HAL
open science

LT Network Codes: Low Complexity Network Codes

Mary-Luc Champel, Kévin Huguenin, Anne-Marie Kermarrec, Nicolas Le Scouarnec

To cite this version:

Mary-Luc Champel, Kévin Huguenin, Anne-Marie Kermarrec, Nicolas Le Scouarnec. LT Network Codes: Low Complexity Network Codes. 5th ACM International Conference on emerging Networking EXperiments and Technologies (CoNeXT), Student Workshop, Dec 2009, Rome, Italy. inria-00429680

HAL Id: inria-00429680

https://inria.hal.science/inria-00429680v1

Submitted on 14 Mar 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

LT Network Codes: Low Complexity Network Codes*

Mary-Luc Champel
Thomson R&D, Rennes, France

Kévin Huguenin
Université Rennes 1 / IRISA, France

Anne-Marie Kermarrec
INRIA Rennes, France

Nicolas Le Scouarnec
Thomson R&D, Rennes, France

ABSTRACT

This paper proposes LTNC, a new recoding algorithm to build low complexity network codes. At the core of LTNC is a decentralized version of LT codes that allows the use of fast belief propagation decoding instead of high complexity Gauss reduction used by random linear network coding (RLNC). In the context of a peer-to-peer content dissemination application, we observe that LTNC trades advantageously communication optimality of RLNC with decoding cost as it incurs only 38.5% of bandwidth overhead for a gain of almost 99% in CPU cycles.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer-Communications NetworksDistributed Systems

General Terms

Algorithms, Design, Performance, Theory

Keywords

LT Codes, Network Coding, Peer-to-peer Networks

1. INTRODUCTION

Network coding [1] is an appealing paradigm enabling to significantly improve the global throughput in content dissemination applications. Avalanche [3] is one of the most celebrated examples of such an application that uses random linear network coding (RLNC) for file sharing. In random linear network coding, nodes send random linear combinations of packets they have received, thus increasing the chances to be innovative to the receiver. These codes are optimal as n original packets can be recovered from n encoded packets with high probability. However, decoding requires an expensive ($O(n^2m)$ operations, where m is the size of the packets) Gaussian elimination. For this reason, RLNC cannot be used in numerous applications such as time constrained applications (e.g., video on demand) and power constrained applications (e.g., sensor networks) [5].

*A full version of this paper is available as a technical report [2] at `http://hal.inria.fr/inria-00416671/en/`.

Copyright is held by the author/owner(s).
CoNEXT Student Workshop'09, December 1, 2009, Rome, Italy.
ACM 978-1-60558-751-6/09/12.

In this paper, we explore the feasibility of building network codes from LT codes, which can be decoded in $O(mn \log n)$ operations. We propose a recoding technique, namely LTNC, to extend LT codes into new low complexity network codes. In a nutshell, LTNC combines cleverly, using a low complexity algorithm, the packets received into fresh encoded ones that follow the statistical properties specified by LT, thus allowing fast decoding using belief propagation.

We briefly give some background about network coding and LT codes in Section 2 before presenting our recoding algorithm in Section 3.

2. BACKGROUND

LT codes are source codes proposed by Luby in [4]. Similarly to linear codes (which they inherit), LT codes involve linear combinations of original packets. However they differ from random linear network codes in that they *(i)* specify the degree (i.e., the number of original packets involved in a linear combination) distribution of the encoded packets sent and *(ii)* they are decoded using a different algorithm called *belief propagation*.

LT packets are organized into a specific data structure named *Tanner graph*. A Tanner graph is a bipartite graph where nodes in the first set are original packets and the nodes in the second set are the encoded packets received. There exists an edge from an original packet x to an encoded packet y if x is involved in the linear combination forming y. Figure 1 depicts a sample Tanner graph. Every time an original packet x is received or decoded, every encoded packet y involving x (i.e., to which x points) is xor-ed with x and the edge between x and y is deleted. When an original packet is the only one to point to an encoded packet, it can be decoded and its value is propagated along its outgoing edges.

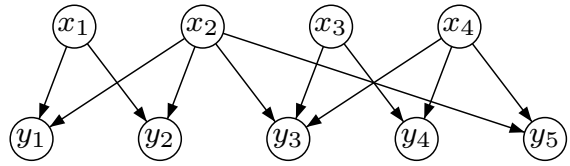


Figure 1: A sample Tanner graph: $y_4 = x_3 \oplus x_4$.

It is clear from the previous paragraph that the belief propagation decoding algorithm requires at least one en-

coded packet of degree one. More generally, the lower the degree of the encoded packets the faster the decoding. On the other hand, the higher the degree of the encoded packets, the less redundant the sent packets. It is shown in [4] that the optimal distribution of degree for the encoded packets is the Robust Soliton (see Figure 2). Secondly, to ensure optimal decoding, the distribution of degrees for the original packets must be a Dirac.

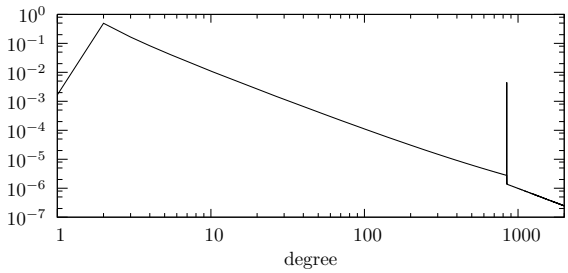


Figure 2: Robust Soliton: optimal distribution of degrees for encoded packets.

3. LT NETWORK CODES (SKETCH)

In a nutshell, our solution works as follows: when a node needs to generate a fresh packet, it *(i)* builds a packet of degree d, where d is drawn from a Robust Soliton distribution, using the encoded packets available ; *(ii)* refines the obtained packet so that the degree distribution of original packets matches a Dirac (i.e., minimum variance).

Coping with a picked degree: the degree d is a random variable drawn from the Robust Soliton distribution. We use a heuristic to detect and discard unreachable degrees. For instance, a node that received two encoded packets of degree 2 and one packet of degree 5 cannot generate encoded packets of degree higher than 9. The packet to be sent is then built iteratively by adding encoded packets available at the node in a greedy fashion. At each iteration, the degree of the packet being generated must increase and must remain smaller than d. The encoded packets available are examined in decreasing order with respect to their degree.

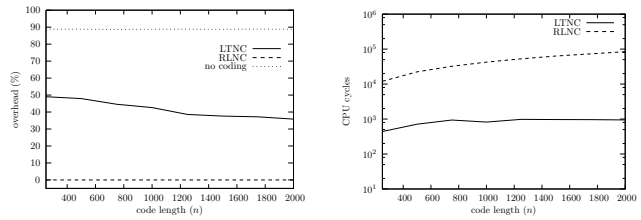
Fitting the degree distribution: in order to fit the desired distribution, an original packet involved in the packet being built can be replaced with another one using encoded packets of degree 2. For instance, if the node has generated a packet $y = x_1 \oplus y'$, x_1 can be replaced with x_2 by xor-ing y with $x_1 \oplus x_2$. Therefore, it enables to balance the degree of original packets without jeopardizing the degree of the generated encoded packet. Note that nodes have numerous opportunities for such refinement operations as half of the encoded packets are of degree 2 (Robust Soliton).

Doing so, recoding packets is achieved with no CPU overhead as compared to RLNC and enables low complexity decoding using belief propagation.

4. EXPERIMENTAL RESULTS

We evaluated LTNC using a round-based simulator. We simulated a random network of 250 peers where a single source broadcasts a file divided into n packets of size m. Each node has a static fixed-size neighborhood. At each round, a node picks uniformly at random one of its neighbor and sends it an encoded packet.

We compare LTNC to RLNC along two metrics: *(i)* the communication overhead, that is proportion of redundant packets sent; *(ii)* the decoding cost, that is the number of CPU cycles required to recover the original packets from the encoded packets.



(a) (b)

Figure 3: Simulation results in a random network.

Figure 3a plots the communication overhead as a function of the code length (n). While RLNC has almost no communication overhead (0.005%), LTNC sends less than 50% of redundant packets. This value decreases with n and reaches 38.5% for $n = 2000$. This communication overhead remains significantly lower than when packets are forwarded without encoding (89%).

Figure 3b plots the decoding cost (log scale) as a function of the code length. The number of CPU cycles required by LTNC is several orders of magnitude lower than for RLNC and the gap widens with n. For $n = 2000$, LTNC requires 100 times less CPU cycles than RLNC, that is a gain of 99%.

These results show that LTNC trades advantageously communication optimality of RLNC with decoding cost making LTNC an appealing alternative to RLNC for time and power constrained applications.

5. CONCLUSION

In this paper we proposed network codes based on LT codes (LTNC). Our experimental evaluations demonstrate that LTNC outperforms RLNC in terms of CPU cycles needed for encoding and decoding at the price of a small overhead in terms of communication. LTNC is therefore an appealing alternative to RLNC for networks of low capabilities devices. Our current work focuses on removing on the fly redundant packets that are useless to the belief propagation algorithm in order to save both bandwidth and CPU.

6. REFERENCES

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

[2] M.-L. Champel, K. Huguenin, A.-M. Kermarrec, and N. Le Scouarnec. LT Network Codes: Low Complexity Network Codes. Research Report 7035, INRIA, September 2009.

[3] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *INFOCOM*, 2005.

[4] M. Luby. LT Codes. In *FOCS*, 2002.

[5] M. Wang and B. Li. How Practical is Network Coding? In *IWQoS*, 2006.