Coding Schemes for Line Networks

Payam Pakzad, Christina Fragouli and Amin Shokrollahi
Laboratoire d'algorithmique (ALGO)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{payam.pakzad, christina.fragouli, amin.shokrollahi}@epfl.ch

Abstract— **We consider a simple network, where a source and destination node are connected with a line of erasure channels. It is well known that in order to achieve the min-cut capacity, the intermediate nodes are required to process the information. We propose coding schemes for this setting, and discuss each scheme in terms of complexity, delay, achievable rate, memory requirement, and adaptability to unknown channel parameters. We also briefly discuss how these schemes can be extended to more general networks.**

I. Introduction

Networked systems arise in various contexts such as the public internet peer-to-peer networks, ad-hoc wireless networks, and sensor networks. Such systems are becoming central to our everyday life. The networked systems today employ traditional coding schemes for end-to-end connections and are generally not tailored to the network environment. For example, for reasons of design simplicity, intermediate nodes at a network are only allowed to forward and not to process incoming information flows. However, as the size of communication networks grows, it becomes less clear if the benefits of the simple end-to-end approach outweigh those of coding schemes that employ intermediate node processing.

From a theoretical point of view it is well-known that if intermediate nodes are allowed to decode and re-encode the information sent by the source, –with no constraints on complexity and/or delay,– then the information capacity between a sender and a receiver is upper bounded by the *min-cut capacity* of the network, as described in [2]. A crucial point in making schemes that employ intermediate node-processing practical and attractive, is in realizing benefits without incurring excessive complexity and delay.

In this paper we propose coding schemes that employ intermediate node processing and discuss their performance. These schemes are based on *fountain codes*, a set of rateless codes recently proposed [4], [7] that have a number of desirable properties for networked environments. We compare different coding schemes based on their *complexity*, *delay*, *memory requirement*, *achievable rate*, and *adaptability*; we will define these metrics precisely in Section II.

For example, if we use an LT-code [4] to encode k information bits at the source and simply forward any received bit at the intermediate nodes, we would need $O(k\log(k)/C)$ XOR operations at the transmitter, and $O(k\log(k))$ XOR operations at the receiver, where C is the end-to-end capacity of the overall channel measured in bits per channel use. Intermediate nodes would have no processing or memory requirements, and would not introduce delay. This scheme would further adapt to unknown channel parameters. However, the achievable rate can only approach the end-to-end capacity of the overall channel, which is in general less than the min-cut capacity of the network.

In [5] the authors examined the benefits of intermediate node processing from an information theoretic point of view. Our work can be viewed as approaching the same problem from a coding theory point of view.

In [8] a scheme was proposed that takes advantage of intermediate node processing to approach the min-cut capacity, and puts emphasis on the queuing theory aspects of the problem. The authors show that if we allow intermediate nodes to transmit random linear combinations of the incoming packets over a finite field $\mathrm{GF}(q)$, the transmission rate approaches the min-cut capacity as q goes to infinity. In this paper we will present alternative optimal coding schemes that approach the min-cut capacity using a constant field size, and in particular a binary field.

The paper is organized as follows. In Section II we present our model and performance metrics in more detail. In Section III we describe our proposed coding schemes. In Section IV we discuss generalization to other networks; In Section V we compare our results with some related work in more details, and finally we conclude the paper in Section VI.

II. Model

We consider a linear network that models a path between a source and a destination. The corresponding graph is comprised of a source node, a destination node and a series of $L-1$ intermediate nodes. The L edges between the nodes correspond to independent memoryless erasure channels, and the information units sent over the ith link are erased with probability ϵ_i.

We assume a discrete time model, where each node can transmit one unit of information at each time slot. For coding purposes, we will treat each information unit as a symbol, but in general we can have a packet of symbols, and apply to each symbol of the packet the same encoding/decoding operation; in the following, we will refer to information units as packets or symbols interchangeably. Intermediate nodes have the capability to process the packets they receive, and use them to generate new packets. We ignore the transmission delay along channels (as it is beyond our control), i.e., we

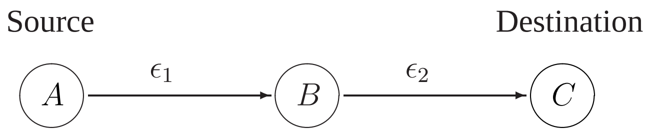

Fig. 1. A path between a source A and a receiver C with L=2 links.

assume that a packet transmitted at time d, if not erased, is received immediately at the next node in the chain.

Throughout this paper we will use as illustrating example the simple configuration depicted in Fig. 1 with $L = 2$ links; we will also discuss the generalization of our results to longer chains. The source node A encodes k symbols to create n_1 coded outputs using a code $\mathbf{C}_1$ and sends them over the channel AB. Node B will receive on average $n_1(1-\epsilon_1)$ coded symbols over n_1 time slots. Node B will send n_2 packets, using a code (more generally, processing) $\mathbf{C}_2$. If node B finishes transmitting at time d, where $\max\{n_1, n_2\} \leq d \leq n_1 + n_2$, then node C will receive on average $n_2(1 - \epsilon_2)$ packets after d time slots. For each coding scheme of this type, we define the following metrics:

1) *Complexity* for encoding/processing/decoding at nodes A, B and C: the number of operations required as a function of k, n_1 and n_2.
2) *Delay* incurred at the intermediate node B: this is the time $(d - k/C_{\text{mc}})$, where C_{mc} is the min-cut capacity. We will remark more on this notion of delay in Section II-A below.
3) *Memory requirement*: the number of memory elements needed at node B. Section II-A will also comment on the minimal memory requirements of any coding scheme over the line network.
4) *Achievable rate*: the rate at which information is transmitted from A to C. We say that a coding scheme is optimal in rate, if each individual link is used at a rate equal to its capacity. Thus it can achieve the min-cut capacity between the source and the destination.
5) *Adaptability*: whether the coding scheme needs to be designed for specific erasure probabilities ϵ_1 and ϵ_2 or not. Fountain codes, for example, are adaptable in this sense.

We observe that, although it is possible to design a code over a single link that is both adaptable and is optimized for achievable rate and delay, the overall coding scheme cannot be adaptable if we want to jointly optimize for achievable rate and delay. Indeed, assume that $\epsilon_2 = 0$. Then the scheme that jointly optimizes the delay and the achievable rate requires node B to transmit (forward) only when it receives a new packet. However, if ϵ_1 and ϵ_2 are equal and large, then a large fraction of the packets will get erased. In order to optimize for delay, node B should transmit about $\frac{1}{1-\epsilon_2}$ packets for each packet it receives, without waiting to receive the next packet from node A. Therefore a single scheme cannot be rate-optimal for both cases.

Depending on the application, different emphases might be placed on these performance metrics. For example, consider a real-time application, where information is collected into blocks of k packets that are encoded and sent over the channel. In other words, we want to transmit the real-time information from a source, as it is produced. Assume that we have M such blocks. Then the delay overhead at intermediate nodes can be considered to be a "set-up" delay for the connection, experienced only once, and hence insignificant if M is large. On the other hand, the memory requirements at intermediate nodes may be restrictive. Indeed, there might exist a large number of connections (paths) that share an intermediate node that performs processing. Thus, the memory available for each individual connection might need to be scaled down accordingly.

A. Optimal Delay and Memory Requirements

Recall that our notion of delay is linked with the optimal time of communication over a *single* channel with equivalent min-cut capacity. Note however that with this definition, it is impossible to achieve a 'zero delay' scheme even for the simple network of Fig. 1. In fact, even if both links AB and BC provide perfect feedback, there is an inherent delay to be suffered due to the existence of sequential links. As we will see, even in this perfect setting, there is also a need for memory storage, in amounts that grow with k. In this section we will calculate the memory requirements, as well as the minimal delay which is incurred when perfect feedback exists; certainly no coding scheme that does not rely on feedback can transmit in less time.

The obvious optimal scheme in the presence of feedback is one where each node repeats transmission of each packet until it is successfully received at the destination. Node A then completes transmitting in time $n \approx k/(1 - \epsilon)$. The operations at node B can be described using a Markov chain with states $x_i \in \{0, 1, 2, \cdots\}$, indicating the number of received packets still to be sent at each time; therefore at each time i, x_i packets need to be stored in memory. At each time (when $x_i \neq 0$), with a probability $1 - 2\epsilon(1 - \epsilon)$ the state is unchanged, and with a probability $2\epsilon(1-\epsilon)$, the state is increased or decreased by 1, with equal probability. Therefore, after n time slots, the dynamics of this system resembles that of a random walk with a reflecting boundary at 0, over $n' = 2\epsilon(1-\epsilon)n$ steps; (there is slight correction, due to 'longer stays' at state 0, but for large n, the probability of being at that state is insignificant.) Thus the expected value of x_n is the expected value of the absolute value of a random walk after n' steps. Therefore $\mathbf{E}[x_n] = O(\sqrt{n'}) = O(\sqrt{2\epsilon k})$, where we have used that $n \approx k/(1-\epsilon)$. Node B then completes transmitting the remaining x_n packets in a time $d \approx x_n/(1-\epsilon)$. Therefore, the 'delay' of this scheme is $O(\sqrt{\epsilon k}/(1 - \epsilon))$, while the expected memory requirement is $O(\sqrt{\epsilon k})$.

This argument can be extended to show that in linear network with L similar links, where L is a fixed finite number, each intermediate node incurs a delay of $O(\sqrt{\epsilon k}/(1 - \epsilon))$ and requires $O(\sqrt{\epsilon k})$ units of memory.

III. Coding Schemes

In this section we describe and compare a number of coding schemes for a line network with L links. In the next section we will discuss how these schemes can be extended to more general settings.

We will use the configuration in Fig. 1, with $L = 2$, as the illustrating example, and assume for simplicity that $\epsilon_1 = \epsilon_2 =: \epsilon$, in which case $n_1 = n_2 =: n$. In all schemes below we will use as code $\mathbf{C}_1$ over the link AB, a fountain code, such as an LT-code or a Raptor code; as demonstrated in [4] and [7], these codes are low complexity, rate optimal, adaptable codes over erasure channels. Then for each different coding scheme, we will specify the code $\mathbf{C}_2$ over the link BC. A summary of the properties of all these schemes will be provided in Table I.

A. Complete Decoding and Re-encoding

An obvious scheme is to use a separate code for each of the L links of the line network, and have each intermediate node completely decode and re-encode the incoming data. Then it is obvious that we can achieve the min-cut capacity by using optimal codes (e.g. LT-codes) over each link. However, the system suffers a delay of about $k\epsilon/(1-\epsilon)$ time-slots due to each intermediate node. Indeed, at node B, we can directly forward the $(1-\epsilon)n$ received coded bits without delay, and then, after decoding, create and send an additional ϵn bits over the second channel.

This straightforward scheme imposes low complexity requirements. We only need $O(k \log(k))$ binary operations at each intermediate node to decode and re-encode an LT-code, and the complete decoding and re-encoding scheme has memory requirements of the order $O(k)$. Moreover, LT-codes adapt to unknown channels in the sense defined previously.

B. Systematic Codes

The complete decoding and re-encoding scheme of the previous section is adaptable, rate optimal and has low complexity. However it requires each intermediate node to store in memory the entire k packets of information in order to re-encode. We propose a class of coding schemes, which we call *systematic schemes*, which minimize the memory requirement at the intermediate nodes, but require the knowledge of the erasure probabilities of the links.

Once again we consider the network in Fig. 1 and assume that we use a fountain code $\mathbf{C}_1$ for link AB. In a systematic scheme, the intermediate node B first forwards each coded bit (packet) from $\mathbf{C}_1$ as they are received; these are the systematic bits (packets). Meanwhile, B forms (about) $n\epsilon = \frac{k\epsilon}{1-\epsilon}$ linear combinations of the systematic bits, which are transmitted in the $n\epsilon$ time slots following the transmission of the systematic bits. Thus all systematic codes will incur an average delay of ϵn, and will require ϵn memory elements. The savings in memory, as compared to the complete decoding and re-encoding, is significant when the erasure probability ϵ is small.

In a linear network with L links, the same scheme can be repeated at each intermediate node. Since the operation at each intermediate node is rate-optimal, it follows that for each fixed L, the overall end-to-end transmission is also rate-optimal for large enough block length k, while each intermediate node requires about ϵn memory elements and contributes a delay of $n/(1-\epsilon)$.

Below we will discuss a few possible methods to design systematic codes.

1) Fixed Codes: Here we use a fixed systematic code, consisting of k systematic bits (packets) and $k\epsilon/(1-\epsilon)$ parity coded bits, to transmit the information over link BC. A systematic LT-code [7], or a Tornado code [3], for example, can be used to generate the parity bits, and in fact any fixed systematic code can be used for this purpose. Although not adaptable to unknown channel parameters, these codes have very low encoding and decoding complexities. Tornado codes for example can be encoded and decoded with $O(n \log(1/\delta))$ operations, where δ is a constant expressing the (fixed) rate penalty.

2) Sparse Random Codes: In this scheme, the non-systematic packets are formed as random (sparse) linear combinations of the systematic ones. More precisely, whenever a new packet is received at B, it is added to the storage space allocated to each of the non-systematic packets independently and with a (small) probability p.

Theorem 1: With $p = (1+\delta)\log(\epsilon k)/(\epsilon k)$ for $\delta > 0$, the described systematic random code asymptotically achieves the capacity over the channel BC.

Proof: [Sketch] Suppose $k' \approx k(1-\epsilon)$ systematic symbols are received at C, and let $l = k - k' \approx \epsilon k$. We will then wait for a further $l + c\log_2(l)$ non-systematic symbols to be also received at C, where $c > 1$ is a constant. After eliminating the received systematic symbols, these linear combinations can be described by a random $(l + c\log(l)) \times l$ binary matrix, with i.i.d. entries which are nonzero with probability $p = (1+\delta)\log(\epsilon k)/(\epsilon k)$. The results of [1] can be extended to show that, if $p > \log(l)/l$, the probability that such a matrix is not full-rank approaches zero polynomially fast with l. Using this and the law of large of numbers then, with high probability C can retrieve all the k symbols received at B, –e.g. by applying Gaussian elimination to this sparse matrix,– which can then be used to decode the fountain code $\mathbf{C}_1$. This code can decode the k information symbols from an average of $k + c\log(k\epsilon)$ received symbols at C, and hence this scheme rate optimal for large k. ■

The complexity of decoding this code is that of inverting the sparse $k\epsilon \times k\epsilon$ matrix, which is $O((k\epsilon)^2 \log(k\epsilon))$. In fact, it can be shown that $O(\log(k)/k)$ is the smallest possible value for the probability p, and equivalently the density of the non-systematic part of the code, if the code is to be decodable with negligible overhead. In that sense, the scheme provided here offers the lowest decoding complexity for any such random code where the parity bits are chosen as linear combinations of the systematic bits with i.i.d. distribution.

C. Greedy Random Codes

In this scheme, at each time slot the intermediate node B transmits random linear combinations (over $\mathrm{GF}(2)$) of all the

TABLE I

CODING SCHEMES THAT SEND k BITS FROM THE SOURCE TO THE DESTINATION.

Scheme	Intermed. node complexity	Delay	Memory	Adaptable	Rate Optimal
Optimal (Feedback)	0	$\sqrt{k\epsilon}/(1-\epsilon)$	$\sqrt{k\epsilon}$	yes	yes
Complete Dec-Reenc	$k\log k/(1-\epsilon)$	$k\epsilon/(1-\epsilon)$	k	yes	yes
Systematic Fixed	$k\log(1/\delta)/(1-\epsilon)$	$k\epsilon/(1-\epsilon)$	$k\epsilon$	no	yes
Systematic Random	$(k\epsilon)^2\log(k\epsilon)$	$k\epsilon/(1-\epsilon)$	$k\epsilon$	no	yes
Greedy Random	$k^2\log(k)$	$\sqrt{k\epsilon\log(k\epsilon)}/(1-\epsilon)$	k	yes	yes

packets it has received thus far.

The main advantages of this random scheme are its adaptability and optimality in terms of delay. The drawbacks are large memory requirement, and high decoding complexity, which is $O(k^2\log k)$ XOR operations on packets.

We will need the following proposition to analyze the optimality of greedy random codes.

Proposition 1: Given a constant $c>1$, let A be a 'random lower-triangular' $(k+c\log(k))\times k$ binary matrix, where the entries $A_{i,j}$ are zero for $1\le i<j\le k$, and all the other entries are i.i.d. $\text{Bernoulli}(1/2)$ random variables. Then

$$\Pr\left[\text{rank}(A)<k\right]\le\frac{1}{2k^{c-1}}.$$

Proof: Let K denote the right kernel of A, i.e.,

$$K:=\{\mathbf{x}\in\text{GF}(2)^k\mid A\cdot\mathbf{x}=0\}.$$

We will find the expected size of K. Let

$$V_i:=\{\mathbf{x}\in\text{GF}(2)^k\mid x_i=1,\text{ and for } j<i\ x_j=0\},$$

that is, V_i is the set of vectors which have their first 1-components at position i; then are 2^{k-i} such vectors. Let A_j denote the jth row of A. Then it is easy to verify that, for any $\mathbf{x}\in V_i$, the probability that $A_j\cdot\mathbf{x}=0$ is one for $j<i$, and is $1/2$ for $j\ge i$. Therefore the expected size of the intersection of V_i and K is

$$2^{k-i}\cdot(\frac{1}{2})^{k+c\log(k)-i+1}=\frac{1}{2k^c}.$$

The sets V_i for $i=1,\cdots,k$ partition $\text{GF}(2)^k\backslash\{0\}$, thus the expected size of K is

$$\mathbf{E}[|K|]=1+\sum_{i=1}^{k}\frac{1}{2k^c}=1+\frac{1}{2k^{c-1}}. \tag{1}$$

Now the expected size of the kernel can be used to bound the probability that A is not full-rank:

$$\begin{aligned}\mathbf{E}[|K|]\quad&=\textstyle\sum_{i=0}^{k}\Pr[\text{rank}(A)=k-i]2^i\\&\ge\Pr[\text{rank}(A)=k]+2\Pr[\text{rank}(A)<k]\end{aligned}$$

It follows that $\Pr[\text{rank}(A)<k]\le\mathbf{E}[|K|]-1=\frac{1}{2k^{c-1}}$. ■

An immediate consequence of Proposition 1 is that, if the channels were noiseless, i.e., $\epsilon=0$, then the greedy random coding scheme described above is rate optimal; this is because, with high probability, node C can perform Gaussian elimination on the generator matrix of the code $\mathbf{C}_2$, which is a random lower-triangular matrix of the type discussed in Proposition 1.

A closer examination of the proof of Proposition 1 reveals that, in order to make $\mathbf{E}[|K|]-1$ converge to zero, it is sufficient that, for each column i, the matrix A contains at least $k+c\log(k)-i$ rows with $\text{Bernoulli}(1/2)$ random variables at the ith position; this will then guarantee that the size of $K\cap V_i$ is no more than $1/k^c$, for some $c>1$, and we use (1) to obtain the desired result. The interpretation of this statement in the context of our coding scheme is that, in order to be able to decode with high probability at C, it is sufficient that for each $i=1,\cdots,k$, at least $k+c\log(k)-i$ packets are successfully transmitted over BC after B has received the ith coded packet from A.

Let α_d and β_d denote the number of packets successfully transmitted over links AB and BC respectively. Suppose now that we end transmission at a time n when C has received $k+l$ packets, i.e., $\beta_n=k+l$, where $l=o(k)$ will be appropriately chosen. Then the number of packets that will be received by C after a time d is equal to $k+l-\beta_d$; this, we would like to be at least $k+c\log(k)-\alpha_d$. In other words, the sufficient conditions above require that at each time $d=1,\cdots,n$, the quantity $\alpha_d-\beta_d$ be greater than $c\log(k)-l$. But $x_d:=\alpha_d-\beta_d$ behaves similar to a symmetric one-dimensional random walk: in fact, in $1-2\epsilon(1-\epsilon)$ fraction of the time slots, x_d remains unchanged, while in the other $2\epsilon(1-\epsilon)$ fraction, it increases or decreases by 1 with probability $1/2$. Therefore, in $n\approx(k+l)/(1-\epsilon)$ time it takes to complete transmission as described above, x_d's movements are identical to n' steps of a random walk $\{y_i\}$, where $n'=2n\epsilon(1-\epsilon)\approx 2k\epsilon$. Straightforward calculation then shows that with $l=O(\sqrt{n'\log(n')})=O(\sqrt{k\epsilon\log(k\epsilon)})$, the probability that $\{y_i\}$ at any time $i\le n'$ goes below $-l$ is polynomially small in k. This proves that, with high probability, the k packets of information can be retrieved at C from $k(1+\sqrt{\frac{\epsilon\log(k\epsilon)}{k}})$ received packets. The overhead goes to zero as k becomes large, and hence this coding scheme is asymptotically rate optimal.

IV. GENERAL NETWORKS

In this section, we will represent a communication network of binary erasure channels as a directed acyclic graph.

Assume for simplicity that all edges of the graph have the same capacity C_0. Consider a unicast connection; then the min-cut capacity between the source and the destination is

mC_0 for some integer m. It is straightforward to see that if we are employing a capacity-achieving coding scheme, it is sufficient to route the information along m parallel paths $P_1, \cdots, P_m$, where each path P_i consists of L_i links. We can then directly apply the coding schemes previously described to each path separately.

In practice, since coding schemes will employ codewords of finite block lengths, there might exist benefits in combining independent information streams [6]. Moreover, not all edges might be used at the same rate, for example because of cost considerations.

Consider a routing scheme that observes the flow conservation principle and utilizes each edge at rate smaller or equal to its capacity. Since all the component codes are linear, the received symbols along a link l in the network can be described using an $(n_l \times k_l)$ matrix, where k_l is the number of information symbols sent along the link, and n_l is the number of received symbols. The point we make in this section is that, as long as all such matrices corresponding to the intermediate links have full column rank, the end-to-end matrix that the receiver will have to decode in order to retrieve the information bits, will also be full rank and hence decodable.

Indeed, given the matrices associated with all individual links, to create the end-to-end matrix, we will have to perform the following types of matrix operations:

- Partitioning a matrix into parts, to create the equivalent matrix corresponding to splitting an input stream to multiple outgoing streams, such as node A in Fig. 2.
- Multiplication of matrices, in order to create the equivalent matrix corresponding to serially concatenated channels, such as nodes B and C in Fig. 2.
- Finding the direct sum of matrices, to create the equivalent matrix corresponding to merging multiple input streams of a node into a single outgoing stream, such as node D in Fig. 2.

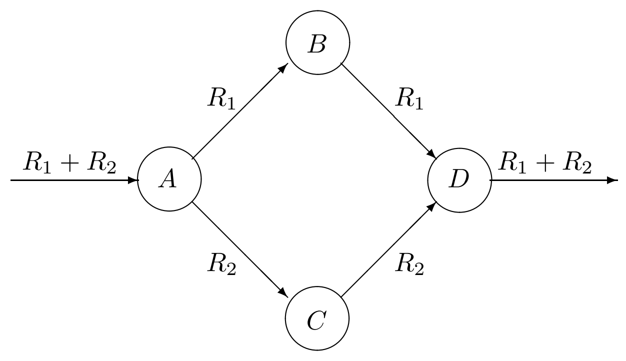

Fig. 2. Splitting and merging of information in the network.

All these operations preserve the full-rank property. Thus, all coding schemes described in Section III can be directly applied over a more general network. However, for this general case, a thorough study of the delay and memory requirements for each scheme is not provided here.

V. Comparison with previous work

In [8] a scheme was proposed that takes advantage of intermediate node processing to approach the min-cut capacity. The authors model the departures and arrivals at nodes as Poisson processes and work out the queuing-theory aspects of the problem. The employed coding scheme allows intermediate nodes to transmit random linear combinations of the incoming packets over a finite field $\mathrm{GF}(q)$. The transmission rate approaches the min-cut capacity as q goes to infinity. This scheme, as described in [8], requires $O(k^2(1-\frac{1}{q}))$ operations to encode k symbols at the transmitter, $O(k^3)$ operations for decoding at the receiver, and $O(k^2(1-\frac{1}{q}))$ operations at each intermediate node. Moreover, the operations are over $\mathrm{GF}(q)$ that are more complex than binary operations. Intermediate nodes require storage capabilities for k packets over $\mathrm{GF}(q)$.

The main benefit of the scheme in [8] is in terms of delay as we do not decode at each intermediate node. Indeed, complete decoding and re-encoding requires a delay of ϵn time-slots. However, note that the scheme in [8] achieves the min-cut rate for large q, i.e., assuming that we are able to send $\log_2(q)$ bits per time-slot instead of one bit per time-slot as we assume. Thus in this sense it is not clear that the comparison is fair.

In fact, the coding scheme employed in [8] can be thought as employing the greedy random codes in Section III-C, where the linear combinations are performed over $\mathrm{GF}(q)$ instead of the binary field, and where the encoding matrix is not sparse. Thus our results can be viewed as an improvement over the coding scheme proposed in [8].

VI. Conclusion

In this paper we have examined the problem of communication over a line network, where processing of information at the intermediate nodes is required in order to achieve the min-cut capacity, and we have included guidelines to extend our results to general networks. We have proposed coding schemes based on fountain codes. Each scheme has been analyzed and evaluated in terms of complexity, delay, memory requirement, achievable rate, and adaptability (see Table I). In general, there is a trade-off between these desirable properties, and an absolute best scheme is not claimed.

References

[1] J. Blömer, R. Karp and E. Welzl, “The Rank of Sparse Random Matrices over Finite Fields,” *Random Structures Algorithms 10*, 1997, pp. 407-419.
[2] T. Cover and J. Thomas, “Elements of Information Theory,” *Wiley-Interscience*, New York, 1991.
[3] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann. “Practical Loss-Resilient Codes,” In Proc. *ACM Symposium on Theory of Computing*, 1997, pp. 150-159.
[4] M. Luby, “LT Codes,” In Proc. *IEEE Symposium on the Foundations of Computer Science (STOC)*, 2002, pp. 271-280.
[5] D. Tuninetti, C. Fragouli, “Processing Along the Way: Forwarding vs. Coding”, *ISITA 2005*, Parma (Italy) Oct. 2004.
[6] D. Tuninetti, C. Fragouli, “On the Throughput Improvement Due to Limited Complexity Processing at Relay Nodes”, to appear, ISIT 2005.
[7] A. Shokrollahi “Raptor Codes”, Preprint available at http://algo.epfl.ch/index.php?p=output_pubs
[8] D. Lun, M. Médard, M. Effros, “On Coding for Reliable Communication over Packet Networks,” In Proc. *Allerton Conference on Communication, Control, and Computing*, 2004.