On-the-fly Overlapping of Sparse Generations

A Tunable Sparse Network Coding Perspective

Sørensen, Chres Wiant; Roetter, Daniel Enrique Lucani; Fitzek, Frank; Medard, Muriel

Published in:
Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th

DOI (link to publication from Publisher):
10.1109/VTCFall.2014.6966091

Publication date:
2014

Document Version
Accepted author manuscript, peer reviewed version

On-the-fly Overlapping of Sparse Generations: A Tunable Sparse Network Coding Perspective

Chres W. Sørensen[1], Daniel E. Lucani[1], Frank H.P. Fitzek[1], Muriel Médard[2]
[1]Department of Electronic Systems, Aalborg University, Denmark
[2]Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
Email:{ cws, del, ff }@es.aau.dk, medard@mit.edu

***Abstract*—Traditionally, the idea of overlapping generations in network coding research has focused on reducing the complexity of decoding large data files while maintaining the delay performance expected of a system that combines all data packets. However, the effort for encoding and decoding individual generations can still be quite high compared to other sparse coding approaches. This paper focuses on an inherently different approach that combines (i) sparsely coded generations configured on-the-fly based on (ii) controllable and infrequent feedback that allows the system to remove some original packets from the pool of packets to be mixed in the linear combinations. The latter is key to maintain a high impact of the coded packets received during the entire process while maintaining very sparsely coded generations. Interestingly, our proposed approach naturally bridges the idea of overlapping generations with that of tunable sparse network coding, thus providing the system with a seamless and adaptive strategy to balance complexity and delay performance. We analyze two families of strategies focused on these ideas. We also compare them to other standard approaches both in terms of delay performance and complexity as well as providing measurements in commercial devices to support our conclusions. Our results show that a judicious choice of the overlapping of the generations provides close-to-optimal delay performance, while reducing the decoding complexity by up to an order of magnitude with respect to other schemes.**

I. Introduction

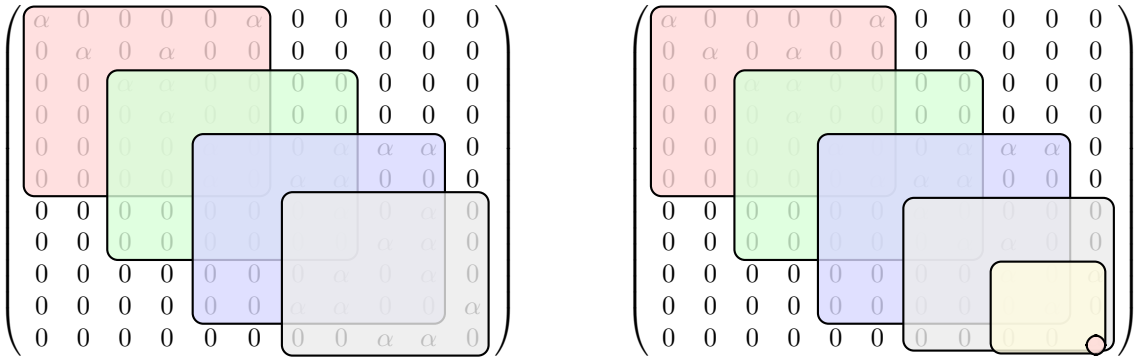
The transmission of large amounts of data to multiple users in wireless networks requires mechanisms and protocols that are (i) resilient to packet losses, (ii) able to maintain a low overhead for transmissions, and (iii) adaptive to the network devices' heterogeneous capabilities and channel conditions. Fountain codes, such as LT [1] and Raptor codes [2], pose a potential end-to-end solution to this problem. Since they exploit a belief propagation algorithm for decoding, receivers can implement a very resource efficient mechanism for decoding thus catering to a wide variety of devices. A key limitation of these codes is the fact that encoders need to follow a very strict density distribution to ensure decodability with low overhead (delay). Thus, LT and Raptor codes are useful only for end-to-end applications, which is inefficient in multi-hop scenarios.

Network coding provides an alternative solution by encouraging intermediate nodes in the network to operate on its incoming coded packets in order to generate new coded packets. The impact of recoding at intermediate nodes, i.e., coding in the network, allows to achieve the multicast capacity in lossless wireline networks and on lossy, multi-hop wireless networks. The latter comes in part from the ability to generate redundancy that is tailored to each wireless link, instead of generating redundancy end-to-end. Random linear network coding (RLNC) showed that recoding can be carried out in a distributed fashion by simply generating linear combinations of received packets using random coefficients drawn from a finite field [3]. In contrast, recoding capabilities in LT [1] and Raptor codes [2] at intermediate nodes has proven difficult to achieve without modifying the underlying code structure, e.g., [4].

A key limitation in RLNC lies in its decoding complexity, which is more resource expensive than belief propagation. In fact, given N packets of size K symbols in the given finite field, Gaussian elimination requires $O(N^3+N^2K)$ operations to decode. Some approaches, such as systematic network coding [5] provide simple alternatives to reduce this complexity by sending uncoded packets at first, followed by RLNC packets later on. However, its applicability is typically limited to a few hops, as less uncoded packets will be received when traversing multiple, lossy links. From a practical perspective, complexity is reduced by splitting larger files into multiple disjoint generations of packets [6]. Thus, the system retains its recoding capabilities and maintains a complexity that is linear on the number of generations (although with a large constant), but at the cost of increased overhead. Generations can be transmitted sequentially or in a round-robin fashion [6] as well as by using a random schedule [7], using more or less feedback messages and smaller or larger storage, respectively.

The overhead introduced by splitting the file into smaller generations can be reduced by letting the chunks overlap [8], [9]. That way, an original packet may be contained within multiple generations. When a packet gets decoded within one generation, it may be back substituted into any other generation that contains it. This insight has spawned a variety of approaches from considering overlaps of generations with different sizes [10] to trade-off delay/overhead and complexity, to codes that use a sparse pre-coder before creating generations, e.g., BATS codes [11]. Existing approaches have relied in the use of RLNC for coding within generations and an attempt to restrict the use of feedback in the transmission process.

This paper advocates that exploiting sparse coding within generations, instead of RLNC, and leveraging occasional feedback is instrumental to generating overlapping generations on-the-fly and providing a low complexity, low overhead solution. More generally, our approach allows us to trade-off the overhead in the use of the channel with decoding complexity



(a) Overlapping method

(b) Overlapping method with variable n

Figure 1. Example of proposed overlapping generation methods. ($m = 11, n_{max} = 6, r = 4$)

to allow resource limited devices to exploit network coding. Our proposal is inspired in part by the results of [12] in Tunable Sparse Network Coding (TSNC) and its potential for recoding sparse codes. In fact, our proposal constitutes a specific implementation of TSNC, where the coding density is increased by dropping original packets that have been "seen" at the receiver as part of the pool of packets considered to generate coded packets (using the notation in [13]).

This paper proposes and analyzes families of on-the-fly, sparsely coded generations and compares it to various non-overlapping and overlapping generations approaches. We focus our evaluation on delay/overhead performance as well as complexity. For the latter, we consider measurements on commercial platforms to understand the processing time required by the different approaches. We show that specific configurations of feedback and sparsity in our approaches can provide a low overhead solution with several fold to an order of magnitude gain in processing time compared to all other approaches.

II. Model and Preliminaries

A. System Model

We consider the case of a sender transmitting a large group of m data packets to a set of receivers over packet erasure channels. We order the packets with a given index with the lowest index assigned to the first packet in the file. The sender organizes the data packets in generations with a smaller subset of the packets. Coded packets are generated by using linear combinations of the packets in each generation choosing the coding coefficients in a sparse fashion, i.e., by choosing only a limited number of non-zero coefficients.

A receiver transmits feedback packets to signal to the sender which packets have been *seen* up to that point, using a similar notation to [13]. A *seen* packet constitutes a packet that has been included in a received linearly independent coded packet. Each linearly independent coded packet can provide a single and unique *seen* packet at the time of sending a feedback packet. Packets with lower index are prioritized to have a greater overlap across multiple receivers. If r packets were not *seen* in the previous generation, we say that the overlap between two generations is r packets.

A signaling event, i.e., reception of feedback packets at the sender, is generated before the transmission of all packets in the generation has been completed. The signaling event triggers the creation of a new generation, which overlaps with the previous one. The overlap is given by those packets that were not *seen* by all receivers. At this point, the sender can eliminate from its queue all packets that were *seen* packets by all receivers. This provides us with a coefficient matrix of the structure illustrated in Figure 1a. Finally, feedback is assumed to be lossless and delay free, for simplicity.

B. Proposed Approaches

The use of sparse coding for the overlapping generations causes the probability of receiving a packet that is linearly independent of previously received packets to decrease as more coded packets of the generation are received. The main idea of letting generations overlap is to increase the innovation probability of coded packets, i.e., the probability of coded packets to be linearly independent, such that decoding can be performed with less received coded packets. Thus, generating a signaling event more often, i.e., increasing the frequency of feedback, results in a higher overlap between generations and, more importantly, in a lower overhead overall. The latter is a consequence of maintaining a high probability of receiving linearly independent coded packets during the entire transmission.

We propose two methods for overlapping generations based on the above paradigm. First, a method that defines a fixed generation size of n packets and a target overlap size of r. The last generation size will be lower or equal to the others in general. This method is referred to as OG in the remaining. Figure 1a shows an example of this method in terms of the senders coding coefficients per sent generation. The example has $m = 11$ packets in total, which are split into smaller equally sized overlapping generations of $n = 6$ packets plus a potentially last generation of n packets or less. The generation overlap is $r = 4$ packets. Finally, the last generation will be $n_{\text{last}} = 5$ packets.

Since the last generation will be responsible for the highest overhead, i.e., additional received coded packets, it may be beneficial to reduce the sizes of the last generations as

illustrated in Figure 1b. This approach will be referred to as decreasing overlapping generations (DOG), and differs from overlapping generations (OG) by letting the overlap size, r, decrease such that generations may shrink in the end.

C. Metrics

We will focus on two performance measures throughout this paper. First, the number of received packets required to decode all m data packets. This allows us to measure the overhead of the different schemes. Second, the decoding time required to decode the m data packets using commercial devices.

III. Analysis

This section presents an analysis for our proposed overlapping sparse generation schemes described in Section II. We also provide a similar analysis for comparison schemes. We will start by defining an upper bound for the estimate of the probability of a coded packet to be innovative, i.e., linearly independent, to a receiver that has accumulated i linearly independent packets. This probability can be calculated for a generation size of n data packets and a density, d as

$$P(i,n,d) = P_{innovative}(i,n,d) \geq 1-(1-d)^{n-i}. \quad (1)$$

This bound was used in [14].

Using Eq. (1), the expected number of packets needed to be received to increase a decoders rank by one can be calculated by $1/P(i,n,d)$. With that in mind, we can derive the expected number of packets needed to be received to decode a generation.

If we use a single generation (SG), the expected number of received coded packets is

$$E_{\text{SG}}(m,d) = \sum_{i=0}^{m-1} \frac{1}{P(i,m,d)}. \quad (2)$$

In contrast, a non-overlapping generation scheme (NOG) consisting of k disjoint generations, $(k-1)$ equally sized and one generation of the same size or smaller. The expected received packets required to decode can therefore be found as

$$E_{\text{NOG}}(m,n,d) = (k-1)\sum_{i=0}^{n-1} \frac{1}{P(i,n,d)} + \sum_{i=1}^{n_{last}-1} \frac{1}{P(i,n_{last},d)}. \quad (3)$$

The number of generations is given by $k = ceil(m/n)$. The last generation size is found to be $n_{last} = m-((k-1)n)$.

For our proposed OG scheme, if we consider Figure 1a, we see that only the first $n-r$ packets of each generation, except the last one, are transmitted. The last generation should however be transmitted as a normal generation. This means that the expected number of received coded packets is given by

$$E_{\text{OG}}(m,n,r,d) = (k-1)\sum_{i=0}^{(n-1)-r} \frac{1}{P(i,n,d)} + \sum_{i=0}^{n_{last}-1} \frac{1}{P(i,n_{last},d)}. \quad (4)$$

The number of generations can be calculated as $k = 1 + ceil((m/n)(n-r))$, and the last generation will have the size $n_{last} = m-(k-1)(n-r)$.

Four our proposed DOG scheme, there is a decreasing overlap size and the reduction in generation sizes, which means that

$$E_{\text{DOG}}(m,n,r,d) = \sum_{j=0}^{k-1} \left(\sum_{i=0}^{(n_j-1)-r_j} \frac{1}{P(i,n,d)} \right). \quad (5)$$

The number of generations by $k = ceil(m/(n-r))$. For each generation $j = \{0,1,\ldots,k-1\}$, we find the j'th generation size $n_j = min(n, m-j(n-r))$, and the decreasing overlap $r_j = max(0, n_j - n + r)$.

Finally, we consider a systematic approach with overlapping generations (SR). In each generation, the packets are first transmitted uncoded, and then finished using RLNC. The analysis is similar to the NOG scheme, where we have $(k-1)$ equally sized generations and one generation of same size or smaller. We complete one generation at a time, so we still sum the expected received packets required to decode in order to find a total amount of packets required to decode for m packets.

However, SR differs from the other methods since it does not consider a sparsely coded set of generations. Therefore, we are dependent on which packets are lost, and the packet erasure, e, has therefore been included into the expression

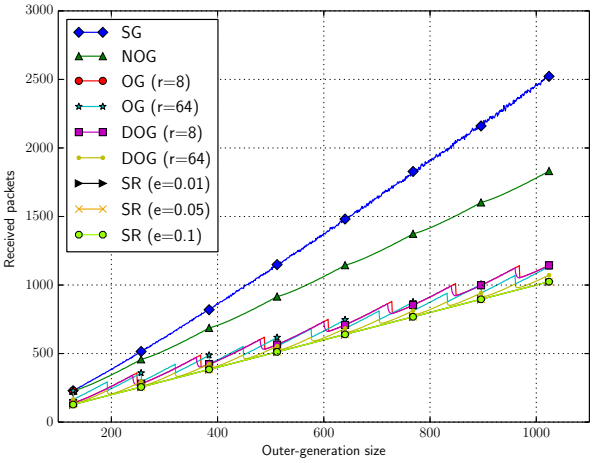
$$E_{\text{SR}}(m,n,d,e) = (k-1)\sum_{l=0}^{n} \left(\text{Bi}(l,n,1-e) \left(l + \sum_{i=l}^{n-1} \frac{1}{P(i,n,d_{rlnc})} \right) \right) + \sum_{l=0}^{n_{last}} \left(\text{Bi}(l,n_{last},1-e) \left(l + \sum_{i=l}^{n_{last}-1} \frac{1}{P(i,n,d_{rlnc})} \right) \right). \quad (6)$$

where $Bi(l,n,p)$ represents a binomial distribution, where n is the generation size, p is the probability of successfully receiving coded packets, and $l = \{0,1,\ldots,n\}$ represents the number of uncoded packets received. We find the number of generations, $k = ceil(m/n)$, and the last generation size, $n_{last} = m-((k-1)n)$.

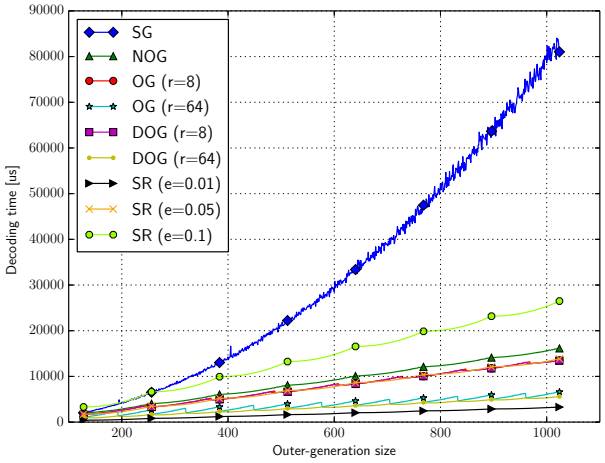
IV. Performance Evaluation

This section will be used to present the performance of our proposed methods, OG and DOG. The proposed methods will be compared to three other methods for transmitting a large group of packets: (1) transmitting all packets in a single generation using a 3-sparse density, named SG; (2) transmitting packets in smaller non-overlapping generations one generation at a time using 3-sparse, named NOG; (3) transmitting non-overlapping generations one at a time as in (2), but using systematic coding with RLNC to complete each individual generation that experienced packet losses. We refer to this method as systematic RLNC (SR).

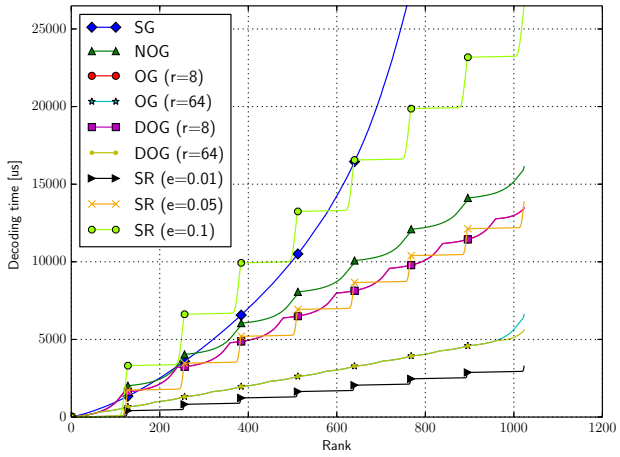
We have measured the average time spend decoding a generation until a given rank i is obtained. These measurements



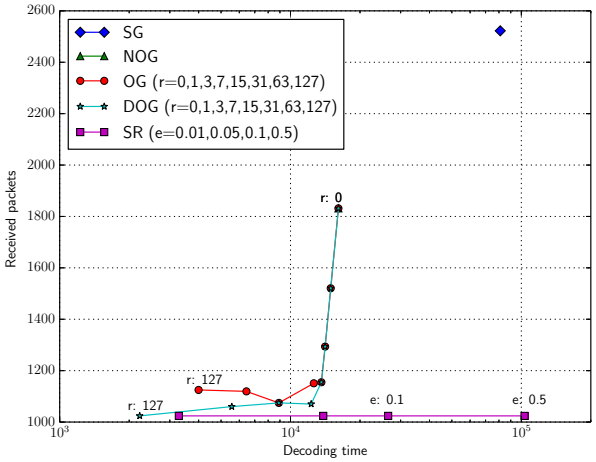
(a) Received packets required to decode as the outer-generation size is increased. (3-sparse, m={n..1024}, n=128, r={8,64}, e={0.01,0.05,0.1})



(b) Decoding time when the outer-generation size is changed. (3-sparse, m={n..1024}, n=128, r={8,64}, e={0.01,0.05,0.1})



(c) Time spend on decoding to obtain a given rank (3-sparse, m=1024, n=128, r={8,64}, e={0.01, 0.05, 0.1})



(d) Received packets as a function of decoding time (3-sparse, m=1024, n=128, r={0,1,3,7,15,31,63,127}, e={0.01, 0.05, 0.1, 0.5})

Figure 2. Results using $GF(2^8)/\{0\}$.

were performed on a decoder implemented in KODO [15]. Because we have the time spend to achieve a given rank, the measurements can simply be inserted in the equations of Section III to archive an estimate of the decoding time of the schemes given that they were implemented.

More specifically, we implemented a single-hop, single receiver setup and measured the time spend and the average number of symbols received by the decoder at each obtained rank. This was done for a single generation of size ranging from 1 to 1024 symbols, and a 3-sparse coding density such that $d = \min\left(0.5, \frac{3}{n}\right)$ for $GF(2)/\{0\}$ and $d = \min\left(1, \frac{3}{n}\right)$ for $GF(2^8)/\{0\}$. All measurements have been performed with packets of size 1500 bytes in $GF(2)$ and $GF(2^8)$, but the results will only be presented for $GF(2^8)$ since both fields show the same tendencies.

Given the measurement data, we can plot the packets required to decode an outer-generation of various sizes for each methods using the results from Section III. This is illustrated in Figure 2b, where the inner generation is $n = 128$ symbols is kept constant.

Figure 2a shows that even with a density as low as 3-sparse, we obtain a performance that is far below SG and NOG, while performing only slightly worse than SR, which is optimal in terms of delay performance. Furthermore, DOG seem to perform slightly better than OG in terms of overhead, i.e., received coded packets.

Figure 2b measures the decoding time of the same schemes. This time only measures the time invested in processing and, thus, is not affected by packet erasures on the communication channel. The packet erasures do however punish the SR method since an increased packet loss probability will cause more systematic packets to be lost and eventually replaced by

RLNC packets. This is due to the fact that systematic packets require essentially no processing time, while RLNC packets are very dense and thus very time consuming to decode.

Figure 2b shows also that SR performs better with low erasure probabilities, as expected, while OG and DOG perform better in case of increased erasures (> 5 %). We also see that a higher overlap is better in terms of decoding time. This may however change in a final implementation due to changes in the back-substitution and book-keeping mechanisms [16].

Figure 2c considers the average decoding time it takes to obtain a given rank during transmission of an outer-generation of size $m = 1024$ packets. It is based on time measurements and generated using the equations presented in previous sections. Obtaining a rank is essentially the same as receiving an innovative packet, but does not mean that the packets can be decoded yet. Again, we see the same tendencies as in the previous figures. The SR is very dependent on the erasure probability and will perform better in case of low erasures, but even with a relative small erasure probability it will be outperformed by OG and DOG.

Finally, Figure 2d shows the explicit trade-off between received coded packets as a function of decoding (processing) time, considering the effect from the overlap size, r, and channel erasures on OG, DOG, and SR. The performance of SG is mediocre both in overall processing and delay performance, while the NOG method performance has similar performance to our proposed OG and DOG without overlap, $r = 0$. Increasing the overlap between generations, decreases the processing time on the overlapping methods due to less dependent packets. SR has the lowest probability of receiving dependent packets, but increasing the erasure even mildly will cause its decoding processing time to increase dramatically by more than an order of magnitude. Thus, DOG can provide close-to-optimal performance in delay (overhead) performance while providing a significantly smaller processing effort on the receivers. The feedback requirements for DOG and OG are mild and comparable in many cases to those of SR, i.e., a single feedback per generation used.

V. Conclusions

This paper advocates for an on-the-fly strategy for overlapping generations of data packets, while maintaining a sparse coding over the packets of each generation. More specifically, we propose two families of solutions that leverage a small amount of feedback to provide a controllable complexity-delay trade-off. Inherently, this article brings together the problems of overlapping generations and the tunable sparse network coding in a common setting.

Our comparison to alternative schemes were based on both delay/overhead performance and processing time on commercial devices. Our results showed that our proposed overlapping of sparse generation significantly decreases the number of received packets required to decode a large group of data packets. The level of overlap between generations has an important effect on performance, where a higher overlap maps into a better delay performance. We also showed that our proposed methods are very dependent on the last generation size, thus opening the door for future research in optimizing the generation sizes of the generations along the entire transmission process. Overall, we showed that our proposed methods can provide close-to-optimal delay performance, while reducing the processing effort by orders of magnitude in real systems.

Future work shall focus on more complex network settings, considering the effect of imperfect feedback, and considering the effect of recoding coded packets at intermediate nodes.

Acknowledgements

This work was financed in part by the Green Mobile Cloud project granted by the Danish Council for Independent Research (Grant No. DFF - 0602-01372B) and TuneSCode project granted by the Danish Council for Independent Research (Grant No. DFF 1335-00125)

References

[1] M. Luby, "Lt codes," in *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, 2002, pp. 271–280.

[2] A. Shokrollahi, "Raptor codes," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2551–2567, 2006.

[3] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[4] S. Puducheri, J. Kliewer, and T. Fuja, "The design and performance of distributed lt codes," *Information Theory, IEEE Transactions on*, vol. 53, no. 10, pp. 3740–3754, Oct 2007.

[5] D. Lucani, M. Medard, and M. Stojanovic, "Systematic network coding for time-division duplexing," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, June 2010, pp. 2403–2407.

[6] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. of Allerton Conference on Communication, Control, and Computing*, October 2003.

[7] P. Maymounkov and N. J. A. Harvey, "Methods for efficient network coding," in *Proc. of Allerton Conference on Communication, Control, and Computing*, September 2006, pp. 482–491.

[8] D. Silva, W. Zeng, and F. Kschischang, "Sparse network coding with overlapping classes," in *Network Coding, Theory, and Applications, 2009. NetCod '09. Workshop on*, June 2009, pp. 74–79.

[9] A. Heidarzadeh and A. H. Banihashemi, "Overlapped chunked network coding," *CoRR*, vol. abs/0908.3234, 2009.

[10] Y. Li, W.-Y. Chan, and S. Blostein, "Network coding with unequal size overlapping generations," in *Network Coding (NetCod), 2012 International Symposium on*, June 2012, pp. 161–166.

[11] S. Yang and R. Yeung, "Coding for a network coded fountain," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, July 2011, pp. 2647–2651.

[12] S. Feizi, D. E. Lucani, and M. Médard, "Tunable sparse network coding," in *Proc. of the Int. Zurich Seminar on Comm.*, March 2012, pp. 107–110.

[13] J. Sundararajan, D. Shah, and M. Medard, "Arq for network coding," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, July 2008, pp. 1651–1655.

[14] S. Feizi, D. E. L. Roetter, C. W. Sørensen, A. Makhdoumi, and M. Medard, "Tunable sparse network coding for multicast networks," *2014 International Symposium on Network Coding*, 2014.

[15] M. V. Pedersen, J. Heide, and F. Fitzek, "Kodo: An open and research oriented network coding library," *Lecture Notes in Computer Science*, vol. 6827, pp. 145–152, 2011.

[16] J. Heide, M. Pedersen, and F. Fitzek, "Decoding algorithms for random linear network codes," *Lecture Notes in Computer Science*, vol. 6827, pp. 129–137, 2011.