Systematic Network Coding for Packet Loss Concealment in Broadcast Distribution

Rui Prior, *Member, IEEE*, and André Rodrigues
Instituto de Telecomunicações—Universidade do Porto

***Abstract*—This paper describes a system for the distribution of content over multicast and broadcast media that uses network coding techniques for concealing packet losses from the applications, improving the received Quality of Service. We propose a systematic network code designed for this purpose, analyze the performance of the code, comparing it with a standard approach based on random linear network coding, and provide experimental results obtained in an implemented prototype of the system.**

I. Introduction

Dealing with packet losses involves one of two general approaches (or a combination of both). The first approach is the use of automatic repeat request (ARQ), a closed-loop technique using a combination of reception acknowledgments, timers and packet retransmission. In a carefully designed ARQ mechanism, the amount of retransmitted packets can be made as close as desired to the amount of lost information, thus avoiding unnecessary overhead, while ensuring perfect reliability. However, ARQ requires a feedback channel (e.g., duplex connection), introduces a recovery delay that is necessarily larger than the round-trip time (RTT), and still can only provide probabilistic reliability within bounded time. The second approach is the use of an erasure code. Erasure codes generally consist on transmitting redundant data such that even if a subset of the transmitted packets is lost, up to a certain fraction, the receiver can still reconstruct the original data. This approach has the disadvantages of transmitting redundant data even if no actual losses occur (unnecessary overhead) and providing only probabilistic reliability; however, it does not require a feedback channel, and can lead to a substantially lower playback delay than ARQ.

This paper proposes a system for multiparty distribution of content with packet loss concealment based on the second approach. However, contrary to traditional erasure codes, the coding (or recoding) process can take place not only at the source, but also in certain network nodes. By locating the coding layer below the application layer, it becomes application-independent, and is provided as a service that can be turned on or off as desired. We propose the use of systematic network code (SNC) where densely coded redundant packets are sent along with the original (uncoded) packets. Compared to a non-systematic code, SNC has the benefits of better erasure correcting performance, lower processing load at the decoder and lower average compound delay.

This work is partly funded by the GEN-CAN project funded by IT and the GTI-CANE project funded by the FCT.

It is worth noting that the proposed system is meant to compensate for losses introduced by link errors, not by congestion—sending redundant data through a congested link would aggravate the situation, and congestion control is not feasible without a feedback loop. However, it can also be used to compensate for congestion-related losses if applied only to a small fraction of the total traffic—older versions of TCP, still widely deployed, use packet losses to trigger congestion control, and the proposed system can protect low rate UDP flows from such induced congestion-related losses.

The rest of the paper is organized as follows. Section II presents relevant related work. Section III describes the general model of the system and the proposed SNC, whose performance is analyzed in section IV. In order to facilitate a better grasp of the actual performance of the system, section V presents some numerical results using the previous analysis along with experimental results obtained in a prototype implementation. Finally, section VI contains some closing remarks and an identification of possible topics for further development of this work.

II. Related Work

The concept of network coding (NC), originally proposed in [1], has drawn a lot of attention in the past few years, and even though most work in the area has been theoretical, some practical applications have emerged. Network coding is based on the principle that network nodes are not limited to simply retransmit the received information according to given rules, they can also transmit combinations of the received information in order to obtain certain benefits including, but not limited to, increased throughput. With linear NC, a receiving node only needs a sufficient number of linearly independent combinations of the original packets in order to decode a flow, the exact combinations received are irrelevant. This fact led to the proposal of random linear network coding (RLNC) [2], an entirely distributed mechanism that dispenses with any coordination between the nodes, and that was later proven to achieve the network capacity with high probability [3]; it also suggested that NC can be used to add resilience against packet losses, similarly to an erasure code.

The use of NC for increased reliability has been proposed before. In [4] the performance of NC for reliable multicast (in terms of the required number of transmissions) is shown to be better than that of ARQ and rateless codes; the authors assumed a dense code. In [5], a combination of NC and ARQ is used in an online algorithm that minimizes the queue

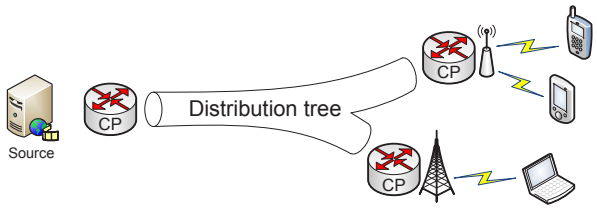

Figure 1. System architecture

size at the sender (transmission window). The use of a mix of uncoded and coded packets (systematic coding) has been proposed in [5] and [6] with the goal of minimizing the decoding delay in media streaming through reliable multicast. All of these models, however, differ from ours in that they use a feedback loop from every receiver to implement reliability. Clearly, such mechanism cannot scale well to a very large number of receivers, and a feedback channel is not always available (e.g., in the case of broadcast networks). Moreover, these works assume that feedback is perfect, i.e., without errors, losses or delay (the remarks on the effects of imperfect feedback in [6] suggest that their algorithm can be used with minor modifications in the case of imperfect feedback, though). In fact, reliability in these models is provided not by the network code itself, but by the feedback mechanism—the network code is used solely to reduce the delay and overhead of retransmissions. In contrast, our approach is more similar to erasure codes, in that it provides stochastic resilience against network packet losses through the inclusion of redundancy, without the use of a feedback loop.

III. System Description

A. General Architecture

The general architecture of the proposed system is illustrated in fig. 1. Unaware of the coding process, a standard application generates the content to be distributed. A middleware layer installed at the source node or the access router captures the packet flow (according to specified filtering rules) and, using the coding scheme described below, generates a flow of coded packets that is sent down a distribution tree. Some nodes downstream may perform recoding of the flow, usually access routers at the receiving side in order to adapt the coding parameters to the characteristics of the access channel. Whenever a feedback channel is available, extremely low rate feedback can be used to tune the coding parameters to the packet loss probability seen by each group of receivers; however, feedback is not considered in our analysis.

At the receivers, a middleware layer receives the flow of coded packets, decodes it, and injects the original packets, in order, into the protocol stack to be delivered to the application (eventually leaving some holes corresponding to packets that could not be decoded).

In addition to the coded flow, information needs to be sent downstream to the receivers for configuration and synchronization. According to the specific needs of the setup, this information can be sent in-band or out-of-band (through a side channel). The system addresses other aspects besides packet losses; however, they are out of the scope of this paper.

B. Systematic Network Code

In packetized network coding, each packet is regarded as a sequence of symbols that are elements of a given finite field. Each coded packet is computed from a set of input packets and a set of coefficients (one per input packet) that are also elements of the same field. Each symbol of the coded packet is computed as a linear combination of the corresponding symbols of the input packets, and the same coefficients are used to compute all the symbols of a coded packet. At the receiver, a set of decoding coefficients is computed from the coding coefficients by matrix inversion, so that the original packets can be computed from the received coded packets.

The traditional approach to RLNC is to choose each coding coefficient uniformly at random from the used finite field [2], a strategy that we will call Dense Network Coding (DNC) since all transmitted packets are dense combinations of the original packets. While this strategy is optimal in the absence of packet losses, it is clearly a bad choice when NC is used to compensate for them. Consider that for a generation of N original packets $N+M$ coded packets are sent. If at most M packets are lost, all of the original packets will be decoded w.h.p.; however, if more than M packets are lost, the matrix of received coding coefficients will not be full rank, and no packets can be decoded. This means that even though DNC can improve the packet delivery ratio when the loss probability is very low, it yields worse results than no coding at all when the probability of losing more than M packets per generation is no longer negligible.

In order to avoid the aforementioned problem, we propose the use of a Systematic Network Code. For each generation of N original packets, $N+M$ packets are sent, where the first N are the original packets. The packets received out of these N will be delivered to the application irrespectively of the reception of other packets, since they are already "decoded". The last M packets, which we will call redundant packets, are coded using coefficients uniformly chosen at random among the elements of the finite field except zero (since in practice M will be low, excluding the zero maximizes the probability that a densely coded redundant packet is useful for compensating the loss of any original packet). The coding matrix is, therefore, a $(N+M)\times N$ matrix where the upper N rows are those of the $N\times N$ identity matrix and the lower M rows contain randomly chosen non-zero elements from the field. The upper N rows are obviously linearly independent, and, given a large enough field, each of the lower M rows is linearly independent from any set of $N-1$ other rows w.h.p..

For decoding, the receiver performs (slightly modified) Gauss-Jordan elimination on a matrix built from the coding matrix with the rows corresponding to lost packets zeroed out. Since the N leftmost columns of many rows in this matrix will already be rows from the $N\times N$ identity matrix, much of the elimination work will not be necessary—an added benefit of

the SNC is that it imposes a lower processing load at the receiver for decoding than DNC.

C. Practical Aspects of the Code

The proposed code works at the network layer of the TCP/IP stack, and is applied only to the data field of the IP packets. However, if the stream uses packets of different sizes, an additional 16-bit header is added to the redundant packets containing the length of the data fields of the packets, coded similarly to the data. This header is necessary to retrieve the data length of lost packets that are restored from the redundants.

The ID field of the IP header is partitioned into two 8-bit fields, one used as a circular counter to identify the generation number, and the other one to identify the packet number inside the generation. The total number of packets in a generation, $N + M$, is limited to 255 (zero is reserved); however, due to practical considerations on decoding complexity, this limitation is not important.

Regarding the finite field, we chose $\mathrm{GF}\left(2^8\right)$ for several practical reasons. Firstly, the representation of each element in $\mathrm{GF}\left(2^8\right)$ occupies exactly one byte, which is advantageous in that it is a natural unit to work with and avoids wasting representation space. Secondly, the necessary operations can be efficiently implemented using small, cache-friendly *log*, *antilog* and *inverse* lookup tables of 256 bytes each. Thirdly, since we will use relatively small generation sizes, $\mathrm{GF}\left(2^8\right)$ is large enough for the probability of generating linearly dependent redundant packets to be negligible[1].

In order to avoid the overhead of sending the coding coefficients along with the data packets, as in [7], we resort to pseudo-random number generators (PRNGs). The PRNG of each receiver has to be synchronized with the PRNG of the sender (or the last coding point downstream). Information for re-synchronizing the receivers is periodically sent through the side channel so that new receivers can join in the middle of a stream.

IV. Performance Analysis

We now analyze the packet loss concealment properties of the proposed algorithm and compare it to the standard transmission over a lossy channel (packet erasure channel). Since our algorithm sends M additional packets for each block of N packets, we also compare its performance to a non-coded algorithm where, in addition to the N original packets, a second copy of each of M packets selected randomly among the original N is sent. For the sake of completeness, we also include in our comparison an optimal non-systematic dense coding scheme where for each set of N original packets, $N+M$ coded packets are sent where all possible subsets of N coded packets among the $N+M$ sent are linearly independent. We will now compute the expected fraction of received and (when applicable) successfully decoded packets at the receiver (i.e., the fraction of original packets that are successfully delivered to the application at the receivers), assuming that packet loss events are independent and occur with probability p.

Without the use of NC or redundancy, for each set of N original packets, N packets are sent (the original ones). Since each packet has a probability of successful delivery of $(1-p)$, the expected fraction of packets that are successfully delivered will be, on average

$$1 - p \tag{1}$$

Without NC but with redundancy, for each set of N original packets, $N+M$ packets are sent—the N original ones plus one additional copy of M of the original packets $(M \leq N)$. These packets are randomly selected, but are all different (i.e., it is impossible to send 3 copies of the same packet). Obviously, the fraction of packets that is successfully received will be somewhat increased—even if the first copy is lost, the packet can be successfully received if there is a copy of it among the M redundant packets and that copy is delivered. Therefore, the expected fraction of successfully delivered packets in this case will be

$$(1-p)(1+\frac{M}{N}p) \tag{2}$$

While this strategy is not used in practice, it is the best that can be done with the same amount of redundancy used in the proposed scheme but without any coding, so it is included in this discussion to provide a fair comparison between coding and non-coding schemes.

Non-systematic, dense NC with redundancy consists on sending, for each set of N original packets, a set of $N + M$ coded packets. The coding coefficients are chosen such that in each possible combination of N coded packets out of the $N + M$ sent there are N linearly independent combinations of the original packets. In other words, the $N \times N$ coding coefficient matrix of the N selected packets is full rank. While this requirement can only be guaranteed with an algorithmic selection of the coding coefficients, a very good approximation can be obtained from RLNC provided that it is performed over a large enough field [3].

In order to compute the average fraction of successfully decoded packets, for each generation of $N+M$ coded packets, we have to consider two cases: (1) N or more coded packets are received, and (2) less than N coded packets are received. In the first case, all N original packets can be recovered, independently of exactly which coded packets were received—under the assumption of linear independence, the first N of the N or more received coded packets are enough to invert the coding matrix. If, however, less than N coded packets are received, there is not enough information to invert the coding matrix; therefore, no original packets from the generation can be decoded. In face of these considerations, we compute the expected fraction of successfully delivered packets as

$$\sum_{i=N}^{N+M} \binom{N+M}{i} (1-p)^i \, p^{N+M-i} \tag{3}$$

[1]With a slight abuse of nomenclature, we call a packet "linearly dependent" if it is linearly dependent from any set of $N-1$ other packets of the same generation.

Since $\begin{pmatrix} n \\ k \end{pmatrix} = \begin{pmatrix} n \\ n-k \end{pmatrix}$, by reordering the terms of the summation we conclude that this is the value of the cumulative binomial distribution function $\mathrm{B}(M; N+M, p)$.

The fact that whenever less than N coded packets are received (out of the $N+M$) no original packet from the generation can be recovered means that, while this strategy improves on the scenarios without NC for low values of the packet loss probability, for larger values of p its performance drops rapidly to levels well below those of even the non-coded scenarios.

It is also interesting to compute the asymptotical behavior of the code as N tends to infinity using a fixed amount of overhead (i.e., constant M/N). The strong law of large numbers [8] states that the sample average converges almost surely to the expected value; therefore, since the probability of receiving each packet is $(1-p)$, the number of received coded packets for the generation when the generation size tends to infinity is $(1-p)(N+M)$. All N original packets will be decoded and delivered to the application as long as this number equals or exceeds N, which happens for

$$p \leq \frac{M}{M+N} = \frac{r}{1+r} \tag{4}$$

where $r = M/N$ is the relative amount of added redundancy (overhead).

In this case, the expected fraction of delivered packets is 1. If, however, the network packet loss probability is larger, no packets can be decoded, and the expected fraction of delivered packets is 0. The expected fraction of delivered packets as a function of p is, therefore,

$$\begin{cases} 1 & \Leftarrow p \leq \frac{r}{1+r} \\ 0 & \Leftarrow p > \frac{r}{1+r} \end{cases} \tag{5}$$

Contrary to the naïf NC strategy described above, in our proposed SNC strategy an uncoded copy of each original packet is sent to the receivers. Therefore, the expected fraction of delivered packets can never be lower than that of the non-coded strategy without redundancy, and only for very high values of packet loss probability (that should not be found in properly dimensioned and managed networks) does it drop below that of the uncoded strategy with redundancy, as will be shown below. With such high values of packet loss probability, the fraction of packets delivered to the application with any strategy will probably be too low for the application to work properly, anyway.

In addition to the N uncoded packets, M additional packets are sent, containing linear combinations of the N original packets such that in each possible combination of N packets out of the $N+M$ sent there are N linearly independent combinations of the original packets. In the case of the first N packets (uncoded) this is trivially true, since the coding matrix is the identity matrix.

With this strategy, a packet is successfully delivered if (1) the uncoded copy of the packet is received or (2) the uncoded copy is not received but at least N out of the remaining $N+M-1$ packets are received. The expected fraction of successfully delivered packets can thus be computed as

$$(1-p) + p \sum_{i=N}^{N+M-1} \begin{pmatrix} N+M-1 \\ i \end{pmatrix} (1-p)^i p^{N+M-1-i} \tag{6}$$

$$= 1 + p\left(\mathrm{B}(M-1; N+M-1, p) - 1\right) \tag{7}$$

It is also possible to compute the asymptotical behavior of the code as N tends to infinity using a fixed amount of overhead. Notice that for very large N the summation in expression (6) is expression (3) with M off by 1. Thus, it converges to the same value as (3), and we conclude that the asymptotic value of the fraction of delivered packets using this code is

$$\begin{cases} 1 & \Leftarrow p < \frac{r}{1+r} \\ 1-p & \Leftarrow p \geq \frac{r}{1+r} \end{cases} \tag{8}$$

V. Numerical and Experimental Results

This section illustrates the behavior of the proposed coding scheme through numerical examples obtained from the expressions derived in the previous section and experimental results obtained in a prototype implementation of the system. The prototype was implemented using the Click modular router [9]. A packet dropper module was used to introduce losses by independently dropping packets with configurable probability p. Twenty repetitions of each experiment were performed, each using a number of packets 1000 times larger than the generation size.

In fig. 2 we plot the fraction of decoded packets against the packet loss probability using a generation size $N = 32$ packets with $M = 4$ redundant packets (when applicable) in the four strategies described in the previous section—the plain transmission of the original packets without coding or redundancy (NNCP), the transmission of the original packets with redundancy consisting on a second copy of selected packets (NNCR), a dense network code (DNC), and our proposed systematic code (SNC). The horizontal axis is in a logarithmic scale for increasing the resolution for lower values of p.

Due to the small amount of redundancy (12.5%), the NNCR curve is just slightly above the NNCP curve—if an original packet is lost, the probability of finding a copy of it among the redundants is low. DNC works well for low values of p; more than 0.999 of the packets are delivered up to p slightly above 0.02. However, at p about 0.057 it drops below the non-coded cases, and above $p = 0.36$ it becomes less than 0.001. This behavior is unacceptable for applications that degrade gracefully in the presence of packet losses. SNC, on the contrary, always performs better than DNC and NNCP, and only becomes (slightly) worse than NNCR for p above 0.18, a region where the packet delivery ratio is too low for most applications to work properly in any one of the strategies.

Figure 3 shows the asymptotic behavior of the different strategies with $N \to \infty$. For SNC, in addition to the asymptotic value (SNC-T) obtained numerically from expression (8), we also show the results of a practical experiment with a

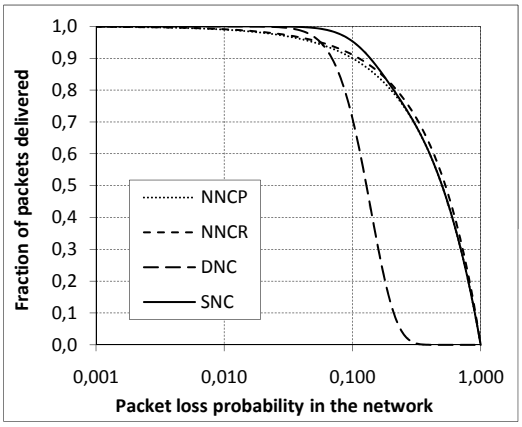

Figure 2. Numerical results with $N = 32$ and $M = 4$

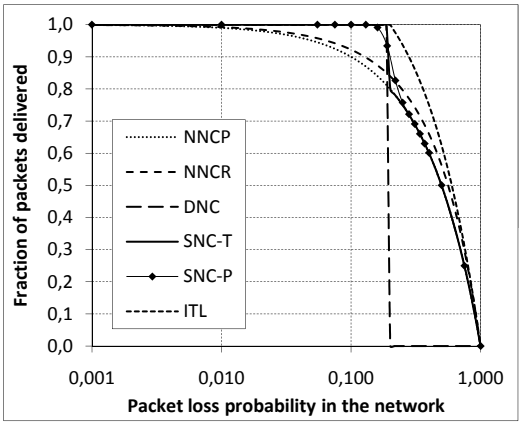

Figure 3. Results when $N \to \infty$ with $M/N = 0.25$

relatively large generation size $N = 200$ packets (SNC-P). The information-theoretical upper bound (ITL) of any code with 25% overhead is also shown, as a benchmark; this limit would be achieved using an omniscient source able to guess exactly which transmitted packets will be lost and retransmit them using an overhead up to M/N (and well approached by ARQ or a similar closed-loop technique with large enough timeout values).

Both SNC and DNC yield perfect decoding for values of p that the used overhead of 25% is enough to compensate for, as given by expression (4); however, above this value the performance of DNC drops to zero, while SNC drops to the same performance of NNCP. With a generation size $N = 200$, it is clearly visible that the experimental results of SNC already approach the asymptotic limit; with a network packet loss of 16%, the application sees less than 0.1% losses, and with a network packet loss of 13%, the application sees only about 0.01%.

In fig. 4 we plot the numerical results for SNC with a fixed generation size $N = 8$ and varying overhead (lines) along with experimental results obtained with our prototype implementation (marks). As expected, increasing the redundancy leads to

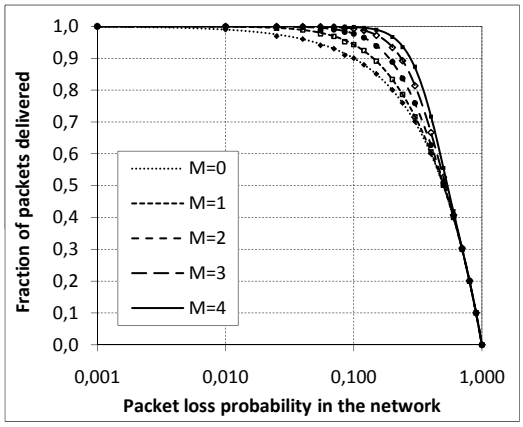

Figure 4. Results with $N = 8$ and varying overhead

higher packet delivery ratios for the application. The difference between the curves is more pronounced around the "knee" in the chart, which is the most interesting region, since for lower values of p uncorrected packet losses are negligible, and for higher values of p they rise rapidly to values that are too high for most applications. We can also observe in the figure a very good agreement between the theoretical prediction and the experimental results.

A perhaps surprising effect of SNC that we observed while plotting numerical results for varying generation sizes and fixed overhead is that while larger generation sizes increase the performance for lower values of p, smaller generation sizes exhibit better performance than larger ones for higher values of p. Larger generation sizes increasingly approach the asymptotical curve for $N \to \infty$ (fig. 3), which has a discontinuity, while smaller generation sizes have a smoother shape. To illustrate this effect, we performed an experiment with varying generation sizes but the constant overhead (M/N) for values of p around the "knee". The results of this experiment are shown in fig. 5, where the error bars represent the 95% confidence intervals for each point[2]; the lines correspond to the theoretical predictions. The fact that small values of N can improve the performance just to the right of the "knee" can be an additional benefit (besides a much lower processing load) of using small generation sizes for applications that can tolerate a larger amount of packet losses.

VI. Conclusion

Using the property of network coding that in order to decode a packet all the receiver needs is a sufficient number of linear combinations of packets (including that one), irrespective of exactly which combinations were received, we proposed a system for the distribution of content over multicast or broadcast media without feedback that provides adjustable resilience against packet losses, improving the quality of service seen by the applications. The basic principle of the system is sending, for each set of N original packets, a set of $N + M$ packets

[2]Confidence intervals are shown only in this zoomed chart since they are too tight to be visible in the others.

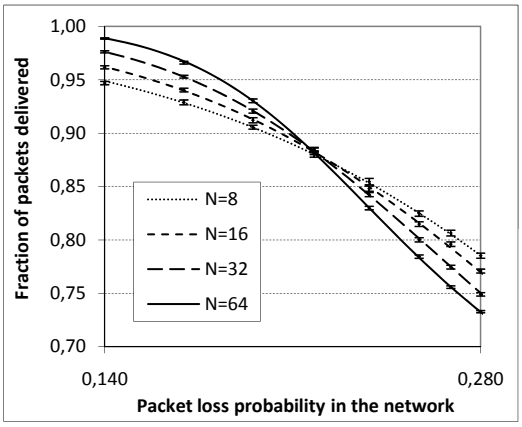

Figure 5. Results with varying generation size and fixed overhead $M/N = 0.25$

such that any N-sized subset of these contains N linearly independent combinations (w.h.p.). We proposed a systematic network code with much better erasure correcting properties than RLNC (which was not conceived for this application) and lower processing load. The performance of the code was evaluated both analytically and experimentally, and the results show that it can greatly reduce the packet loss seen by the applications for important ranges of the network packet loss probability.

Regarding generation sizes, the delivery probability advantages of large sizes should be balanced against the increased delay and the decoding complexity they impose. Moreover, and somewhat counterintuitively, larger generation sizes do not lead to better performance in all cases, and if the application can tolerate some losses, a small generation size can be a better choice. However, small generation sizes imply a coarser granularity on the usable overhead values.

In the near future, we want to optimize the implementation of the decoder, and then evaluate the processing load it imposes in different circumstances. On a longer term, we want to investigate alternative codes for concealing packet losses. Ultimately, we would like to answer the question "How close can we get to the information-theoretical limit with a code working without feedback and within practical operational constraints?"

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1204–1216, July 2000.

[2] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding Over Routing in a Randomized Setting," in *Proceedings of IEEE ISIT 2003*, p. 442, June 2003.

[3] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE Transactions on Information Theory*, vol. 52, pp. 4413–4430, Oct. 2006.

[4] M. Ghaderi, D. Towsley, and J. Kurose, "Reliability Gain of Network Coding in Lossy Wireless Networks," in *Proc. of IEEE INFOCOM 2008*, pp. 2171–2179, IEEE, Apr. 2008.

[5] J. Sundararajan, D. Shah, and M. Medard, "ARQ for Network Coding," in *Proc. of IEEE ISIT 2008*, pp. 1651–1655, July 2008.

[6] J. Barros, R. Costa, D. Munaretto, and J. Widmer, "Effective Delay Control in Online Network Coding," in *Proc. of IEEE INFOCOM 2009*, pp. 208–216, Apr. 2009.

[7] P. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in *Proc. 41st Allerton Conf. on Communication, Control and Computing*, vol. 41, pp. 40–49, Sept. 2003.

[8] M. Loève, *Probability Theory*, vol. I. Springer-Verlag, 4th ed., 1977.

[9] "Click Modular Router." <http://www.read.cs.ucla.edu/click/>.