

The BitCod Client: A BitTorrent Clone using Network Coding*

Danny Bickson and Roy Borer
The Hebrew University of Jerusalem

Abstract

Network coding is an emerging field of research with sound and mature theory supporting it. Recent works shows that it has many benefits like improved fault tolerance, higher flexibility in selection of file parts to transfer and resiliency to network partitions [4, 3]. Despite those appealing properties there is no wide usage of network coding in real file sharing applications.

In this work, we try to bridge the gap between theory of network coding and practice. From the one hand, we deploy one the most successful file sharing client, the BitTorrent client. We use the BitTorrent algorithm for optimizing the neighbor selections for maximizing the upload bandwidth. From the other hand, we propose several simple heuristics that improve significantly the efficiency of the network coding deployed. In a nutshell, we propose computation intensive variant of network coding that can be applied to most of the existing network coding protocols. By changing the random selection of coded parts to a selection based on feedback from the network, we significantly improve the network utilization and the efficiency of the protocol.

In this paper we report our work in progress building the BitCod client. Using extensive simulations we demonstrate that our technique can compete with the performance of the state-of-the-art BitTorrent [2] file sharing client. Next, we plan to implement and test a prototype of the BitCod client over the WAN.

1 Introduction

Network coding is a novel method for improving network utilization for the transfer of data in communication networks. In this technique the transferred file is divided into parts and the code words are linear combination of the original file parts. Unlike source coding, both the source node and the participating nodes are allowed to create new codewords. The decoding is done by collecting enough linearly independent codewords and solving a set of linear equations for constructing the original file parts.

*DB was partially supported by EVERGROW IP 1935 of the EU Sixth Framework

Despite many of works investigating network coding [4, 5, 3, 6, 2] and its applications for Peer-to-peer file sharing networks, network coding is not widely deployed today in Peer-to-peer file sharing networks. Some of the reasons are that network coding complicates the protocol and creates a computational overhead.

We believe that network coding will be eventually used by Peer-to-peer file sharing systems. That is because the methods allows higher flexibility of chosen the file parts to upload and better resiliency to failures.

In this paper, we demonstrate using simulations that the efficiency of network coding in Peer-to-peer file sharing networks can be dramatically improved. The basis to our technique lies in the fact that in most of the network coding schemes, a random selection of coded file parts is taken for creating new coded parts. This is analogous to random selection of file parts in a (non-encoded) Peer-to-peer network.

Most file sharing networks, including the BitTorrent client, use the rarest part first policy, where file parts frequency is estimated locally, and rarest parts are uploaded first to the network. This simple heuristic performed locally, eventually balances the distribution of file parts in a network globally, towards a more balanced and uniform distribution. This helps speed up the download process so the nodes can reconstruct the original file faster.

In this work, we show that this simple heuristic can be applied to network coding as well, achieving a dramatic improvement in download performance.

2 Our proposed heuristics

We have decided to apply and test several simple heuristics for improving the performance of network coding. In the next section, we will analyze their performance gain using simulations and compare them to the state-of-the-art BitTorrent client. As far as we know, our proposed heuristics are generic, since they can be applied with any existing network coding algorithm.

Without the loss of generality, we assume the field $GF(2)$ is used for the coding, in other words the XOR operation is used for creating new codewords. We use the terminology *bitmap* for describing the vector of linear combination of original file parts.

The drawback of using our methods is that like in the rarest-part first, the nodes have to maintain a local estimation of the frequency of file parts. This overhead is usually negligible comparing to the actual data downloaded. In our method, we allow a one-hop feedback between nodes, where nodes update their neighbors about their current set of bitmaps.

For applying our heuristic, prior to the actual coding of new codeword, the downloading node sends the uploading node a bitmap of the codewords it has. Our assumption is that code words are file parts which can be rather large (in BitTorrent the default is 256Kb, in eMule 9.1Mb), while bitmaps are very short.

1. **H1: Missing bit.** In this method, an uploading neighbor gives higher priority to bitmaps which contains bits that do not appear in the downloading neighbor bitmaps.
2. **H2: Rarest-part first.** An uploading neighbor counts the number of bits in its local vicinity and gives priority to bitmaps with lower frequency of their bits.
3. **H3: Full decoding.** An uploading neighbor selects a bitmap that is linearly independent (if any) to the downloading neighbor bitmaps.

Note, that while all above heuristic involves sending of the download node bitmaps to the uploading neighbor, the computation overhead is different. In the first method, a linear search is performed on the array of bitmaps. In the second method, counting is performed. The third method is computationally intensive since it involves computing the rank of a matrix to check if a new bitmap is linearly independent to the existing bitmap. This method is recommended to be used on platforms where the bandwidth is the bottleneck resource, while free CPU cycles are a relatively cheaper resource.

3 Experimental Results

We have used a discrete event simulation written in Java, used by us in [1] for simulating the BitTorrent protocol. In a nutshell, we use a GT-ITM topology network of 1000 physical nodes, and on top of that an overlay network of 200 randomly chosen nodes. The downloaded file is segmented to 100 parts. We assigned transit-stub links a bandwidth of 10 parts per round, and stub-stub links a bandwidth of two parts per round. We use the GT-ITM latencies as given in the topology and compute the bandwidth consumption of the physical links based on the logical overlay traffic. In each simulation a single source node was chosen randomly, and the simulation is run in rounds until all overlay nodes have finished downloading the full file. Simulation results

where averaged over ten runs. Several GT-ITM and random topologies were tested providing the same results. For network coding, we used LT codes. For creating the overlay topology, and the management of bandwidth we use BitTorrent algorithm [2].

Our simulation results are summarized in Figure 1. As expected, BitTorrent achieved the fastest finishing time. Network coding using LT codes was significantly slower. Network coding using heuristic H2 had only a minor improvement. The best results we got using network coding where using heuristic H3. A combined heuristic of all three achieved a slightly better performance (not shown in the graph). To conclude, we have shown that using network coding and heuristic H3, we have a significant performance gain over regular network coding. The main reason is because we prevent the transfer of non-useful codewords in the network.

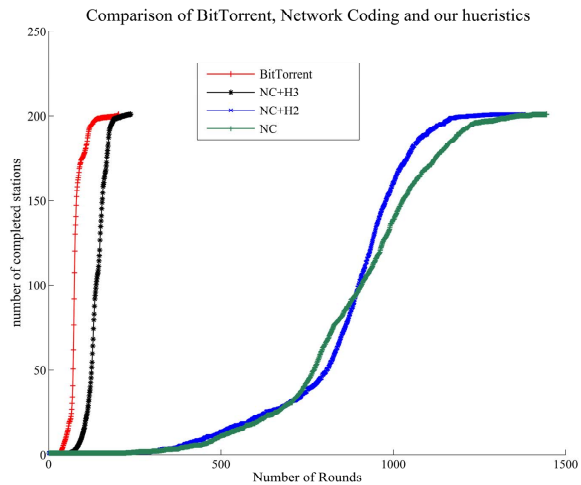


Figure 1. Performance of BitTorrent, Network coding, and network coding with our heuristics.

References

- [1] D. Bickson and D. Malkhi. The julia content distribution network. In *the 2nd Usenix of Real World Distributed Systems (WORLDS 05)*.
- [2] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of P2P Economics Workshop*, 2003.
- [3] C. Gkantsidis, J. Miller, and P. Rodriguez. Anatomy of a p2p content distribution system with network coding. In *IPTPS'06 Santa Barbara, CA, February 2006*.
- [4] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *proc. of INFOCOM 2005*, 2005.
- [5] K. Jain. Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding. *Distributed Computing*, 19:301–311(11), March 2007.
- [6] M. Luby. Lt codes. In *Proceedings of the 43rd Symposium on Foundations of Computer Science table of contents*, page 271. IEEE Computer Society, 2002.