A Family of Concatenated Network Codes for Improved Performance with Generations

Jean-Pierre Thibault, Wai-Yip Chan, and Shahram Yousefi

Abstract: **Random network coding can be viewed as a single block code applied to all source packets. To manage the concomitant high coding complexity, source packets can be partitioned into generations; block coding is then performed on each set. To reach a better performance-complexity tradeoff, we propose a novel concatenated network code which mixes generations while retaining the desirable properties of generation-based coding. Focusing on the code's erasure performance, we show that the probability of successfully decoding a generation on erasure channels can increase substantially for any erasure rate. Using both analysis (for small networks) and simulations (for larger networks), we show how the code's parameters can be tuned to extract best performance. As a result, the probability of failing to decode a generation is reduced by nearly one order of magnitude.**

Index Terms: **Concatenated codes, erasure channels, network coding (NC).**

I. INTRODUCTION

Random network coding (NC) [1], [2] is an alternative to packet routing which promises benefits such as increased throughput, decentralized operation, and erasure protection. The cost of these benefits lies in the required encoding and decoding. When the number of source packets is high, this cost can be significant; however, it may be controlled by sectioning the source packets into disjoint sets called *generations* [3]. Smaller generations yield smaller encoding and decoding costs. To capitalize on the forward erasure correction (FEC) properties of NC, redundant packets may be included with each coded generation. On lossy networks such as the Internet, this enables reliable transmission through either a pure FEC approach or a hybrid FEC/automatic repeat request (ARQ) approach.

Performing NC in this manner is analogous to linear block coding, with the randomness of NC being the most notable differentiator. Pursuing this analogy, generation size is simply a new name for block size. Since increasing a code's block size can yield a better code, we are faced with the classical block coding dilemma: Do we use smaller generations to obtain a low-complexity code, or do we use larger generations to obtain a code with better FEC properties? We instead turn to code concatenation, a well-known technique for increasing the power of a code [4]. We propose a novel generation-based NC scheme featuring two serially concatenated codes: An outer convolutional code and an inner block code. This new coding scheme subsumes the single-layered block coding scheme of Chou *et al*. [3].

The power of our code is conferred by the *mixed-generation* packets generated by the outer code. While Chou *et al*. [3] define generations as disjoint packet sets which are not allowed to intermix, we introduce here the novel concept of *limited* intergeneration coding, or "generation mixing." The main contribution of this paper is in showing that this new coding scheme offers considerably improved decoding performance on erasure channels while retaining the low complexity characteristics of traditional generation-based network coding (GBNC). The improvement in decoding performance can be taken as a reduction in either decoding failure probability, generation size (which in turn reduces encoding and decoding complexity), or a combination of the two. Moreover, given a proper choice of coding parameters, this improvement holds for channels with *any* erasure rate.

Some parameter tuning is required to extract the best possible performance from our proposed coding method; this echoes the observations of other recent research in network coding [5], [6]. The analysis and simulation of the code on various networks—from small to large and with various channel models—allows us to determine how the coding parameters can be tuned to extract superior performance from the code.

Our concatenated coding method is an enabler for efficient multicasting and broadcasting on lossy networks. It allows GBNC to operate more efficiently and does not require feedback. It can be used in large-scale multicasting or broadcasting applications, where the absence of feedback removes limits on the number of receivers that can be supported. The absence of feedback also allows generations to be more easily pipelined.

The paper is structured as follows. Section II begins with an overview of related work. Section III formally defines the structure of the new code as well as its encoding and decoding phases. Probability analysis is used in Section IV to show how concatenated coding can be advantageous and to study the effect of several coding parameters. In Section V, simulations are used to evaluate the code over larger network and different erasure models; Section VI concludes.

II. RELATED WORK

Random linear network coding originated in [2] with a lower bound on the probability of successful decoding; this opened the door to the practical use of network coding for many applications. Subsequently, Chou *et al*. [3] proposed generations as a way to overcome the high cost of encoding and decoding. By segmenting source packets into generations and coding only

J.-P. Thibault was with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada. He is now with Elliptic Semiconductor, Ottawa, ON K2K 2A9, Canada, email: jpthibault@ellipticsemi.com.

W.-Y. Chan and S. Yousefi are with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada, email: {geoffrey.chan, s.yousefi}@queensu.ca.

This work was supported in part by NSERC.

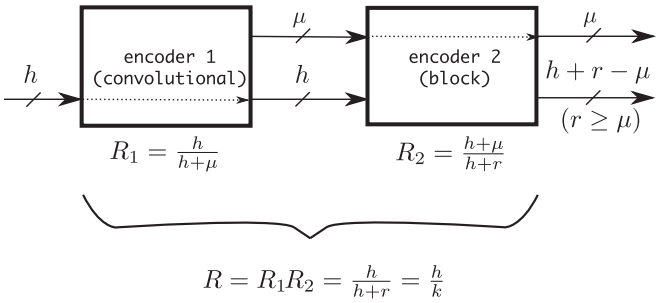

Fig. 1. Block diagram of the encoding phase.

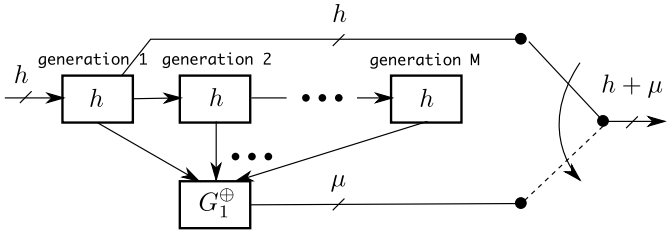

Fig. 2. Rate-$h/(h+\mu)$ systematic convolutional encoder.

within generations, cost can be greatly reduced; moreover, practical limits on the size of the data to be coded are eliminated. Encoding cost can also be mitigated with methods such as limiting the number of network nodes which must perform network coding [7]. However, generations remain the most effective way of managing both the encoding and decoding cost of large data sets; its concept has been pursued by many and given a variety of different names [8]–[11].

The repeated re-encoding of packets in network coding may lead it to be seen as a special form of concatenated coding, as each encoding node in the path from source to sink applies a successive layer of encoding to the source data. The analogy becomes stronger if a specific encoding is defined to leverage this successive re-encoding, as proposed by Kim [12] for specific network topologies. Our use of the term "concatenated" is closer to its traditional meaning in coding theory in the sense that in our coding scheme, concatenation occurs at a single node. This paper builds on the authors' initial study of concatenated network coding [13], [14] by delving into more realistic types of networks, with larger topologies and different channel models.

III. FRAMEWORK

We define a concatenated code where the outer convolutional code generates mixed-generation packets while the inner block code generates regular or unmixed packets. Let h be the number of source packets per generation, r the number of redundant packets per generation, μ the number of mixed packets per generation ($\mu \leq r$), and k the total number of coded packets per generation ($k = h + r$). The overall coding rate R is h/k. A general block diagram of the encoding phase for one generation is shown in Fig. 1.

Rate is allocated between the inner and outer codes via the parameter μ. For each generation of packets, the outer code generates μ redundant *mixed-generation* packets. Unlike regular coded packets, which are formed by combining packets from the same generation, these mixed packets are formed by combining packets from M successive generations. The inner code generates $(r-\mu)$ redundant unmixed packets; this yields a total of r redundant FEC packets per generation.

A. Encoding

The outer encoder is a systematic convolutional encoder of rate $h/(h+\mu)$ and memory length $M \geq 2$; it is denoted C_M. Its operation on a single generation of h input source packets is as follows. Since it is a systematic encoder, the h input packets are passed on, unmodified, to the inner code; μ new packets are formed by taking μ random linear combinations of the current generation input packets and of the $(M-1)$ previous generations' input packets. These μ packets are labeled "mixed-generation" packets and M is the number of generations which are mixed. The generations which are used to form a mixed packet are called the component generations for that packet.

The general structure for the C_M encoder is illustrated in Fig. 2. $G_1^{\oplus}$ is an $hM \times \mu$ encoding matrix which governs the formation of mixed-generation packets; its coefficients are randomly generated for each newly received input generation.

A restriction is imposed on $G_1^{\oplus}$'s structure to ensure that each mixed packet is formed from at least one packet from each of its component generations. If α_{ij} is the i,j-th entry of $G_1^{\oplus}$, the restriction may be expressed as:

$$\{\alpha_{ij}\}_{i=l}^{l+h-1} \neq \{0\}^h,\ l = 0, M, 2M, \cdots, (h-1)M \quad (1)$$

for each j from 1 to μ.

The inner code's encoder is a simple random linear block code of rate $(h+\mu)/(h+r)$: For each generation, it takes the h packets passed without modification by the outer code and creates $(k-\mu)$ random linear combinations of these over GF(q). The μ mixed packets created by the outer code are left unchanged; together with the $(k-\mu)$ unmixed packets created by the inner code, these form the k coded packets created by the encoding system and passed to the channel. The encoding matrix $G_2^{\oplus}$ for this coding stage therefore has the following structure:

$$G_2^{\oplus} = \begin{bmatrix} G_3^{\oplus} & \mathbf{0}_{h\times\mu} \\ \mathbf{0}_{\mu\times(k-\mu)} & I_\mu \end{bmatrix} \quad (2)$$

where $G_3^{\oplus}$ is a $h \times (k-\mu)$ random linear encoding matrix whose coefficients change for each input generation and I_μ is the $\mu \times \mu$ identity matrix. A similar restriction to that seen for $G_1^{\oplus}$ is imposed on the structure of $G_3^{\oplus}$; denoting its coefficients β_{ij}, we have:

$$\{\beta_{ij}\}_{i=1}^{h} \neq \{0\}^h \quad (3)$$

for each j from 1 to $k-\mu$.

Defining $S_i = [s_{i1}\ s_{i2}\ \cdots\ s_{ih}]$ to be the input source packets for generation i and $Y_i = [y_{i1}\ y_{i2}\ \cdots\ y_{ik}]$ the output coded packets for generation i, the encoding of packets for one generation may be succinctly expressed as follows:

$$Y_i = [S_i\ [S_i\ S_{i-1}\ , \cdots,\ S_{i-M+1}]G_1^{\oplus}\,]G_2^{\oplus}. \quad (4)$$

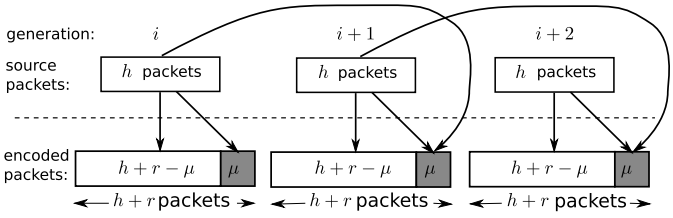

Fig. 3. Forming coded packets from source packets.

When transmitted, each packet is augmented with a vector of global encoding coefficients which describe its formation, as described by Chou *et al.* [3]. For an unmixed packet y_{ij}, this vector is column i of the encoding matrix $G_3^{\oplus}$, which represents hq bits of overhead. For a mixed packet, the coefficient vector is column i of the encoding matrix $G_1^{\oplus}$, which represents hqM bits of overhead. We will later see that this additional overhead for mixed packets is quite negligible, due to the small number of mixed packets which are generally required. To give a concrete example, consider the following representative case: $h = 40$, $r = 6$, $\mu = 2$, $M = 2$, and $q = 256$. Each non-mixed packet will have an overhead of 40 bytes while each mixed packet will have an overhead of 80 bytes. Mixed packets contribute only $(2 \times 80)/(2 \times 80 + 44 \times 40) = 8\%$ of the network coding overhead. Assuming 1500-byte packets, this is only 0.2% of the total data transmitted per generation. Since μ is always much smaller than h (as we will see in Sections IV and V), we are not concerned with the overhead of mixed packets.

The formation of coded packets by the concatenated code is illustrated in Fig. 3 for the $M = 2$ case. This figure shows mixed packets being sent after all unmixed packets; this minimizes decoding latency. Packet scheduling has no other effect on decoding performance. However, other practical considerations can dictate a different schedule; this is discussed in Section III-C.

B. Decoding

Decoding is done in the reverse order of encoding. For each generation, decoding via Gaussian elimination is first attempted by the inner decoder on the $(k - \mu)$ unmixed packets, using the encoding coefficient vectors embedded in these packets. If all h original source packets are recovered, no further decoding is required. A copy of recovered source packets is kept by the outer decoder as it may need them to decode future generations. If a rank deficiency is found, all received packets for this generation are passed on to the outer decoder, which attempts decoding using both mixed and unmixed packets.

In order to be useful, mixed packets must first be *unlocked*. A mixed packet is said to be *locked* if more than one of its component generations has not yet been decoded. Upon reception of a set of mixed packet for a given generation, the outer decoder can be in one of three states; for simplicity, we consider the $M = 2$ case, but the idea is readily extended to all M.

- **State 1:** If the previous generation was decoded, the decoder can eliminate the previous generation components of the mixed packets and use them as regular current-generation unmixed packet and re-attempt decoding. In this case, there is no gain (nor loss) from having mixed generations.
- **State 2:** If the previous generation was not decoded but the current generation is already decoded, the current generation components of the mixed packets may be eliminated to transform them into non-mixed packets of the previous generation. This allows a "second chance" of decoding the previous generation which would not have been possible without mixed packets; this is the advantage gained from the concatenated code.
- **State 3:** If neither the previous nor the current generation have been decoded, the mixed packet are currently of no use. They may become useful if the *next* generation is decoded: The mixed packets of that next generation could allow the current generation to be decoded, as described in the second scenario above. This would in turn unlock the generation's mixed packets, allowing them to be used towards decoding the *previous* generation. This "trickle-down" effect can last many generations; in the extreme, successful decoding of the very last generation received could trigger decoding of all previously undecodable generations.

The high-level interaction between these three states is illustrated by the diagram in Fig. 4. Note that this is not a complete state diagram for the decoder; such a diagram is presented in Section IV. The intent of Fig. 4 is to provide a simplified visual representation of the decoding process; it is not meant to be rigorous.

C. Transport Nodes

The above operational description is specific to source nodes. We now consider transport nodes: These are nodes which are neither sources nor destinations. In order to form mixed packets, transport nodes must buffer packets from the current and previous $(M - 1)$ received generations. A generation identifier is embedded in each packet—this is required by both sink nodes (for decoding) and transport nodes (for encoding). Transport nodes manage generations with the hard-flushing scheme defined in [3]: The arrival of a packet from a newer, previously unseen generation causes packets from the oldest buffered generation to be discarded. Hence, transport nodes only need to buffer Mh packets. The most recently received generation is said to be the *active* generation; this is the generation for which output coded packets are formed and sent.

If a transport node does not receive mixed packets, it forms coded packets (both mixed and unmixed) in exactly the same manner as a source node. If it does receive mixed packets, these can be added to each mixed packet generated by the outer code: A received mixed packet composed of generations x to $x-M+1$ is added (with a randomly chosen multiplicative coefficient) to each mixed packet which is created for that same set of component generations. This potentially increases the dimension of the vector space spanned by these mixed packets: From a rank-maximizing point of view, this is advantageous if the node's unmixed packets do not confer full rank for each of the M currently buffered generations. Since it can be expensive for a transport node to have knowledge of its rank, it is sensible to simply always combine the received mixed packets with the locally generated mixed packets; this is the method we employ.

When a transport node creates a coded packet, the new packet's vector of encoding coefficient is formed at the same

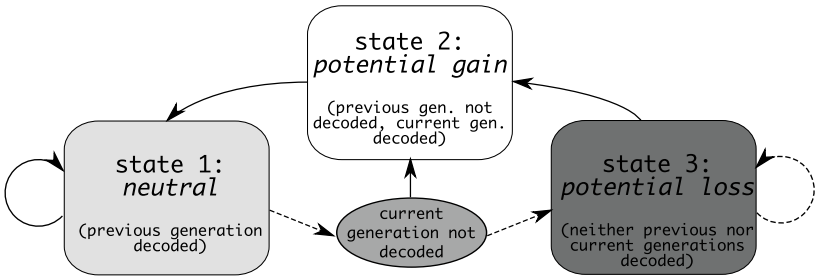

Fig. 4. A simplified depiction of the main decoder states with $M = 2$. Dashed transitions are taken when the current generation is not decoded. For concatenated coding to be beneficial, gains obtained in state 2 must offset the losses of state 3.

time. Just as the new packet is created with Galois Field additions and multiplications of existing encoded packets, the accompanying coefficient vector for the new packet is obtained by carrying out the same additions and multiplications on the coefficient vectors of the existing encoded packets used in forming the new packet. In this manner, the amount of encoding coefficient overhead is constant and does not increase with the number of encoding nodes in the network.

Transport nodes must decide when to send mixed packets and when to send unmixed packets. We propose a completely decentralized approach where each transport node simply aims to maintain the desired ratio of mixed and unmixed packets. For example, if $h = 70$, $r = 20$, and $\mu = 10$, transport nodes should send one mixed packet for every $(h + r)/\mu = 9$ packets sent. The reason for this approach is that while a source node knows exactly the number of packets that are sent for each generation and can therefore schedule the sending of mixed packets as it pleases to achieve the desired inner and outer code rates, we do not assume that transport nodes have such knowledge. The number of packets sent by a node for a given generation depends on the node's egress rate, as well as the erasures suffered on its incoming links. With our scheme, the total number of packets sent per generation no longer matters: Each node will receive, on average, the desired ratio of mixed and unmixed packets.

D. Evaluating Code Performance

To study the merits of the concatenated code, we analyze its performance when it is used on a network which suffers packet erasures and no feedback mechanism is provided to recover lost packets. Reliable transmission is therefore the full responsibility of the code. Our measure of performance is the probability of successfully decoding a generation, denoted P_{SD}, and the complementary decoding failure probability, P_{DF}. It should however be remembered that because of the relationship between generation size and decoding probability, any P_{SD} improvement can be replaced with a reduction in generation size, which in turn reduces encoding and decoding complexity.

The feedback-free framework is similar to the reduced-feedback framework defined by the authors in [11]. We emphasize that the utility of our code extends to other scenarios, including ones that employ feedback to dynamically adjust the code rate in response to packet losses or changing network conditions. The preclusion of feedback in this work simply serves to place the focus on the code's FEC performance.

E. Coding Parameters

For a given generation size h and code rate h/k, three key parameters affect the code's performance. First, the memory length M affects the number of mixed packets which can go towards decoding a generation. A larger M increases the number of mixed packets potentially available to the outer code to decode a generation left rank-deficient by the inner code: The number of such packets is $M\mu$. The disadvantage of a larger M is in the decreased probability of unlocking mixed packets: $(M - 1)$ of a mixed packet's component generations must be already decoded to enable the mixed packet to be used by the outer code on the remaining undecoded component generation. The selection of M therefore presents an important optimization problem.

Second, the number of mixed packets per generation μ controls the code rate of the inner and outer codes and presents a tradeoff similar to that seen with M. If a generation is not successfully decoded, a larger μ will increase the number of extra packets from the $(M - 1)$ following generations potentially available to help decode this currently undecoded generation. On the other hand, a larger μ will also increase the probability that future generations generation will not be decodable, which in turn limits the availability of the required mixed packets. Achieving good coding performance is therefore a joint optimization problem on the variables M and μ.

Third, we introduce a decoding latency limit L, defined to be the maximum decoding latency allowed, measured in generations. L limits the trickle-down decoding effect discussed in Section III-B. While M and μ must be carefully selected to obtain good code performance, the choice of L follows from strictly practical concerns: Increasing L always improves P_{SD}, but at the cost of higher decoding latency. We denote by $C_{M,L}$ the concatenated code with memory length M and decoding latency L.

There is an important link between L and M: Setting L to less than $(M-1)$ is unsound since doing so prevents $\mu(M-1-L)$ mixed packets from being *directly* applied to a rank-deficient generation by the outer code. Consider for example the $M = 3$, $L = 1$ case: If generation i is rank-deficient but generations $(i+1)$ and $(i+2)$ have full rank, the mixed packet of generation $(i+2)$ *could* be directly applied towards decoding generation i, but the decoding latency limit prevents this; those mixed packets are therefore effectively wasted. Here, reducing M to 2 such that the $L \geq M - 1$ condition is met would result in both better

code performance *and* lower complexity.

In comparing different $C_{M,L}$ codes, it is useful to consider three general cases for choosing L. In the first case, L is set to one less than the M for each code. Tying each code's decoding latency to its memory length is appropriate when code complexity is an overriding concern over decoding latency. Secondly, L may be set to one less than the highest M being considered. Having a common decoding latency for codes of different memory lengths is appropriate when decoding latency is the overriding concern over code complexity. Thirdly, when the best possible performance is of interest, no limit should be imposed on L, thus allowing boundless trickle-down decoding. In reality, the only limit on trickle-down decoding which remains is imposed by the finite number of generations used.

In the following, as our interest shifts to the concatenated code performance, we take $C_{M,L}$ to refer to the concatenated code instead of only the outer convolutional code.

IV. ANALYSIS

We first consider the code's performance on the 2-node line network, where the source and destination nodes are connected by a single link modeled as a binary erasure channel (BEC). This simple topology enables exact analysis of the probability of successfully decoding a generation with a single assumption: That coded packets are formed over a field of infinite size (to guarantee linear independence of packets sent by the source). The impact of this assumption is negligible if a large enough field size is used in practice. While only the source node performs network coding here, simulation results presented in the next section show that the performance of the code with this topology is a good indicator of its performance on larger topologies.

The dependence of the current generation's decodability on the previous generations' decoding status lends itself to a Markov chain analysis. Since the number of states required to capture the chain is 2^{L+1} and because values of M greater than $L+1$ are unsound, we choose $M = 2$ and $L = 2$ (simulation results will later show these to be generally good choices). We define eight states based on the decoded status of the current and previous two generations in Table 1. The corresponding state transition diagram is shown in Fig. 5. Shaded states indicate generation loss due to definite undecodability. State transitions occur after the reception of each generation. We adopt the notation p_{ij} to denote the transition probability from state i to state j, and π_i to denote the steady state probability of being in state i.

Lemma 1: The probability of successfully decoding a generation with the $C_{2,2}$ code is given by:

$$\begin{aligned} P_{\text{SD}}(C_{2,2}) &= \pi_1 + \pi_2 + \pi_3 + \pi_4 \\ &= [(1-p_{24})(p_{11} + p_{12}(2-p_{23})) + \\ &\quad p_{12}p_{24}(p_{41} - p_{47})]/(1 + p_{12} - p_{24}). \end{aligned} \quad (5)$$

Proof: The steady state probabilities $\{\pi_i\}_{i=1}^{8}$ are readily obtained using the ergodic property of the Markov chain. The state transition probabilities can be evaluated with proper conditioning, taking particular care with states $2,\ 4,\ 5$, and 8 as their transitions depend on sequences of events which occur across generations. Derivation of the p_{ij}'s is given in Appendix VI-A. □

Table 1. States for the Markov chain analysis of the $C_{2,2}$ code.

State	Decoding successful?			Generations lost
	Current generation	Previous generation	Second previous generation	
1	yes	yes	yes	0
2	no	yes	yes	0
3	yes	no	yes	0
4	no	no	yes	0
5	yes	yes	no	1
6	no	yes	no	1
7	yes	no	no	1
8	no	no	no	1

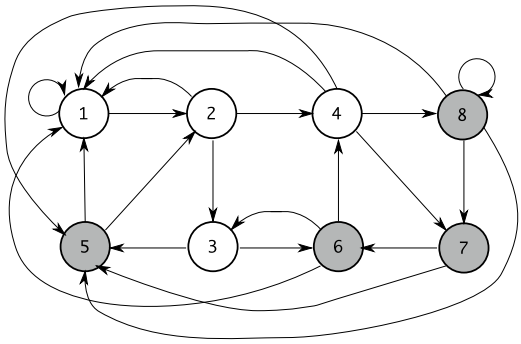

Fig. 5. State transition diagram for the $C_{2,2}$ code. Shaded states indicate generation loss.

This analysis also allows us to directly obtain $P_{\text{SD}}(C_{2,1})$, since reducing L from 2 to 1 does not change transition probabilities, only the states in which losses occur:

Corollary 1:

$$\begin{aligned} P_{\text{SD}}(C_{2,1}) &= \pi_1 + \pi_2 + \pi_5 + \pi_6 \\ &= \frac{(1-p_{24})(1+p_{12}p_{21}) + p_{12}p_{24}(p_{41}+p_{45})}{1 + p_{12} - p_{24}} \end{aligned} \quad (6)$$

To better understand the effect of M, we also wish to consider the $C_{3,2}$ and $C_{4,3}$ codes.

Corollary 2: The state definitions of the $C_{2,2}$ code in Table 1 are also valid for the analysis of the $C_{3,2}$ code, but the state transition diagram is slightly modified: The possible transitions from states $3,\ 4,\ 7$, and 8 differ from those seen with the $C_{2,2}$ and $C_{2,1}$ codes. This is illustrated in Fig. 6. We obtain the following:

$$P_{\text{SD}}(C_{3,2}) = \pi_1 + \pi_2 + \pi_3 + \pi_4. \quad (7)$$

Expressing $P_{\text{SD}}(C_{3,2})$ in terms of the p_{ij}'s is unwieldy. A detailed derivation for the π_i's and p_{ij}'s is given in Appendix VI-B.

Analysis for the $C_{4,3}$ code can be carried out in the same fashion but is omitted for brevity (a doubling of the number of states from 8 to 16 leads to very lengthy derivations). Note that setting $\mu = 0$ in (5), (6), or (7) yields P_{SD} for the simple block code, for which the value of M has no effect; this is useful to measure the merits of the concatenated code. All of the above results have been verified and validated by simulation.

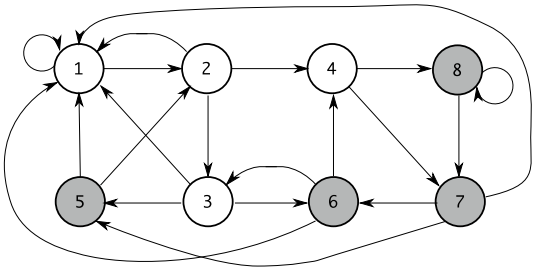

Fig. 6. State transition diagram for the $C_{3,2}$ code. Shaded states indicate generation loss.

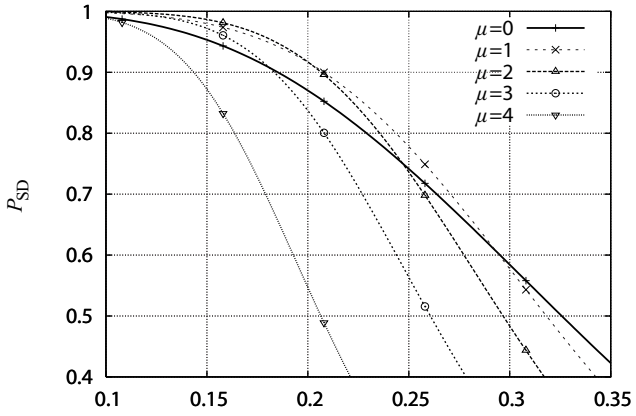

Fig. 7. Probability of successful decoding for the $C_{2,2}$ code on the 2-node line network with $h = 10$ and $r = 4$.

We can now evaluate and compare the performance of the $C_{2,1}$, $C_{2,2}$, $C_{3,2}$, and $C_{4,3}$ codes on the 2-node line network. We first focus on $C_{2,2}$ with the goal of examining the basic characteristics of this code's performance. Fig. 7 plots $P_{\text{SD}}(C_{2,2})$ as a function of ϵ with $h = 10$, $r = 4$, and $0 \leq \mu \leq 4$ (the $\mu = 0$ case corresponds to block coding only). This figure clearly shows that while an inappropriate choice of μ can have a disastrous effect on P_{SD}, our proposed code *can* achieve a significantly higher P_{SD} than the baseline block coding case. The $\mu = 1$ curve lies above the $\mu = 0$ curve for the widest ϵ range; however, $\mu = 2$ brings a further P_{SD} increase when ϵ is sufficiently small. This is characteristic of all concatenated codes considered here: The widest ϵ range for which a given $C_{M,L}$ code is superior to a block code is obtained with $\mu = 1$, with higher μ values yielding superior performance over narrower ϵ ranges. While Fig. 7 suggests that concatenated coding may only be useful for a specific range of ϵ, numerical analysis of (5) shows that concatenated coding can be advantageous for any ϵ, *as long as the code rate is such that the* P_{SD} *obtained with block coding only is sufficiently high*. In the case of the $C_{2,2}$ code, this P_{SD} threshold has been observed to be in the range of 0.6 for the $C_{2,2}$ code.

Taking this relatively low threshold as a preliminary indicator of code performance suggests our coding scheme to be beneficial for a wide variety of network coding frameworks (such as that of [11]). Section V presents simulation results which support the extension of these results to general network topologies. The effect of μ is as anticipated in the discussion of Section III-E: Increasing μ can be beneficial, but past a certain point, the inner code lacks sufficient erasure protection and consequently, code performance begins to deteriorate. These results show that the deterioration can be severe and that the tipping point depends on the P_{SD} at which we are operating.

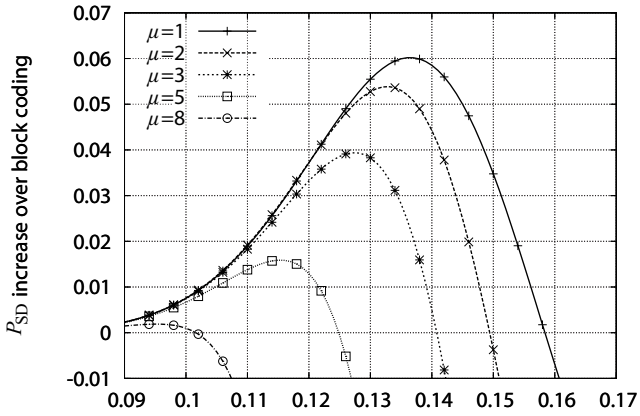

Fig. 9. Effect of μ on the P_{SD} increase over the block coding-only case for the $C_{4,3}$ code on the 2-node line network, with $h = 100$ and $r = 20$.

We next consider the effect of memory length by comparing $P_{\text{SD}}(C_{2,2})$ and $P_{\text{SD}}(C_{3,2})$. We take $h = 100$ and $r = 20$ to allow finer granularity on the relative code rates of the inner and outer codes. Rather than plotting P_{SD} directly, Fig. 8 plots the *difference* between the P_{SD} obtained with each concatenated code and that of the non-mixing block code, for various values of μ; positive values indicate that the concatenated code yields a higher P_{SD} than the block code. With this representation, several interesting points are brought to the fore. While roughly the same P_{SD} increase can be obtained with either $M = 3$ or $M = 2$, the two codes behave very differently with respect to μ. When $M = 2$, the left graph of Fig. 8 shows that each μ curve intersects at least one other curve; the choice of μ here must therefore depend on ϵ. In contrast, if we remove the $\mu = 1$ curve from the right graph of Fig. 8, all remaining curves are *strictly below* the $\mu = 2$ curve. For the $C_{2,2}$ code, the largest P_{SD} increase is obtained with $\mu = 3, 4$, or 5, depending on ϵ, but for the $C_{3,2}$ code, the largest increase is obtained with $\mu = 2$ for any $\epsilon < 0.14$.

Further increasing M to 4 (and L to 3, by necessity) in Fig. 9, the trends highlighted above continue. Here, $\mu = 1$ yields the highest P_{SD} increase for most values of ϵ, although $\mu = 2$ is very slightly superior when $\epsilon < 0.12$.

To better examine code performance when P_{SD} is high, the left graph of Fig. 10 plots the probability of decoding *failure* P_{DF} for the three concatenated codes considered above as well as for the block code. Here, we have reduced the stretch factor to $f = 1.12$ to capture the erasure region of interest; h and r are chosen to obtain good granularity for μ. For each code, μ is selected to maximize P_{SD} over the range of ϵ shown. Increasing M from 1 (block coding) to 2 brings a large decrease to the P_{DF}; further increasing M to 3 and 4 brings only a small additional decrease. These results agree with the discussion of Section III-E: Increasing M has both a positive effect (increasing the number of mixed packets available to decode a generation) and a

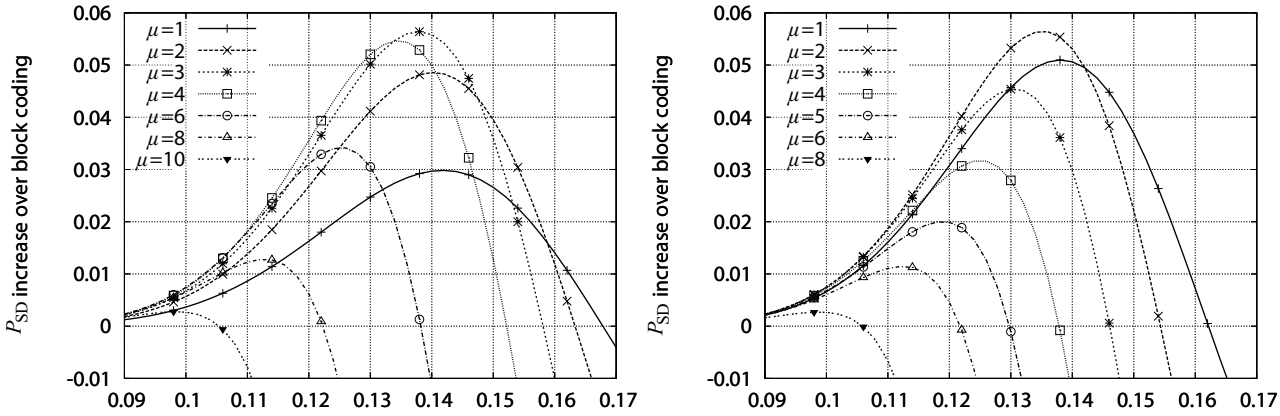

Fig. 8. Effect of μ on the P_{SD} increase over the block coding-only case for the $C_{2,2}$ (left) and $C_{3,2}$ (right) codes on the 2-node line network, with $h = 100$ and $r = 20$.

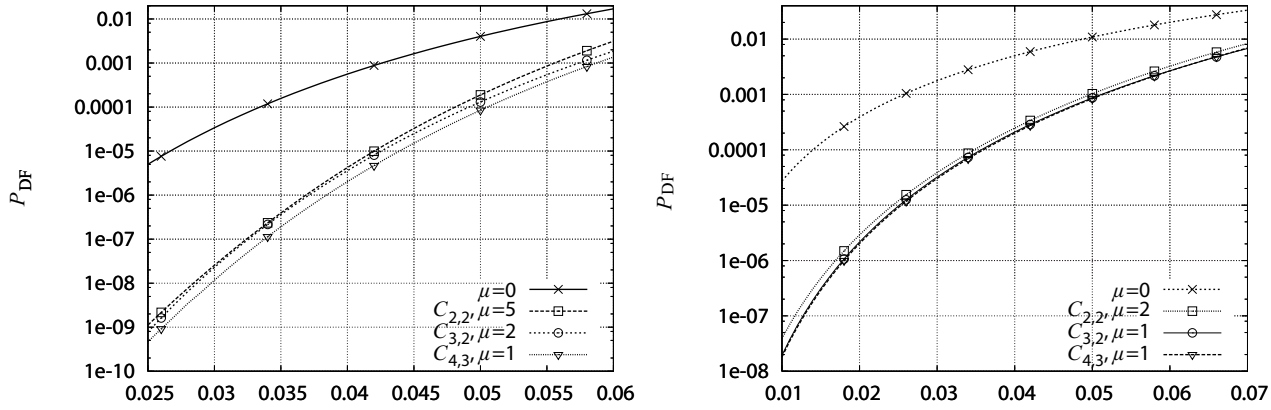

Fig. 10. Probability of decoding failure when P_{SD} is high on the 2-node line network, with $h = 100$, $r = 12$ (left), and $h = 15$, $r = 3$ (right). Improving code performance with higher M is dependent on the number of redundant packets per generation.

negative effect (decreasing the probability that mixed packets will be unlocked). It is therefore not surprising to observe that increasing M brings diminishing returns. More importantly, we note that the improvement in P_{DF} resulting from concatenated coding is substantial: At $P_{\text{DF}} = 10^{-5}$, the $C_{4,3}$ code allows ϵ to be increased by a factor of 1.6 when compared to the block code.

Consider also how Figs. 8 and 9 have shown that a smaller μ is preferable as M increases. Here, for $M = 4$, μ is very small (1) compared to r (12). Does the $C_{4,3}$ code perform as well when r is reduced? Since μ is restricted to integral values, can the smallest possible μ still be "too large"? To answer this question, we reduce r from 12 to 3 and h from 100 to 15 (while the actual stretch factor f is increased to 1.2, the range of ϵ and P_{DF} is roughly unchanged due to the greatly reduced generation size). The right graph of Fig. 10 shows that the decoding probabilities of $C_{4,3}$ and $C_{3,2}$ are nearly coincident under these parameters. This strongly suggests that in order to obtain the best decoding performance, an increase in M requires a decrease in μ. When μ can no longer be decreased, further increasing M may not be advantageous.

These results also suggest that codes with higher M require a larger stretch factor $f = (h + r)/h$ to perform well. To verify this hypothesis, Fig. 11 plots P_{SD} as a function of the stretch factor f, with a fixed $\epsilon = 0.1$. Note that the point at which the P_{SD} of each concatenated code surpasses that of the block code is indeed increasing with M as a function of f. Each concatenated code is operated with $\mu = 1$ since this is the μ which minimizes the f at which each concatenated code surpasses the block code.

V. SIMULATION RESULTS

The previous section having revealed some fundamental characteristics of our proposed code's performance on the simple 2-node line network with a BEC-modeled link, we now show that these characteristics hold on larger network topologies and more complex erasure models.

A. Methodology

Monte Carlo simulations are used to examine the effect of concatenated coding on larger network topologies. To strike a balance between simulation time and result accuracy, each simulation run consists of 20 generations. The first $(M - 1)$ generations of each run are not counted towards P_{SD} statistics since they cannot contain regularly-formed mixed packets. The last $(M - 1)$ generations of each run are also not included in the

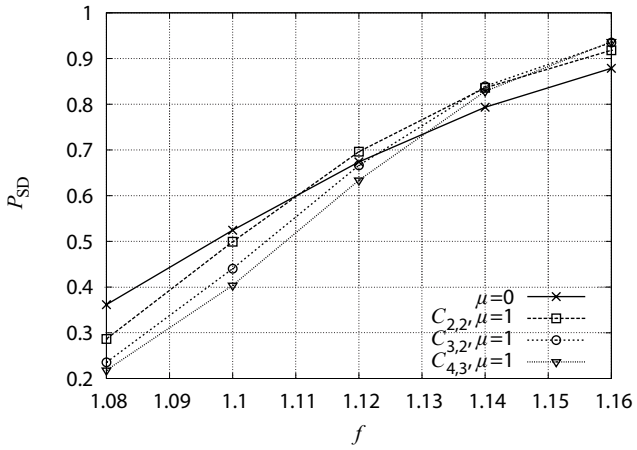

Fig. 11. Probability of successful decoding as a function of stretch factor on the 2-node line network, with $h = 50$ and $\epsilon = 0.1$.

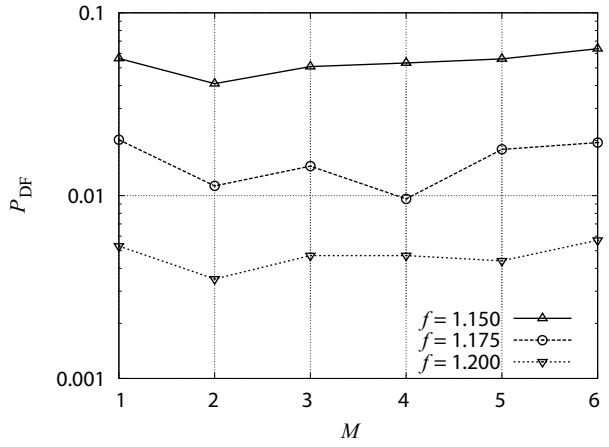

Fig. 12. The effect of memory length on the probability of decoding failure on the Rocketfuel network with BEC-modeled links, $h = 40$ and $\epsilon = 0.05$.

P_{SD} statistics since those generations do not have access to μM mixed packets as the other generations do. No limit on decoding latency is imposed beyond that which comes from the finite number of generations used; hence, if generations are sequentially numbered from 1 to 20, generation i is afforded a maximum decoding latency of $(20 - i)$ generations. In all cases, coding is performed over GF(256).

B. Topologies

To investigate the code performance on large topologies, an 87-node topology obtained from the Rocketfuel project [15] is used. Seven sinks equidistant from the source node and receiving the same flow rate are chosen; the results are averaged over these sinks. We also consider the simple 3-node line network, where the source and sink nodes are separated by a single transport node.

C. Results

We first consider the Rocketfuel topology with BEC-modeled links. Fig. 12 shows P_{DF} as a function of M with 40 packets per generation, $\epsilon = 0.05$ and three different stretch factors. Note that $M = 1$ represents the baseline block coding case. For each (M, f) pair, μ is empirically chosen to obtain the lowest P_{DF}. Although the chosen stretch factors give a large range of possible values for μ, best performance is always obtained with $\mu = 1$ or $\mu = 2$, thus confirming that the concatenated code works better with small μ values. On this topology, we conclude that the C_2 code is generally superior.

We next studied code performance at very low P_{DF}. Fig. 13 compares block coding with C_2, $\mu = 2$ and C_3, $\mu = 1$ (these μ values were chosen because they minimize P_{DF}) Similar performance to that seen analytically on the 2-node line network is observed here. For a constant $P_{DF} = 10^{-3}$, the C_2 code allows ϵ to be increased by a factor of 1.4 when compared to the block code. Fig. 13 suggests that more impressive gains would be obtained at lower P_{DF}, but computational resources are a limiting factor in obtaining such results.

Results were also obtained for the 3-node line network using the two-state Gilbert-Elliott channel (GEC) model. The GEC

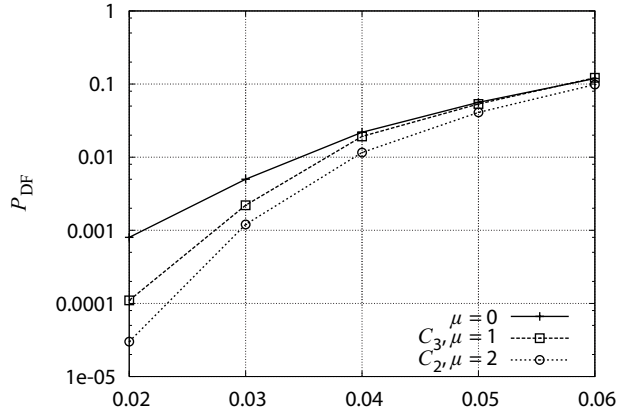

Fig. 13. Probability of decoding failure on the Rocketfuel network with BEC-modeled links, $h = 40$ and $f = 1.15$.

model is defined by the constant erasure probability associated with each state and the transition probabilities between these states. We define a "good" state with an erasure rate of $\epsilon_G = 0.01$ and a "bad" state with an erasure rate of $\epsilon_B = 0.13$. The transition probabilities from good to bad and from bad to good are $P_{GB} = 0.02$ and $P_{BG} = 0.10$ respectively; this yields an average erasure rate $\epsilon_{avg} = 0.05$.

Fig. 14 shows that with this scenario, concatenated coding yields large improvements: With $f = 1.25$, the C_2 code decreases P_{DF} by almost one order of magnitude. C_2 performs well for all stretch factors. As with the Rocketfuel network, $\mu = 1$ or 2 yields the smallest P_{DF} in all cases. The $f = 1.15$ curve appears to present a special case, with the lowest P_{DF} observed at $M = 5$. We believe that this is a consequence of approaching the operating region where concatenated coding is no longer beneficial—$f = 1.1$ is very close to this limit. Otherwise, Fig. 14 is remarkably similar to Fig. 12; in both cases, the best results are obtained with $M = 2$ and $\mu \leq 2$ when f is sufficiently high.

To further study the effect of the GEC channel on code performance, we consider three sets of GEC parameters which yield the same average erasure rate $\epsilon_{avg} = 0.03$ but which vary in

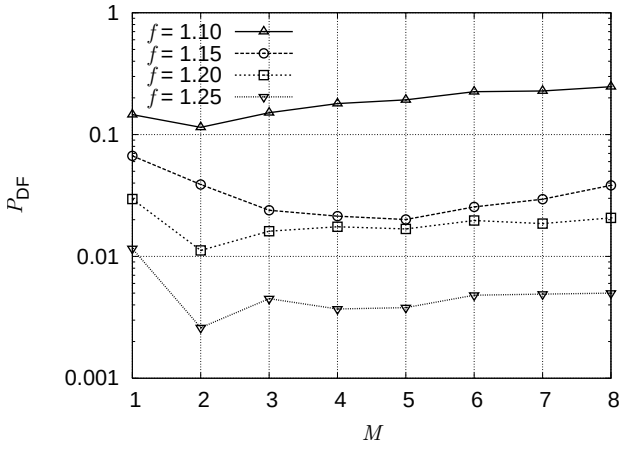

Fig. 14. The effect of memory length on the probability of decoding failure on the 3-node line network with GEC-modeled links, with $h = 20$ and $\epsilon_{\mathrm{avg}} = 0.05$.

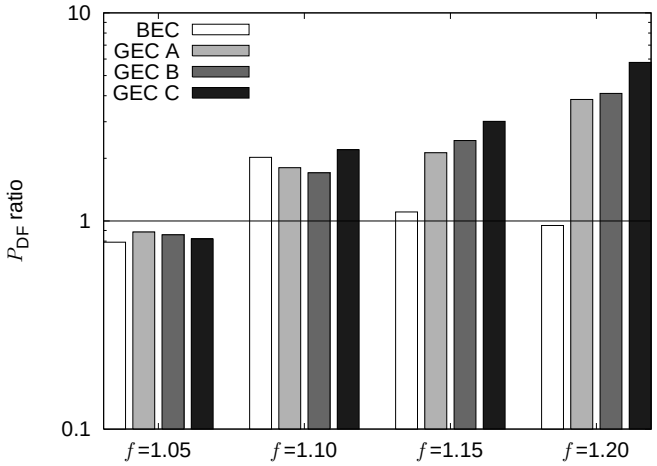

Fig. 15. Ratio of the P_{DF} obtained with block coding to the smallest P_{DF} obtained with C_2, for various channel models on the 3-node line network, with $h = 20$ and $\epsilon_{\mathrm{avg}} = 0.03$.

Table 2. Channel parameters used.

Scenario	P_{GB}	P_{BG}	ϵ_G	ϵ_B	ϵ_{avg}
GEC A	0.04	0.20	0.01	0.13	0.03
GEC B	0.05	0.25	0.01	0.13	0.03
GEC C	0.20	0.50	0.01	0.08	0.03
BEC	—	—	—	—	0.03

their proportion of time spent in the good and bad states. The average number of consecutive time units the channel remains in a particular state is a geometric random variable with mean $1/P_{\mathrm{GB}}$ and $1/P_{\mathrm{BG}}$ for the good and bad states respectively. We denote these consecutive sequences "bursts" and consider three scenarios which are differentiated by their average "good" burst lengths: (A) Bursts larger than the generation size, (B) bursts equal to the generation size, and (C) bursts smaller than the generation size. The GEC parameters for each scenario are given in Table 2. The resulting variance for the number of erasures per generation is therefore relatively large for GEC A, moderate for GEC B, and small for GEC C.

Fig. 15 shows the ratio of the P_{DF} obtained with the block code to the smallest P_{DF} obtained with the C_2 code for each stretch factor:

$$P_{\mathrm{DF}} \text{ ratio} = \frac{(P_{\mathrm{DF}} : \mu = 0)}{\min(P_{\mathrm{DF}} : \mu \geq 1)}. \tag{8}$$

The first observation is that with the GEC model, the benefits of concatenated coding increase with f. This is in accordance with the analytical results obtained in Section IV. With the GEC C model and $f = 1.20$, a nearly *six-fold* decrease in P_{DF} is obtained with the C_2 code and $\mu = 1$.

A second observation is that concatenated coding performs better when the channel state changes rapidly with respect to the generation size, i.e., with GEC C. If the time spent in the good state tends to be longer than the generation size (as with GEC A), the code's erasure protection is wasted; conversely, spending considerable amounts of time in the bad state results in too many erasures for the code and decoding fails. It is when the channel state changes rapidly that the flexible erasure protection provided by concatenated coding is most beneficial.

With the BEC model, the trend with respect to f is different: As f increases and P_{DF} approaches zero, the block and concatenated codes perform very nearly equally well; the benefits of concatenated coding diminish.

VI. CONCLUSION

We have defined a novel concatenated network coding scheme and shown it to offer performance superior to that of block network coding: For a given coding rate, concatenated coding increases the probability of successfully decoding a generation. Improvements of almost one order of magnitude have been obtained; these improvements allow the generation size to be reduced, which in turn reduces encoding and decoding complexity. This can enable the use of feedback-free GBNC for efficient large-scale multicasting and broadcasting applications.

Several parameters affect the performance of concatenated coding: The memory length, the decoding latency, and the inner and outer coding rates. In order to reduce P_{SD}, it is crucially important to properly select these parameters. Through analysis and simulation, we have shed light on the effect of the coding parameters. While larger M may increase decoding probability, most of the benefits of the concatenated coding approach are realized with $M = 2$. In general, best performance is obtained when μ is small relative to r; larger values of M require even smaller μ/r ratios. This renders the additional overhead required for the encoding coefficients of mixed packets negligible.

We have also shown that concatenated coding can be even more advantageous on bursty erasure channels, especially if the generation size is chosen with the channel characteristics in mind. This allows the flexibility of the effective stretch factor to be fully leveraged.

The good performance of our code with small values of M and μ means that the additional cost required for both encoding and decoding is small; the additional decoding latency can also be kept low. Our proposed decentralized mixed-packet scheduling scheme for transport nodes allows our code to be used as a direct drop-in replacement for conventional block-coding-like

network coding schemes; it can therefore be applied to a wide variety of network coding applications.

APPENDIX

A. $C_{2,2}$ and $C_{2,1}$ Analysis

Here we derive the transition probabilities and limiting probabilities for the Markov chain defined in Table 1 and Fig. 5 for the P_{SD} analysis of the $C_{2,2}$ and $C_{2,1}$ codes. We adopt the notation p_{xy} to denote the transition probability from state x to state y. Transitions occur after the reception of each generation. The transitions from state 1 are straightforward. Noting that up to r erasures can be tolerated, we condition on the number of erasures i to obtain:

$$p_{11} = \sum_{i=0}^{r} \epsilon^i (1-\epsilon)^{h+r-i} \binom{h+r}{i}. \tag{A.1}$$

While many of the other transition probabilities are straightforward, those emanating from states 2 and 4 are less so. Let us first consider p_{21}, which corresponds to the second scenario described in Section III-B; it may be evaluated in two parts. p_{21}^a is the probability that the current generation of state 1 is decoded without the use of mixed packets:

$$p_{21}^a = \sum_{i=0}^{r-\mu} \epsilon^i (1-\epsilon)^{h+r-\mu-i} \binom{h+r-\mu}{i}. \tag{A.2}$$

p_{21}^b is the probability that the previous generation (which had not been decoded on its first attempt) can now be decoded by using the current generation's mixed packets. The initial rank deficiency must be taken into account. Defining M_i to be the event that i mixed packets from the current generation are not erased and therefore applicable towards decoding the previous generation, R_{i-} to be the event that the previous generation was rank deficient by up to i packets, and R the event that the previous generation was rank deficient, we have:

$$\begin{aligned} p_{21}^b &= \sum_{i=1}^{\mu} P(M_i, R_{i-} \mid R) \\ &= \sum_{i=1}^{\mu} \frac{P(M_i)P(R_{i-})}{P(R)} \\ &= \frac{\displaystyle\sum_{i=1}^{\mu} \epsilon^{\mu-i}(1-\epsilon)^i \tbinom{\mu}{i} \sum_{j=1}^{i} \epsilon^{r+j}(1-\epsilon)^{h-j} \tbinom{h+r}{r+j}}{\displaystyle\sum_{j=r+1}^{h+r} \epsilon^j (1-\epsilon)^{h+r-j} \tbinom{h+r}{j}}. \end{aligned} \tag{A.3}$$

By the independence of these events, we obtain $p_{21} = p_{21}^a p_{21}^b$. p_{23} may be obtained in a similar fashion; we have $p_{23} = p_{21}^a p_{23}^b$, where p_{23}^b is only slightly different from p_{21}^b. Defining R_{i+} to be the event that the previous generation was rank deficient by more than i packets, we have:

$$\begin{aligned} p_{23}^b &= \sum_{i=0}^{\mu} \frac{P(M_i)P(R_{i+})}{P(R)} \\ &= \frac{\displaystyle\sum_{i=0}^{\mu} \epsilon^{\mu-i}(1-\epsilon)^i \tbinom{\mu}{i} \sum_{j=i+1}^{h} \epsilon^{r+j}(1-\epsilon)^{h-j} \tbinom{h+r}{r+j}}{\displaystyle\sum_{j=r+1}^{h+r} \epsilon^j (1-\epsilon)^{h+r-j} \tbinom{h+r}{j}}. \end{aligned} \tag{A.4}$$

The last transition out of state 2 is simply $p_{24} = 1 - p_{21}^a$.

The transition probabilities out of state 4 are similar to those out of state 2 but must account for the previous generation having not been decoded, which means that the current generation cannot use its mixed packets. There are three components to p_{41}: The next generation must be decoded (p_{41}^a), the previous generation must be decoded with a first trickle-down (p_{41}^b), and the second previous generation must also be decoded with a second trickle-down (p_{41}^c). p_{41}^a is equal to p_{21}^a and p_{41}^b is given as follows:

$$p_{41}^b = \frac{\displaystyle\sum_{i=1}^{\mu} \epsilon^{\mu-i}(1-\epsilon)^i \tbinom{\mu}{i} \sum_{j=1}^{i} \epsilon^{r+j-\mu}(1-\epsilon)^{h-j} \tbinom{h+r-\mu}{r+j-\mu}}{\displaystyle\sum_{j=r-\mu+1}^{h+r-\mu} \epsilon^j (1-\epsilon)^{h+r-\mu-j} \tbinom{h+r-\mu}{j}}. \tag{A.5}$$

Note that p_{41}^c is equal to p_{21}^b. In the same manner, p_{45} is easily obtained as $p_{45} = p_{21}^a p_{41}^b p_{23}^b$. Similarly, $p_{47} = p_{21}^a p_{47}^b$, where:

$$p_{47}^b = \frac{\displaystyle\sum_{i=0}^{\mu} \epsilon^{\mu-i}(1-\epsilon)^i \tbinom{\mu}{i} \sum_{j=i+1}^{h} \epsilon^{r-\mu+j}(1-\epsilon)^{h-j} \tbinom{h+r-\mu}{r+j-\mu}}{\displaystyle\sum_{j=r-\mu+1}^{h+r-\mu} \epsilon^j (1-\epsilon)^{h+r-\mu-j} \tbinom{h+r-\mu}{j}}. \tag{A.6}$$

Finally, $p_{48} = p_{24}$.

All other transition probabilities follow directly; in particular, $p_{35} = p_{11}$, $p_{36} = p_{12} = 1 - p_{11}$, and the transitions out of state i, $5 \le i \le 8$ are equal to the transitions out of state $(i-4)$. For easier notation, we define the following:

$$\begin{aligned} p_A &= p_{11} = p_{35} = p_{51} = p_{75}, \\ p_B &= p_{12} = p_{36} = p_{52} = p_{76}, \\ p_C &= p_{21} = p_{61}, \\ p_D &= p_{23} = p_{63}, \\ p_E &= p_{24} = p_{48} = p_{64} = p_{88}, \\ p_F &= p_{41} = p_{81}, \\ p_G &= p_{45} = p_{85}, \\ p_H &= p_{47} = p_{87}. \end{aligned} \tag{A.7}$$

Since the Markov chain is ergodic, its limiting probabilities π_i (where i denotes the state) are easily derived:

$$\begin{aligned} \pi_4 &= \frac{p_B p_E (1-p_E)}{1 + p_B - p_E} \\ \pi_1 &= \pi_4 \left(\frac{p_A - p_B(p_A p_D - p_C)}{p_B p_E} + \frac{p_F - p_A p_H}{1 - p_E} \right) \end{aligned}$$

$$\begin{aligned}
\pi_2 &= \pi_4\left(\frac{1-p_Bp_D}{p_E}-\frac{p_Bp_H}{1-p_E}\right)\\
\pi_3 &= \pi_4\frac{p_D}{p_E}\\
\pi_5 &= \pi_4\left(\frac{p_Ap_D}{p_E}+\frac{p_Ap_H+p_G}{1-p_E}\right)\\
\pi_6 &= \pi_4\left(\frac{p_Bp_D}{p_E}+\frac{p_Bp_H}{1-p_E}\right)\\
\pi_7 &= \pi_4\frac{p_H}{1-p_E}\\
\pi_8 &= \pi_4\frac{p_E}{1-p_E}.
\end{aligned} \tag{A.8}$$

From these, (5) and (6) are obtained.

B. $C_{3,2}$ Analysis

Here we derive the transition probabilities and limiting probabilities for the Markov chain defined in Table 1 and Fig. 6 for the P_{SD} analysis of the $C_{3,2}$ code. Given the similarities between the state transition diagrams for the $C_{2,2}$ code analyzed in Appendix VI-A and the $C_{3,2}$ code, it is not surprising to also find numerous commonalities in the transition probabilities of these two codes. It is easy to verify that the transition probabilities for states 1, 2, and 5 of the $C_{3,2}$ code are identical to those of the $C_{2,2}$ code.

The transitions from state 3 here differ from those of the $C_{2,2}$ code as it is now possible for the previously undecoded generation to get decoded. There is similarity between p_{21} and p_{31}, but the difference is that the previously undecoded generation has already had a chance at decoding from the outer code mixed packets (in state 2) and now receives a second chance, from a second set of mixed packets. Defining M_i^a to be the event that i mixed packets from the previous generation were not erased and made available to the second previous generation while in state 2, M_j^b to be the event that j mixed packets from the current generation were not erased and also made available to the second previous generation, $R_{(i+j)-}$ to be the event that the second previous generation was initially rank deficient by up to $(i+j)$ packets, we may obtain p_{31} as the product of p_{21}^a (as defined in (A.2)) and p_{31}^b:

$$\begin{aligned}
p_{31}^b &= \sum_{i=0}^{\mu}\sum_{j=0}^{\mu} P(M_i^a, M_j^b, R_{(i+j)-} \mid R)\\
&= \sum_{i=0}^{\mu}\sum_{j=0}^{\mu} \frac{P(M_i^a)P(M_j^b)P(R_{(i+j)-})}{P(R)}\\
&= \sum_{i=0}^{\mu}\sum_{j=0}^{\mu} \epsilon^{2\mu-i-j}(1-\epsilon)^{i+j}\binom{\mu}{i}\binom{\mu}{j}\times\\
&\qquad \frac{\sum_{k=1}^{i+j}\epsilon^{r+k}(1-\epsilon)^{h-k}\binom{h+r}{r+k}}{\sum_{k=r+1}^{h+r}\epsilon^{k}(1-\epsilon)^{h+r-k}\binom{h+r}{k}}.
\end{aligned} \tag{A.9}$$

The same considerations apply to p_{35}; defining $R_{(i+j)+}$ to be the event that the second previous generation was initially rank deficient by more than $(i+j)$ packets, we have $p_{35} = p_{21}^a p_{35}^b$, where:

$$\begin{aligned}
p_{35}^b &= \sum_{i=0}^{\mu}\sum_{j=0}^{\mu} \frac{P(M_i^a)P(M_j^b)P(R_{(i+j)+})}{P(R)}\\
&= \sum_{i=0}^{\mu}\sum_{j=0}^{\mu} \epsilon^{2\mu-i-j}(1-\epsilon)^{i+j}\binom{\mu}{i}\binom{\mu}{j}\times\\
&\qquad \frac{\sum_{k=i+j+1}^{h}\epsilon^{r+k}(1-\epsilon)^{h-k}\binom{h+r}{r+k}}{\sum_{k=r+1}^{h+r}\epsilon^{k}(1-\epsilon)^{h+r-k}\binom{h+r}{k}}.
\end{aligned} \tag{A.10}$$

The last transition out of state 3 is identical to the last transition out of state 2: $p_{36} = p_{24}$.

The transitions from state 4 are much simplified. Due to the increased interdependence of generations in mixed packets, previously undecoded generations have no chance of being decoded here; generation loss is assured. Either the current generation is decoded (without mixed packets), leading to state 7, or it is not, leading to state 8. We thus have $p_{47} = p_{21}^a$ and $p_{48} = 1 - p_{21}^a = p_{24}$.

While state 6 can transition to states 1, 3, or 4 as it did with the $C_{2,2}$ codes, the transition probabilities are slightly different, owing to the fact that the mixed packets of state 6's current generation are now locked (this is again due to the increased interdependence of generations in mixed packets). Consequently, p_{61} may be obtained as the product of p_{21}^a and p_{41}^b of the $C_{2,2}$ code (defined in Equation A.5). Similarly, p_{63} may be obtained as the product of p_{21}^a and p_{47}^b of the $C_{2,2}$ code (defined in (A.6)). Finally, p_{64} is identical to p_{24}.

The transition probabilities for states 7 and 8 are easily seen to be closely related to those of states 6 and 4, respectively. For easier notation, we define the following:

$$\begin{aligned}
p_A &= p_{11} = p_{51},\\
p_B &= p_{12} = p_{52},\\
p_C &= p_{21},\\
p_D &= p_{23},\\
p_E &= p_{24} = p_{36} = p_{48} = p_{64} = p_{76} = p_{88},\\
p_F &= p_{31},\\
p_G &= p_{35},\\
p_H &= p_{47} = p_{87},\\
p_I &= p_{61} = p_{71},\\
p_J &= p_{63} = p_{75}.
\end{aligned} \tag{A.11}$$

Proceeding as in Appendix VI-A, we obtain the following transition probabilities:

$$\begin{aligned}
\pi_1 &= \pi_7\frac{y-zp_B}{p_B},\\
\pi_2 &= \pi_7 y,\\
\pi_3 &= \pi_7\frac{xp_E}{1-xp_E},\\
\pi_4 &= \pi_7,\\
\pi_5 &= \pi_7 z,\\
\pi_6 &= \pi_7\frac{p_E}{1-xp_E},
\end{aligned} \tag{A.12}$$

$$\pi_7 = (\frac{y - zp_B}{p_B} + y + \frac{p_E(x+1)}{1 - xp_E} + 2 + z + \frac{p_E}{p_H})^{-1},$$
$$\pi_8 = \pi_7 \frac{p_E}{p_H}$$

where:

$$x = \frac{p_D - p_D p_E^2 + p_E^2 p_J}{p_E(p_E + p_D)},$$
$$y = \frac{1 - p_E(x + p_E)}{p_E(1 - xp_E)},$$
$$z = \frac{xp_G p_E}{1 - xp_E} + p_J.$$

Equation (7) may be evaluated from these results.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, pp. 1204–1216, July 2000.

[2] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE ISIT*, Yokohama, Japan, June 2003, p. 442.

[3] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Allerton Conf. Commun., Control, and Computing*, Monticello, IL, 2003.

[4] G. D. Forney Jr., *Concatenated Codes*. Cambridge, MA: MIT Press, 1966.

[5] M. Wang and B. Li, "How practical is network coding?," in *Proc. IEEE Int. Workshop on Quality of Service*, New Haven, CT, June 2006, pp. 274–278.

[6] M. Wang and B. Li, "Network coding in live peer-to-peer streaming," *IEEE Trans. Multimedia*, vol. 9, pp. 1554–1567, Oct. 2007.

[7] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2386–2397, June 2006.

[8] C. Gkantsidis, J. Miller, and P. Rodriguez, "Anatomy of a P2P content distribution system with network coding," in *Proc. Int. Workshop on Peer-to-Peer Systems*, Santa Barbara, CA, Feb. 2006, pp. 2235–2245.

[9] P. Maymounkov, N. J. A. Harvey, and D. S. Lun, "Methods for efficient network coding," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sept. 2006.

[10] D. Niu and B. Li, "On the resilience-complexity tradeoff of network coding in dynamic P2P networks," in *Proc. IEEE Int. Workshop on Quality of Service*, Chicago, IL, June 2007, pp. 47–55.

[11] J.-P. Thibault, S. Yousefi, and W.-Y. Chan, "Throughput performance of generation-based network coding," in *Proc. Canadian Workshop on Inf. Theory*, Edmonton, Alberta, June 2007, pp. 89–92.

[12] S. W. Kim, "Concatenated network coding for large-scale multi-hop wireless networks," in *Proc. Wireless Commun. and Network. Conf.*, Kowloon, China, Mar. 2007, pp. 985–989.

[13] J.-P. Thibault, W.-Y. Chan, and S. Yousefi, "Efficient mixed-generation concatenated network coding," in *Proc. the 24th Queen's Biennial Symp. Commun.*, Kingston, Canada, June 2008.

[14] J.-P. Thibault, "Throughput performance and complexity in generation-based network coding," Master's thesis, Queen's University, Kingston, Ontario, May 2007.

[15] "Rocketfuel: An ISP topology mapping engine." http://www.cs.washington.edu/research/networking/rocketfuel/.

Jean-Pierre Thibault received his B.A.Sc. degree from the University of Ottawa, Canada in 1999 and his M.Sc. (Eng.) degree from Queen's University, Canada in 2007, both in electrical engineering. He was a recipient of the Natural Sciences and Engineering Research Council (NSERC) Canada Graduate Scholarship in 2005. From 2000 to 2005, he was an IC Development and Verification Engineer with Zarlink Semiconductor. He is currently a Senior IC Designer and Architect with Elliptic Semiconductor, Ottawa, ON, Canada.

Wai-Yip Chan, also known as Geoffrey Chan, received his B.Eng. and M.Eng. degrees from Carleton University, Ottawa, and his Ph.D. degree from University of California, Santa Barbara, all in Electrical Engineering. Geoffrey is currently with the Department of Electrical and Computer Engineering, Queen's University, Canada. He has held positions with the Communications Research Centre Canada, Bell Northern Research (Nortel), McGill University, and Illinois Institute of Technology. His current research interests are multimedia coding and communications, and speech quality measurement and enhancement. He is an associate editor of EURASIP Journal on Audio, Speech, and Music Processing, and a member of the IEEE Signal Processing Society Speech and Language Technical Committee. He has helped organize conferences on speech coding, image processing, and communications. He held a CAREER Award from the U.S. National Science Foundation.

Shahram Yousefi received his B.Sc. degree in electrical engineering from University of Tehran, Tehran, Iran, in 1996, ranking second in his graduating class. In September 1997, he moved from industry to join the Department of Electrical and Computer Engineering at the University of Waterloo, Waterloo, ON, Canada, where he received the Ph.D. degree in electrical engineering in September 2002. He then moved to Kingston, ON, Canada, where he is currently an Assistant Professor in the Department of Electrical and Computer Engineering, Queen's University. His research areas of interest are in the general areas of communication and information theory, in particular, channel coding/decoding, performance evaluation of codes, and the application of graphical representations in decoding. He is the recipient of or has been nominated for more than 20 awards and scholarships including the Natural Sciences and Engineering Research Council of Canada award, the Sandford Fleming Foundation Award, and the Golden Apple as well as Frank Knox Awards at Queen's University.