 2: Blockencoding of a file from [4]

has and the new linear factors it asserts. This can be seen in Figure2 on Client A sending a newly encoded Block to Client B.

4.3 Decoding

When ever someone sends us a block he will first send the coefficient Vector he used for encoding the Block. We put that Vector in a Matrix where we store the Vectors of all received Blocks. We check if the rank of the Matrix increased, if not the Block is linear dependent and not of use to us. If it increased we download the Block. If we downloaded n Blocks, our Matrix of coefficient Vectors is Rectangular, has the full rank and can therefore be inverted. The inverted matrix then holds the coefficients we need for decoding our blocks.

5 Gain from Network coding

In the Motivation we have seen how network coding can improve performance in static or quasi-static enviroments by reaching the Max Flow of a given network.

Another gain is that NetworkCoding solves the CouponCollector problem. That has three benefits for p2p-Systems.

- The protocoll gets easier because a peers don't have to tell each other what they already have to determine what should be transmitted, if the uploading peer has something useful for us the probability is high

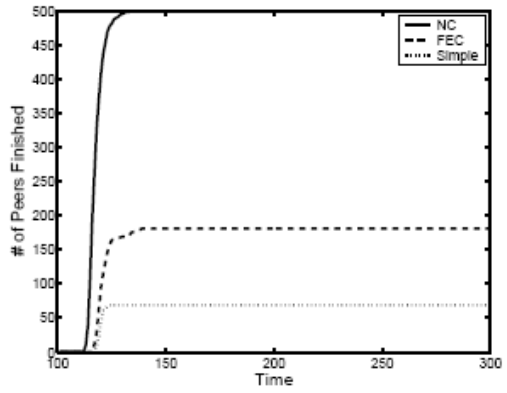

Figure 3: In the figure we can see the chances of peers finishing if the source only uploads 105% of the file and then leaves while normal peers leave as soon as they are finished. (figure taken from [4])

that a randomly encoded block from all blocks he has, will be useful for us. This could also be used in a p2p-media-broadcast to reduce the delay of a network.

- If the original seed fails the chances are greatly increased that a remaning copy stays in the p2p network. ie. Two peers that have each 50 percent of a file may usually have together about 75% but with networkcoding they may have 100% (if they didn't upload to each other before). See Figure 3.

- Often p2p-systems use inscentive mechanisms to reward peers that upload much with a better download rate, while peers that upload few get penalty. While Bittorrent tries to solve the Coupon Collector problem by giving each peer a lot of other peers he is directly connected with so he can use schemes like local rarest to achieve a good block distribution. Inscentive mechanisms make it harder for the peers to keep a good distribution and at the same time upload more to "better" peers. NetworkCoding would untangle these two problems. So it would be easier to produce better inscentive Mechanisms for a p2p system doing Network coding.

6 Problems

6.1 Overhead transmitting Coefficient Vectors

The smaller the blocksize is choosen the larger the coefficient vectors get and the more coefficient vectors have to be transmitted. For example a 4GiB file with 512KiB blocks would result in 8000 blocks meaning for GF(2^8) 64MB of data in vectors

In practice though this seems to be less a problem for example in BT we rather have 100-1000 Blocks so the total size of our coefficient vectors would vary only from 10KB to 1MB

6.2 Computation power needed

Inverting the matrix There exist asymptotically faster Algorithms than O(n^3) but those would need more time for such small Matrixes. And we need to check the Rank on the Matrix anyway on each sent coefficient Vector, with Gaussian-Elimination we can do this. If we use Gauss-Jordan- Elimination to partially invert the partially received Matrix, we also get the inverted Matrix at the same time. So while checking for the usefulness of an coefficient Vector we get the inverting nearly for free.

Decoding the blocks The O(n^2) for decoding all the blocks seems to be the part where most computing power goes in the end. After we received n blocks we have to read the data from all Blocks and multiply with the matching vector from the inverted coefficient Matrix.

Encoding the blocks Needs as much computing power as the decoding process, but this is done during the whole process of uploading the blocks. One for one so it gives us no additional wait time after finishing our download just constant computation in the background.

In [3] all these problems are described as neglictible in their tests with a 2GHZ Pentium 4 and 512MB Ram. While downloading (encoding and inverting) their test shows a usage of 20% of the cpu on finishing going up to 40% during decoding (about 5% of the downloadtime), then when finished and only uploading going as low as 10% for only encoding. The problem with this is that their cpu usage is only that low because the harddisc is the bottleneck in the decoding process at the end. Todays harddiscs are not

build for reading at 100 places of a file at the same time, but this is needed because we need the data of all blocks at once for decoding.

7 Security

Networkcoding has benefits and at the same time problems with some security aspects.

7.1 Security Problems

Homomorphic Hashfunctions In usual p2p applications files are verified using secure Hashfunctions like SHA or Tiger. Each peer can instantly verify each block he receives with the help of a set of hashvalues he may have received before starting downloding a file. Problem is that this mechanisms don't work on linear combinations of blocks. So we could only check our blocks when we have finished downloading. In between every linear combination we create that contains a corrupted block will propagate the corruption to more and more peers which will do likewise corrupting the whole network very fast.
In p2p systems using Networkcoding this is not as easy to accomplish. A solution to this would be a homomorphic Hashfunction, meaning a hashfunction H that would survive the linear encoding operations. Problematic is according to [5] that such hashfunctions are very slow about a factor of 1000 slower than a usual hashfunction. So they propose several ways to circumvent this problem. We could for example create a linear combination from all blocks we have and then hash that block. So we could check all our blocks at once and we would only have to check multiple blocks if the hashfunction finds this combination of all our blocks corrupted. Problem with this is that we have to wait rather long until we have got a bulk of blocks to verify, in this time we can't (or shouldn't) upload these blocks, especially in the beginning of the download this adds a very critical delay where we can't upload although we already have downloaded several blocks.

Another potential method proposed in [5] is cooperative hashing where everyone hashes his blocks with some probability and then tells everyone that got a linear combination from the block found to be corrupted. The problem to this method is that it is vulnerable to DoS attacks. Peers can give false alarms and like this try to jam the system. To prevent this Secure Random Checksums (SRCs) are used.

SRCs (proposed in [5]) SRCs are "Mask Based" checksums. They are generated by creating a random vector $\vec{r}$ with the lenght of a block then building the scalar product with each unencoded block. What we get as a checksum is an Element of our GaloisField. If we now want to check some block $\vec{b}$ we simply multiply the checksum of each block with the coefficient of the coefficient vector for this block and then check if the sum matches the scalar product of $\vec{r}$ and $\vec{b}$. Note that the seed must compute these checksums for us on entering the network and comunicate these checksums and the vector $\vec{r}$ to us (or the random seed it used to create $\vec{r}$)

Basically this is a fast verification of false alarms [5] proposes only to use this for the prevention of DoS attacks because the SRCs are not equally secure as Homomorphic hashfunctions though about a 1000 times faster. Much easier according to [3] may be to just compute a multiple SRCs for each new arriving peer. And send them securely to that peer. Because an attacker can only generate a block that tricks the SRCs if he knows what SRCs a peer has. Sending multiple SRCs to a peer may even increase the security because the chance of creating a corrupted block that can't be detected by pure luck gets less feasible.

7.2 Security Benefits

Tapping the wire Listening to the packets that are sent over a network doesn't give a listening peer much information except he can get all n packets. This protection may be less secure than encryption, but encryption is still not allowed everywhere, while networkcoding does not encrypt data while still giving attackers a hard time finding out what is sent in the network.

Modifieing Data Though data modificaion can usually be prevented with the help of hashfunctions. In a network doing network coding an attacker may be able to change and corrupt package. But he can't change a file directly in a way he wants, because if he is only able to modify the blocks of the file he gets he can't predict the outcome of his modifications.

8 Other Application than P2P

SAN Network coding can be used to build Storage Area Networks. Especially in p2p the redundancy Network coding can provide without the need for a central sheduling of the file blocks is very compfortable.

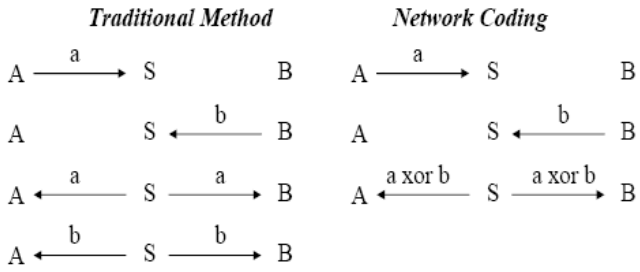

Figure 4: Throughput of S is highered. From [2]

WLAN In WLAN or other scenarios where Multicast is possible, we can get higher throughput with Networkcoding / use less energy. See Figure 4.

9 Conclusion

Network coding is a rather new and very active field of science. I am shure we will see many things evolve out of it, especially the next generation of p2p systems may use it to its benefit. The problem of computation power and disc access may be huge but the benefit of Networkcoding seems to be too large to not use it. Also computation power is growing rather fast while Solid State and hybrid discs becomeing a new standard for pcs, could solve the problem with the disc access. Also a lot of people are working on the problem trying to reduce the total cost of the coding.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] C. Fragouli, J.-Y. L. Boudec, and J. Widmer. Network coding: An instant primer. Technical report, 2005.

[3] C. Gkantsidis, J. Miller, and P. Rodriguez. Anatomy of a p2p content distribution system with network coding. *IPTPS*, 2006.

[4] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *INFOCOM*, pages 2235–2245. IEEE, 2005.

[5] P. Rodriguez and C. Gkantsidis. Cooperative security for network coding file distribution. Technical Report MSR-TR-2004-137, Microsoft Research (MSR), Oct. 2004.

[6] R. W. Yeung and Z. Zhang. Distributed source coding for satellite communications. *IEEE Transactions on Information Theory*, 45(4):1111–1120, 1999.