To appear in the Proceedings of the IEEE Information Theory Workshop (ITW-08), Porto, Portugal, May, 2008

Relaying a Fountain code across multiple nodes

Ramakrishna Gummadi, R.S.Sreenivas
Coordinated Science Lab
University of Illinois at Urbana-Champaign
{gummadi2, rsree} @uiuc.edu

***Abstract*—Fountain codes are designed for erasure channels, and are particularly well suited for broadcast applications from a single source to its one hop receivers. In this context, the problem of designing a rateless code in the network case for on-the-fly recoding is very important, as relaying the data over multiple nodes is fundamentally useful in a network. Clearly, fountain codes are unsuited for on-line recoding (and simply forwarding over subsequent hops is provably suboptimal). Random linear codes are throughput optimal, but they do not enjoy the low complexity that is a prime feature of fountain codes. Can we get the low complexity of say, LT codes, while maintaining on-the-fly recoding and being throughput optimal?**

This paper proposes a novel solution to the above question. We consider packet level coding on a line network of discrete memoryless erasure channels (with potentially unlimited nodes), and exhibhit a coding scheme with (1) Ratelessness (2) Logarithmic per-symbol coding complexity (3) Throughput optimality (achieves rates equal the min cut capacity) and (4) avoids the delay of having to decode and then re-encode entire block lengths at intermediate nodes.

I. Introduction

Fountain codes are erasure coding schemes which are rateless, in the sense that they adapt to erasure channels with unknown parameters. Apart from ratelessness, a very low encoding/decoding complexity is another important feature they can afford. However, they are designed for a single erasure channel and are not applicable to a network - the only way to use them would be to completely decode and reencode at each intermediate node. While this strategy might be considered a "capacity achieving" scheme just counting channel uses, it is not asymptotically throughput optimal. Further, it is neither practical nor efficient due to the high delay and overhead involved in waiting for an entire block to be decoded before further encoding to the next node. As the blocklength (the number of packets encoded together) goes to infinity to drive the decoding error probability to zero, the throughput of decode and reencode scheme becomes arbitrarily bad.

II. Relevant Background

"LT Codes"([1]), are an erasure coding scheme for a single erasure channel with a per symbol logarithmic complexity. They operate by encoding a block of k input packets together. Each coded packet is a binary addition of a random subset of input packets. The number of packets added is called the degree of the output packet, and is chosen according to a cleverly designed probability distribution that leads to a simple decoding. It was shown (as $k \to \infty$) in [1] that a set of $k + O(\sqrt{k}\ln^2 \frac{k}{\delta})$ coded packets is sufficient to recover the k packets with a probability of at least $1-\delta$ through the simple belief propagation decoder. Here, δ is a positive parameter and both encoding and decoding have a complexity of $O(\log \frac{k}{\delta})$.

For an erasure code on a network, Random Linear Coding (RLC) is defined as uniformly chosen random combinations of the packets taken over a finite field. They are rate optimal because w.h.p, a random matrix tends to be full. However, the complexity of decoding is high due to the matrix inversion decoding. They have an average packet "degree" that is linear in k, implying an $O(k)$ complexity for encoding and decoding operations. It is to be noted that although matrix inversion itself takes a quadratic complexity, one should only estimate the complexity of the actual packet operations. Once the inverse matrix is computed, it takes an $O(k)$ complexity per decoded packet for the matrix multiplication. Hence, the complexity is linear *per symbol*, which is still much higher than the $O(\log k)$ achieved over a single channel.

There has also been work addressing this specific problem on a line network of erasure channels in [6], in which some solutions were discussed that trade off between complexity, delay and adaptability. *Delay* is defined as the additional time it takes for the coding scheme to complete the transfer of all packets *beyond* what it would have taken if the throughput was precisely the min-cut capacity. This delay is analyzed in [4] as *overhead*, which is defined as $n-k$ where n is the total number of received coded symbols(packets) and k is the number of encoded symbols. For a memoryless erasure channel, *delay* and *overhead* reflect each other barring an unknown constant (that depends on the erasure probabilities of the channel). Asymptotic throughput optimality in these cases should correspond to a sub-linear (in k) delay/overhead metric.

III. Contribution

Is it possible to have the low complexity of LT codes and low *delay* while maintaining *ratelessness* and achieving a throughput equal to capacity for anything beyond a single erasure channel? In this paper, we show that the above question can be answered positively, with two caveats: **(1) We consider networks that can be represented as a *tree* of DMC erasure channels of unknown erasure probabilities.** However from here on, we describe the scheme for a line network of erasure DMCs (as in [6]). Due to the ratelessness, it is easy to apply the same to a tree network wherein each node broadcasts packets to all of its children. This easy extensibility to tree networks is another advantage of the ratelessness. **(2) While**

the scheme is still rateless (assumes no estimate of the erasure probabilities and involves no feedback), it needs an estimate for some universal upper bound $0 \leq \alpha < 1$ on all erasure probabilities. A practical motivation for this scenario is a sequence of relay nodes with reasonably good, but unknown erasure channels that drop packets at rates anything between say, 0 - 10% of the time. Most networks do not have erasure channels that are arbitrarily bad, hence it would not be too unreasonable to assume such an α for design.

IV. Preliminary ideas

We approach this problem with an aim to use the low complexity of the LT codes, by devising methods for generating a similar code in the network case. The aim is to make the set of all coded packets that a node in the network ultimately receives to be *equivalent* to an LT code. This could potentially lead to a loss of independence between successively generated codewords, but since the erasures themselves are independent, it implies that a node successfully receives a uniformly random subset of the generated code words. In other words, we attempt to generate of a set of code words that is identical to an LT generated process even though the sequence of packets is not identical to an LT generated process. In this context, we have two basic issues to be tackled in order to use an fountain code in a network, as we discuss next.

A. Online encoding

Problem: Consider a set of k information packets to be encoded. We seek an algorithm to generate a set of k output coded packets that are identical to a set of LT generated coded packets with the following restriction: *The i^{th} coded packet being generated should be a combination of only the first i information packets.*

Discussion: It is easy to calculate the asymptotic fraction, f of coded packets which are formed out of solely the first i packets as a function of i given a degree distribution. It turns out that for any degree distribution, this is a convex function lying strictly below the line $f = i$. This has the discouraging implication that most of the code packets can be encoded only after we have access to most of the packets that are being coded.

However if we assume memoryless channels, the order in which we send the packets on the channel can be whatever we choose without changing the effectiveness of the code. Given a set of LT coded symbols on the indices $1 \ldots k$, an association of the denser parts of the hypergraph[1] of coded packets to the smaller indices and the sparser ones to larger indices of packets to be encoded could potentially give us a procedure for online encoding. We next describe a surprisingly simple solution that identifies a permutation, π of indices with the property that the first i indices in the permutation have a total of at least i coded packets with combinations from them alone. A similar idea has also been used earlier in generating systematic Raptor codes ([2]).

[1]As in the random hypergraph model of Darling and Norris [3], information packets can be interpreted as vertices of a hypergraph, where the hyperedges identify a coded packet.

Solution: If we relax our restriction by generating $k + o(k)$ LT coded packets rather than k, a probabilistic solution to this arises from the LT *decoding* process. Note that the scope of this toy problem is only to the extent that was posed above, although it serves as a building block to the actual coding scheme proposed later on. Consider the following algorithm for online encoding at a node:

Algorithm $SEQCODE$

(1) Generate a random set of $k + o(k)$ *symbols* according to the LT encoding process.
(2) Obtain a permutation π from the encoded symbols as follows:
  (a) Run an LT decoder on the encoded symbols.
  (b) If the index decoded at i^{th} iteration of the decoding process has a label j, then set $\pi(i) = j$.
(3) Now perform actual packet encoding using π as follows iteratively:
  (i) Pick a new coded symbol that includes index i, and involves all other indices of values less than i.
  (ii) Generate a coded output packet according to the rule given by the symbol obtained in step (i).

Proposition 1. *$SEQCODE$ generates a set of $k + o(k)$ packets which are LT distributed in an* online fashion*.*

Proof: Consider packets indexed $\pi(1) \ldots \pi(i)$. The fact that all these packets have been decoded by the i^{th} stage implies that there were at least i coded packets that were all combinations of $\pi(1) \ldots \pi(i)$. In fact, the decoding process runs to completion implies that each new index generates at least one new symbol to encode. This implies that step 3-(i) will be successful. ■

B. Recoding coded packets at intermediate nodes

Ignoring the real-time encoding aspect, one solution to the problem of recoding is to do concatenated coding. In other words, an intermediate node encodes packets that it receives treating them as information packets themselves. The decoding should peel off the successive coding layers and hence will involve t successive instances of the LT decoding process at a node which is t hops away from the source.

Consider a sequence of $n+1$ nodes with source node labeled as node 0 that starts off with k message packets (or a "block length", k). Fix a sequence of "block lengths", $k_0 \ldots k_n$ to be specified later, for the $n + 1$ nodes to perform the recoding). The lengths are defined with $k_0 = k$ and for each $1 \leq i \leq n$, k_i is set to ensure that collecting a total of k_i LT coded packets at node i is enough to recover the k_{i-1} packets that were recoded by node $i - 1$, with high probability. However, for this coding to be optimal, we need to ensure that the asymptotic *overhead* incurred at each node does not accumulate over hops.

Asymptotic Overhead: By appropriately scaling the block length k, in terms of the number of nodes, n, it is possible to

maintain cumulative rate optimality. For LT codes, a total of at least $k(1+\frac{\log^2\frac{k}{\delta}}{\sqrt{k}})$ packets gives an error probability of at most δ. On a line network of $n+1$ nodes for a union bound error probability of δ, fix a $\frac{\delta}{n}$ error at each intermediate node. For each i, we choose $k_{i+1}=k_i(1+\frac{\log^2\frac{k_i n}{\delta}}{\sqrt{k_i}})$ to ensure this, where $k_0=k$. Since $\frac{\log^2 k}{\sqrt{k}}$ is a decreasing function of k, we can upper bound k_i as

$$k_i \leq k_0(1+\frac{\log^2\frac{kn}{\delta}}{\sqrt{k}})^i \tag{1}$$

Finite nodes: Clearly, for $i \leq n$ and n constant, equation 1 implies $k_i \sim k$ as $k \to \infty$. The complexity of encoding at node i is the complexity of LT coding for blocklength $k_i(\sim k)$ which is $O(k\log k)$. Decoding complexity is $O(ik\log k) = O(k\log k)$ since it involves i successive instances of LT decoding.

Unlimited nodes($n \to \infty$): We now show that even for an arbitrary number of nodes, the overhead can be maintained asymptotically optimal. From equation 1, we have $k_n = k_0(1+\frac{\log^2\frac{kn}{\delta}}{\sqrt{k}})^n$. Setting $k=\Omega(n^3)$, it can be verified that this bound is asymptotically similar to $k=k_0$. Hence, we can still maintain a negligible overhead even with an arbitrarily growing number of nodes, provided we scale the block length appropriately. Since $k_n \sim k_0$, the encoding complexity is still the same as before - $O(\log k)$ per symbol. The decoding complexity now depends on the relation between n and k and is given as $O(n\log k)$ (per symbol) at the final node. Since, we require k to grow as at least $\Omega(n^3)$, this complexity is at most $O(k^{1/3}\log k)$ and can be further reduced by scaling k higher than n^3. This is again not restrictive, since block length usually is of much higher order compared to the number of nodes.

V. Complete scheme: "LT-relay"

Assume slotted time and a line network of DMC erasure channels as in [6] with i^{th} erasure probability ϵ_i. Intermediate nodes can send a packet that results from coding operations that involve all the packets that have been received until and including the current time slot. The illustrative toy problem considered in section IV-A only deals with an artificial situation of producing coded packets where exactly one new information packet is made available to the encoding node at each time slot. We need to devise a coding scheme when new packets are revealed based on a random process that represents erasures on the previous channel.

Fix a sequence of block lengths, $k_0 \ldots k_n$ as defined in Section IV-B ($k_0 \doteq k$). Node 0 generates standard LT coded packets. The operations performed by the intermediate nodes will be divided into two well defined distinct phases called the online phase - which is roughly defined as while the node is still receiving useful packets from its predecessor - , and the post-online phase. The main challenge is to construct a coding procedure for the online phase so as to ensure that the set of all packets generated during the online phase will be an optimal capacity achieving LT code. Beyond the online phase, the stream of coded packets can continue using standard LT coding. Note that the notion of a node's online phase does not involve its children, and hence is feedback independent.

Definition 1. ***Code Symbol*** *A code symbol is a set of indices, that correspond to packet indices which will be summed(exored) to form a coded packet. For instance, one can say that LT coding is performed using a set of* "code symbols" *generated according to a random LT distribution.*

Example 1 (Code Symbol)**.** *The set* $\{1,3,5\}$ *represents a code symbol that specifies the rule of summing the first, third and fifth packets to form a coded symbol.*

Definition 2. ***Online Code Copy*** An "Online Code Copy" at node i is an ordered sequence of k_{i+1} **code symbols** that can be generated in an online fashion.

Example 2 (Online Code Copy)**.** *Here is a procedure to generate a random instance of an* Online Code Copy*: Consider an instance/realization of a set of* k_{i+1} *(random) LT coded symbols generated from the* k_i *indices,* $1,2,\ldots,k_i$*. Using the solution to the online coding problem detailed in Section IV-A, one can, w.h.p., precisely compute a permutation* π *of* $\{1,2,\ldots,k_i\}$ *such that the number of code symbols containing exclusively the indices* $\pi(1),\pi(2),\ldots,\pi(i)$ *is at least* i*. (specifically, the procedure* $SEQCODE$ *). Now, consider the set of* Code Symbols *in the sequence in which they lend themselves to an online encoding (that provably exists because of the correctness proof of this construction,* $SEQCODE$ *). This set of code symbols together with the above ordering is an instance of an* Online Code Copy.

Definition 3. ***Code Matrix*** *A Code Matrix at node* i *is a* $T(k)\times k_{i+1}$ *random matrix of* Code Symbols *in which each row corresponds to an independently generated* Online Code Copy*. The number of rows of the code matrix,* $T(k)=k^{1+\delta}$ *where* $\delta>0$ *is a constant.*

We assume below that erasure probabilities are strictly increasing down any path from the source. However, it is possible to imagine schemes (that can be ratelessly implemented on top of the main scheme) to artificially force monotonically worsening channels with equivalent min-cut capacity. This leaves a sequence of effective channels of equivalent min cut capacity, $\min_i\{1-\epsilon_i\}$, with modified erasure probabilities, $\epsilon^{'}{}_i \approx \max_{1\leq j\leq i}\epsilon_i$. This is because, a good channel following a bad channel is redundant for the min-cut capacity. The details of how this can be accomplished are not as interesting here because of space constraints. The necessity for this should become clear in lemma 8.

Definition 4. ***Online phase*** *The online phase at node* i *is defined to be the period until the time slot at which node* i *collects a total of* k_{i+1} *coded packets. We further partition the online phase into* $k_{i+1}+1$ *states, indexed through* $0,1,\ldots,k_{i+1}$ *where the node is in state* i *when it has collected a total of* i *packets. Denote by* $\{s_0,s_1,s_2\ldots s_{k_{i+1}}\}$*, the random variables representing the number of erasures seen*

in the corresponding states. The number of time slots spent in state j is thus s_j+1 including the first time slot upon entering state j.

Remark 2. *Note the distinction that we define the Online phase to require collecting k_{i+1} packets, although node i needs only k_i packets to decode w.h.p. Also, node i uses only the first k_i packets it received for further encoding, even though the online phase state is maintained till k_{i+1}.*

A. The Coding Scheme:

We now describe the coding scheme at node i. It is implicit below that the coded packets received are indexed sequentially for the subsequent coding layer.

Procedure $LT-RELAY$ (node i)

1) First generate:
   (i) A random *code matrix*, $\mathcal{M}_i = [c_{ij}]_{1\le i\le T, 1\le j\le k_{i+1}}$
   (ii) Another independent random *online code copy*, $\mathcal{R}_i = \{\theta_j\}_{1\le j\le k_{i+1}}$
2) Initialize state to 0 and define the state as j, when j packets have been successfully received from node $i-1$.
3) *Online phase (i.e. while in state $j, 0\le j\le k_{i+1}$):*
   (i) In the first time slot upon entering state j, send a packet coded according to the code symbol θ_j.
   (ii) After the first time slot (i.e. for the remaining s_j slots): Choose a code symbol uniformly at random from $\hat{e}_j$, the j^{th} column of $\mathcal{M}_i$. Use it for encoding[2], unless it was used in an earlier time slot. Else, it becomes an *idle slot*.
4) Beyond the online phase, generate independent coded packets at each time slot using the standard LT coding procedure for a block length of k_i.

B. Correctness Analysis

We now develop the necessary arguments to show how the proposed solution works.

Lemma 3. *If n balls are thrown into T bins uniformly, the probability $\mathcal{C}(n,T)$ that there is at least one collision is upper bounded by $2n^2/T$.*

Proof: For $n > T/2$, the bound is trivial since $2n^2/T \ge 1$. Assume $n \le T/2$. It is easy to verify that $1-x \ge e^{-4x}$ for $0\le x\le 1/2$.

$$P(\text{No collision}) = \prod_{i=1}^{n-1}(1-i/T) \ge \prod_{i=1}^{n-1} e^{-4i/T} = e^{-2i(i-1)/T}$$

Hence, $P(\text{collision}) \le 1-e^{-2i(i-1)/T} \le 2i^2/T$

■

Theorem 4. *(w.h.p.) For any node the number of idle slots during encoding, N_i, satisfies $N_i \le \log k$*

Proof:

[2]Note that we have ensured that $\hat{e}_i$ contains only indices of at most i by construction.

Let I_j denote the number of idle slots in state j for $0 \le j \le k_{i+1}$, so that $N_i = \sum_{j=0}^{k_{i+1}} I_j$. Then,

$$\begin{aligned} P(N_i > \log k) &= P(\sum_{j=0}^{k_{i+1}} I_j > \log k) \\ &\le P\left(\{I_0 > \log k\} \bigcup \{\cup_{j=1}^{k_{i+1}}\{I_j > 0\}\}\right) \\ &\le P(I_0 > \log k) + k_{i+1}P(I_1 > 0) \\ &\le \alpha^{\log k} + 2kP(I_1 > 0) \end{aligned}$$

I_1 is the number of idle slots (duplications) observed when choosing *code symbols* uniformly from among $T(k)$ choices for a total of s_1 time slots. s_1 is the number of consecutive erasures in state 1 and hence is a Geometric random variable. Let $P(s_1 = j) = (1-q)q^j$ for $j \ge 0$ where $0 \le q < \alpha < 1$.

$$\begin{aligned} P(I_1 > 0) &= \sum_{j\ge 0} P(s_1 = j)\mathcal{C}(j, T(k)) \\ &\le \frac{2}{T(k)}\sum_{j\ge 0} j^2 P(s_1 = j) \text{ (from Lemma 3)} \\ &= \frac{2(1+q)}{1-q}\frac{1}{T(k)} < \frac{4}{1-\alpha}k^{-(1+\delta)} \\ \Rightarrow P(N_i > \log k) &< \alpha^{\log k} + \frac{8}{1-\alpha}k^{-\delta} \\ &= o(k^{-\beta}) \text{ for some } \beta > 0 \end{aligned}$$

■

Definition 5. *Let $\Lambda = [\lambda_{ab}]_{1\le a\le T(k), 1\le b\le k}$ be a random matrix with elements taking $\{0,1\}$ values be defined as follows. Each element λ_{ab} is a Bernoulli random variable defined as:*

$$\lambda_{ab} = \begin{cases} 1 & \text{, if } c_{ab} \text{ was used for encoding} \\ 0 & \text{, otherwise} \end{cases} \quad (2)$$

Theorem 5. *Given any integer c, and arbitrary $S \subset \{1,2,\ldots,T(k)\}\times\{1,2,\ldots k\}$ such that*

$$|\{x : (x,j)\in S\}| \le c \qquad (1\le j\le k)\ ,$$

the Random Variables, $\{\lambda_{ab}\}_{(a,b)\in S}$ are mutually independent, and identically distributed as $k\to\infty$.

Proof: First, note that $b_1 \ne b_2 \Rightarrow \lambda_{a_1b_1}$ and $\lambda_{a_2b_2}$ are independent. Hence, without loss of generality we only need to show that $\lambda_{a_1b}, \lambda_{a_2b}\ldots,\lambda_{a_cb}$ are independent where the a_j are all distinct. Denote for ease of notation, $\beta_j = \lambda_{a_jb}$ for $1\le j\le c$. Since β_j are binary valued, we only need to show that the events $\{\beta_j = 0\}_{j=1}^c$ are independent.

Let $P(s_b = i) = (1-q)q^i$ for $i \ge 0$. For any $1\le j\le c$

$$\begin{aligned} P(\beta_j = 0) &= \sum_{i\ge 0} P(s_b = i)P(\text{ All } i \text{ slots missed } c_{a_jb}) \\ &= \sum_{i\ge 0}(1-q)q^i(1-1/T)^i = \frac{1-q}{1-q+q/T} \end{aligned}$$

Consider an arbitrary subset of these events $\{\beta_{k_j} = 0\}_{1 \le j \le l}$ where $l \le c$.

$$P\left(\bigcap_{j=1}^{l}\{\beta_{k_j}=0\}\right)$$
$$= \sum_{i \ge 0} P(s_b = i) P(\text{ All } i \text{ slots missed all } c_{a_{k_j} b} \text{ for } 1 \le j \le l)$$
$$= \sum_{i \ge 0} (1-q) q^i (1 - l/T)^i = \frac{1-q}{1-q+ql/T}$$

As $k \to \infty$ (and hence, $T(k) \to \infty$),

$$(P(\beta_j = 0))^l \approx \frac{(1-q)^l}{(1-q)^l + l(1-q)^{l-1} q/T}$$
$$= \frac{1-q}{1-q+ql/T} = P\left(\bigcap_{j=1}^{l}\{\beta_{k_j}=0\}\right)$$

■

The above theorem implies the following corollary, which says that the code symbols chosen by the algorithm are scattered "uniformly random" in the following sense:

Corollary 6. *Let $\chi = \{0,1\}^{T(k) \times k}$ denote the ensemble of all possible realizations of the random matrix, Λ. For $\Psi = [\psi_{ij}] \in \chi$ and for any $S \subset \{1, \ldots, T(k)\} \times \{1, \ldots, k\}$, denote*

$$W_S(\Psi) = \sum_{(i,j) \in S} \psi_{ij}$$

Consider any given integer r, and $E \subset \{1, \ldots, T(k)\} \times \{1, \ldots, k\}$, with $|E| = r$ denoted as $E = \{e_1, \ldots, e_r\}$. For any $\phi = (\phi_1, \ldots, \phi_r) \in \{0,1\}^r$ let $\Theta_\phi = \{\Psi \in \chi : \psi_{e_j} = \phi_j$ for $1 \le j \le r\}$. Then, as $k \to \infty$, in the probability space generated by "LT-Relay", $P(\Theta_\phi)$ depends solely on $\sum_{i=0}^{r} \phi_i = W_E(\Psi)\ \forall\ \Psi \in \Theta_\phi$.

Note that, if s_j, number of time slots spent in column j, had been Poisson rather than Geometric, we could have had perfect independence due to the Poisson splitting property. Although this is not quite the case here, we do have a reasonable notion of "asymptotic uniformity" as argued by Corollary 6.

Lemma 7. *Given that (i) the subset of* code symbols *from $\mathcal{M}_i$ used by the algorithm at node i, is uniformly random and (ii) t denotes a time slot past the online phase, the* set *of all coded packets generated by node i till time slot t forms an LT code.*

Proof: Under the above assumptions, the set of code symbols used till time t is the union of

1) $\mathcal{R}_i$
2) An (almost) uniform random subset of code symbols from $\mathcal{M}_i$ (partially justified by Corollary 6)
3) The independent LT coded packets generated past the online phase.

Clearly, the components (1), (2) and (3) are mutually independent by their construction. Also, each of them is an LT coded set of packets - (1) by construction, (2) because of the fact that a uniformly random subset of a set of independent identically distributed(iid) RV's is also iid as the original set. In our case, $\mathcal{M}_i$ is the original set of LT coded random variables. Also, (3) is an LT coded set. Hence, their union is a set of LT coded packets. ■

Lemma 8. *(w.h.p.) For each i, the **first** k_i packets collected at node i can be decoded to recover the k_{i-1} packets that were recoded by node $i-1$. Applying this recursively, the original k_0 packets can be decoded.*

Proof: We will argue that the first k_i packets collected are equivalent to an LT code. The (online) state of a node i counts the number of successfully received packets from node $i-1$. Because the effective channels are ensured to be progressively worse, the *online state* values are monotonically decreasing eventually, w.h.p. Hence, when node i collects k_i packets, node $i-1$ has already exited its online phase w.h.p. (which had only k_i states) These k_i packets collected are a uniform random subset of the set of all packets sent out by node $i-1$ till a time that is past its online phase, which forms an LT code instance by Lemma 7. Hence, they can be decoded. ■

Theorem 9. *The code described is capacity achieving. That is, packets are transmitted from the source to the node i at a rate equal to $\min_{1 \le j \le i}(1-\epsilon_j)$.*

Proof: For recovering the original k packets, the number of coded packets node i needs to collect is $O(k)$ (Lemma 8 and the fact that $k_i = O(k)$ for any i). Also, Coded packets are being sent by node $i-1$ to node i at *every* time slot except for the the vanishing fraction of $O(\log k)$ idle slots (Theorem 4) out of a total number of at least $\Omega(k)$ time slots. Hence we have a throughput rate to node i equal to the success rate on the channel between nodes $i-1$ to i, $\min_{1 \le j \le i}(1-\epsilon_j)$. ■

VI. Conclusion

A throughput optimal rateless coding scheme to relay LT codes across multiple nodes was described. Its complexity of *packet operations* is close to that of the single channel case.

VII. Acknowledgements

The first author would like to thank Professor Bruce Hajek for discussions related to the problem, mainly on the online encoding aspect. We would like to thank an anonymous reviewer for alerting us to systematic raptor codes.

References

[1] M.Luby, "LT Codes", Proc. FOCS 2002.
[2] A. Shokrollahi, "Raptor Codes", IEEE Transactions on Information Theory, vol. 52, num. 6 (2006), p. 2551-2567 2006
[3] R. Darling, J. R. Norris, "Structure of large random hypergraphs", Annals of Applied Probability 2005, Vol. 15, No. 1A, 125-152.
[4] Nicholas Harvey, Desmond Lun and Petar Maymounkov, "Methods for Efficient Network Coding", Proc. Allerton 2006.
[5] D. S. Lun, M. Medard, and M. Effros, "On coding for reliable communication over packet networks", Proc. Allerton 2004.
[6] P. Pakzad, C. Fragouli and A. Shokrollahi, "Coding schemes for line networks", Proc. ISIT 2005.