On the throughput improvement due to limited complexity processing at relay nodes

Daniela Tuninetti
University of Illinois at Chicago
daniela@ece.uic.edu

Christina Fragouli
EPFL
christina.fragouli@epfl.ch

Abstract— **We consider a source that transmits information to a receiver by routing it over a communication network represented by a graph and examine rate benefits that finite complexity processing at the intermediate nodes may offer. We show that there exist configurations where the optimal rate is achieved only when coding across independent information streams (channel coding and routing cannot be separated); that optimal processing is function of the particular set of channel parameters and not only of the network topology; that small constraint length suffices to achieve a large fraction of the ultimate network performance; and that there exists a connection between linear codes and routing for a special class of graphs.**

I. Introduction

The success of the Internet has made large scale communication networks part of everyday life. In the wireless world, ad-hoc and sensor networks promise to offer equally exciting applications. In a network environment the information hops through a number of intermediate relays before attaining the final destination, as opposed to a reach it directly through a single channel. The networked systems today employ traditional coding schemes for point-to-point connections that only require end-to-end processing and that are oblivious to the network environment.

In this paper we consider communication networks represented as graphs. Each edge in the graph represents an interference-free directed noisy channels. Relay nodes are allowed not only to forward the incoming information but also to process it. We are interested in evaluating possible benefits of intermediate node processing from an information-theoretic point of view in terms of overall network throughput.

We distinguish two cases based on whether or not intermediate nodes are subject to processing complexity constraints.

Perfect processing implies that intermediate nodes are allowed to decode and re-encode the information send by the source, without any complexity and/or delay constraints. Thus the use of a capacity-achieving channel code at each node transforms every link in the network into an error-free channel. It is then well known that for a unicast connection (a single source destination pair using the whole network) the min-cut max-flow capacity [**?**] is achievable [**?**]. Furthermore channel coding (how to cope against the channel noise) and routing (how to find the best way to the destination) can be separated without loss of optimality.

In the seminal work [**?**] it was shown that additional coding over these error-free links allows to better share the available resources and increase the overall network throughput to the ultimate min-cut capacity even in multicast connections (many receivers decoding a single source). This type of coding, performed across independent information streams, is referred to as *network coding*. Again, channel and network coding can be designed independently without loss of optimality.

Moreover, [**?**] provides examples of communication scenarios more complex than multicast and networks not restricted to directed links where channel and network coding can be separated and yet the ultimate information-theoretic capacity can be achieved. However, it is still not clear under which conditions separation is optimal.

Partial processing implies that intermediate nodes have complexity and delay constraints. We assume that source and destination can process long sequences of data bits and employ capacity achieving codes. However, intermediate nodes can only process "chunks" of N channel symbols. The blocklength N as measure of complexity allows us to bound the physical resources necessary for processing such as time and memory requirements. Moreover, it is well suited to environments where information is transmitted in packets.

Thus, network links can no longer be considered error-free. This case is the focus of this work and our main results are as follows.

- Channel coding and routing cannot be separated without loss of optimality. In this context, network coding can be thought of as coding across independent streams, while channel coding as coding over the same information stream. Moreover, channel coding is no longer independent of the network topology.
- Network coding offers benefits even in unicast connections over directed graphs. Even in this simple scenario, linear network codes are not optimal in general. That is, there exist configurations where non-linear processing across independent information streams increases the network throughput.
- Optimal processing at intermediate nodes is not only function of the network topology and of the communication scenario, but it also depends on the quality of the individual links. That is, for unicast connection over the same graph, changing the value of the channel parameter, i.e., increasing the noise level, results in a different optimal processing. A similar observation was made in [**?**].

- There exist connections between optimal routing over a specific class of graphs and linear error correcting codes. Thus, results from coding theory on the structure of the generator matrix can be translated into guidelines for selecting routes, and hence optimally combining, information streams.

We shall illustrated these points by examining a unicast connection over an example network configuration. Our results are to be considered a first step in understanding the *ultimate* performance limitations of interference-free multi-hop packet networks. In this respect, our work differs and departs from the main network coding streamline of work.

In a recent work [**?**], we examined how partial processing affects the achievable throughput of line networks using arguments based on error-exponents and worst-case/Fano channels. That is, we looked at finite but not too small N. Independently, a similar problem was also announced to be under examination for very small N in [**?**]. In [**?**] we mentioned the example network we look at in the present paper as future work.

The paper is organized as follows. Section **??** presents the network model and discusses known results that we are going to compare against in subsequent sections. Section **??** analyzes partial processing and presents results for optimal processing for a number of cases. Section **??** discusses the connections of linear error correcting code with routing. Section **??** points out our conclusions and future directions of work.

II. Model and Background Results

Throughout the paper we shall consider the example depicted in Fig. **??**. The model consists of a source and a destination node connected through a network represented as a directed graph.

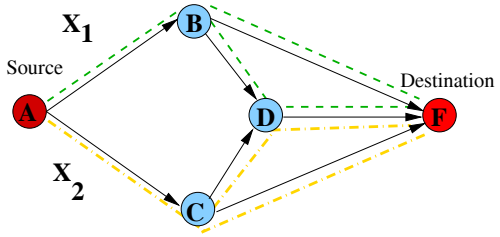

Fig. 1. An example of network.

The edges between intermediate nodes correspond to DMC (Discrete Memoryless Channels) used without feedback, indicated in the following as "$(\mathcal{X}, W)$ physical channels", where $\mathcal{X}$ denotes the input and the output alphabet and W denotes the channel transition probability matrix. In the examples, we shall consider the edges to be BSC(p) (Binary Symmetric Channels with transition probability $p \in [0, 1/2]$).

The source and the destination, located at nodes A and F in Fig. **??**, are not subject to any processing constraints. That is, the source can use a channel capacity achieving code and the destination can perform maximum likelihood decoding. The intermediate nodes, here B, C and D, are only allowed to perform *memoryless processing over "chunks" of N channel symbols from their incoming edges and output the same single block of N channel symbols on their outgoing edges.* N models the delay and memory limitations of the relays nodes. Depending on the value of N we distinguish three cases:

1) Perfect Processing ($N = \infty$): Intermediate nodes are allowed to decode and re-encode the whole codeword send by the source. We identify this case with $N = \infty$ as, in general, the capacity achieving code used by the source has extremely large block length.

   Example 1. *Consider each edge in the network in Fig.* **??** *to be a BSC(p). Fig.* **??** *shows the min-cut capacity as a function of p. Capacity is achieved by sending independent streams of iid equally likely bits on paths $P_1 = \{AB, BF\}$ and $P_2 = \{AC, CF\}$ (two parallel channels).*

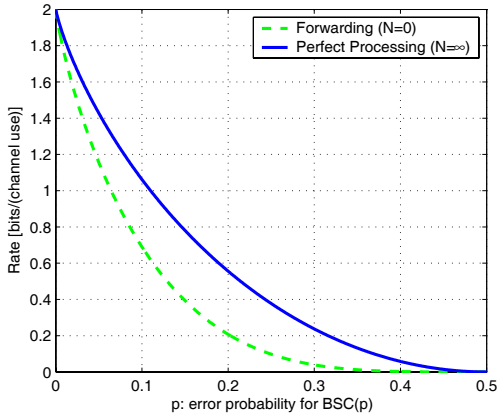

Fig. 2. Min-cut capacity and capacity with forwarding over the network in Fig. **??** when each edge represents a BSC(p).

2) Forwarding ($N = 0$): Each intermediate node is only allowed to forward the received information. If all edges of the network correspond to the same channel, then the length of a path indicates how good the overall channel is (the shorter the path the less noisy the channel). In this case the destination can receive multiple noisy observations of the same information stream from different incoming edges. It can hence optimally combine them to increase the overall throughput. That is, at intermediate nodes forwarding an information stream along branching paths that independently arrive at the receiver generates *path diversity*. Notice that forwarding is different from one symbol processing. In fact, for $N = 1$, node D could perform any operation involving its two inputs.

   Example 2. *Continuing the previous example, the paths $(AB,\ BF)$ and $(AC,\ CF)$ transport the information streams X_1 and X_2 as before. Additionally, path $(BD,\ DF)$ or path $(CD,\ DF)$ can be used to provide the destination with an additional observation of either X_1 or X_2. Fig.* **??** *plots the achievable rate with forwarding.*

3) Partial Processing (N finite): This is the case of interest in this paper, and that we shall discuss it in detail in

the next section. The motivation of this work is to get intuitions on how to bridge the gap between perfect processing and forwarding by using low-complexity intermediate processing.

Remark: Let (Y_1, Y_2, Y_3) be the output of $(BF,\ CF,\ DF)$ and (X_1, X_2) be the input of $(AB,\ AC)$. The overall channel is described by

$$P_{Y_1,Y_2,Y_3|X_1,X_2} = \sum_{Y_B,X_B,Y_C,X_C,Y_{D,1},Y_{D,2},X_D} P_{Y_B|X_1} Q_{X_B|Y_B} P_{Y_1|X_B} P_{Y_C|X_2} Q_{X_C|Y_C} P_{Y_2|X_C} P_{Y_{D,1}|X_B} P_{Y_{D,2}|X_C} Q_{X_D|Y_{D,1},Y_{D,2}} P_{Y_3|X_D}$$

where Y_n and X_n are, respectively, the information received at and sent by node $n \in \{B, C, D\}$. The processing at the intermediate node B, C and D are identified by the "equivalent channels" $Q_{X_B|Y_B}$, $Q_{X_C|Y_C}$ and $Q_{X_D|Y_{D,1},Y_{D,2}}$. Finding the channel capacity amounts to the maximization of $I(Y_1, Y_2, Y_3;\ X_1, X_2)$ over P_{X_1,X_2} and over all possible matrices Q. Since the mutual information is convex-$\cup$ in the channel transition matrix, and since we maximize with respect to the channels Q, it is clear that the optimal Q's correspond to **deterministic mappings**, i.e., each row of Q contains at most one non-zero element. In the following we shall only consider deterministic processing.

III. Partial Processing

With $N = \infty$ (perfect processing) and $N = 0$ (forwarding), information streams corresponding to independent data can be kept separated without loss of optimality when considering unicast connections. In the first case, because the min-cut capacity is achievable by routing the information streams through the paths traversing the graph min-cut. Those paths are distinct by definition of graph min-cut. In the second case, because only one information steam can be relayed through each edge of the graph min-cut due to the impossibility of processing information at intermediate nodes.

Example 3. *In a network of BSC(p) with min-cut M the ratio between C_0, the capacity with forwarding, and C_∞, the min-cut capacity, is*

$$\frac{C_0}{C_\infty} = \frac{1}{M} \sum_{m=1}^{M} \frac{C(p_i)}{C(p)}$$

where $C(x) = 1 + x\log_2(x) + (1-x)\log_2(1-x)$ is the capacity of each BSC(p) and $p_i = 0.5(1-(1-p)^{L_i})$ for $(L_1, ..., L_M)$ defined as the lengths of the M distinct shortest paths through the graph min-cut, which can be found, for example, with the Ford-Fulkerson algorithm.

Interesting questions are whether, given a small processing capability N, the ratio C_N/C_∞ can be made close to one and whether "mixing" independent information streams can be of any advantage.

To start with, in the example in Fig. **??**, assume that the nodes B, C and D are allowed to process only one bit and that all links are BSC(p). Without loss of generality, assume further $p < 1/2$.

It is easy to see that with $N = 1$, the optimal processing at nodes B and C is simply forwarding the incoming bit. At node D, since two bits are available at its input, one bit processing amounts to mapping four input values to two output values. It can be verified that only three functions at node D lead to different overall channels:

1) $f_1 = X_i$ for $i = 1$ or $i = 2$, i.e., send one of the inputs.
2) $f_2 = X_1$ XOR X_2, i.e., send the sum of the inputs.
3) $f_3 = X_1$ AND X_2, i.e., send the product of the inputs.

Let R_1, R_2 and R_3 denote the achievable rates when using, respectively, functions f_1, f_2, and f_3 at node D. Fig. **??** plots the ratios R_2/R_1 and R_3/R_1 when all channels introduce errors. From Fig. **??** it appears that:

- For the same network topology and the same unicast connection, the optimal one-bit processing is a function of the channel parameter p. In particular, for small values of p, superposition (f_2) outperforms forwarding (f_1) and non-linear processing (f_3). On the other hand, for large values of p forwarding is better.
- One-bit processing (f_2 or f_3) is not uniformly superior to forwarding/no-processing (f_1). When the channels are very noisy, combining different information streams does not increase the overall throughput.
- Since the network is composed of symmetric channels, i.e., bit 0 and bit 1 are (mis)treated equally by the channels, one would expect the uniform input distribution to result in better performance than a non-uniform one. However, it turns out that for very noisy channels, breaking the network "symmetry" by performing non-linear processing (f_3) and using a non-uniform input outperforms superposition (f_2) with uniform input.

In order to understand whether the above observations are pertinent only for the specific set of network parameters considered, let us assume that some links in the network in Fig. **??** are error-free. Fig. **??** plots the ratios R_2/R_1 and R_3/R_1 when all channels, but BD and CD, introduce errors.

Interestingly, from Fig. **??** we observe that for this new set of channel parameters non-linear processing (f_3) outperforms linear processing (f_1 and f_2) for values p roughly larger than 0.3. Furthermore, in this regime, the optimal input is two-valued, i.e., one input bit mapped either in the pair (00,11) or in the pair (01,10), and the optimal input distribution is not uniform.

By performing an exhaustive search among all possible deterministic mappings at node D, and by considering several combinations of the channel parameters, we observe that there exist three regime of operation as observed in Fig. **??**. Namely, superposition is always optimal for relative small values of the channel parameters (first regime) while non-linear processing is optimal for relatively large values of the channel parameters (third regime). In the middle/second regime forwarding is optimal. Degenerate cases are possible in which the second and the third regime 'collapses" to a single value of p (like in

the case considered in Example **??**). However, in the example of Fig. **??**, we never observed a different order of optimal processing.

Thus, unlike the cases $N = 0$ and $N = \infty$, the optimal routing is no longer edge-disjoint for independent information streams, which has a similar flavor to network coding.

We conclude the section with an example where analytical computation is easy and straightforward. This example shows that a simple one bit processing ($N = 1$) can greatly improve performance over forwarding ($N = 0$) in the sense that a given fraction of the ultimate network capacity ($N = \infty$) can be attained over a larger set of channel parameter. This confirms our intuition that **a little increase in complexity at the relay nodes achieves most of the ultimate network capacity**.

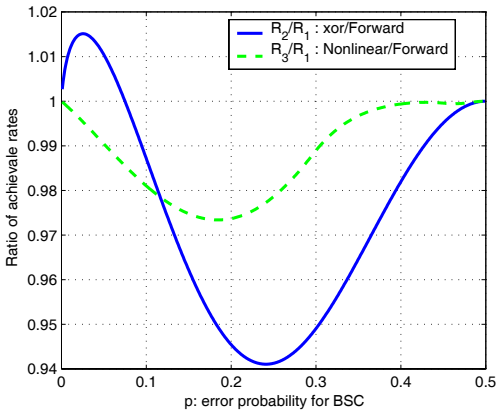

Fig. 3. Performance comparison for the network in Fig. **??** when each edge represents a BSC(p).

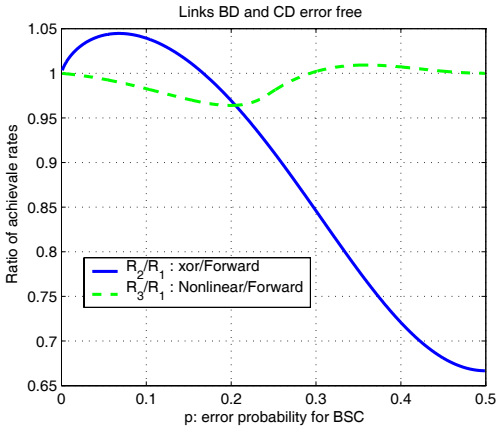

Fig. 4. Performance comparison for the network in Fig. **??** when every edge, except BD and CD which error-free, represents a BSC(p).

Example 4. *Consider the network in Fig.* **??** *when only edges BF, CF and DF are BSC(p) and the others are error-free. The achievable rate without using node D is $R_0 = 2C(p)$. The rates R_1 and R_2 achievable, respectively, with forwarding (f_1) and superposition (f_2), are*

$$R_1 = 3C(p) - C(\alpha), \quad \alpha = 2(1-p)p$$
$$R_2 = 3C(p) - C(\beta), \quad \beta = (1-p)^3 + 3(1-p)p^2$$

while the min-cut capacity C_∞, is

$$C_\infty = \min(2, 3C(p))$$

Exhaustive search over all possible mappings at node D shows that superposition is always optimal for $N = 1$, i.e., $C_1 = R_2$. The capacity with forwarding ($N = 0$) is $C_0 = R_1$.

Fig. **??** *plots the ratios R_2/R_1 and R_3/R_1 for the scenario under consideration. We notice that superposition is always superior to forwarding and non-linear processing. Fig.* **??** *shows R_1, R_2, together with R_0, and the min-cut capacity C_∞. Fig.* **??** *plots the ratios C_1/C_∞ and C_2/C_∞. Notice that one bit processing achieves 95% of the ultimate network capcity C_∞ for $p > 0.18$. On the contrary, forwarding achieves the same level of performance only for $p > 0.3$.*

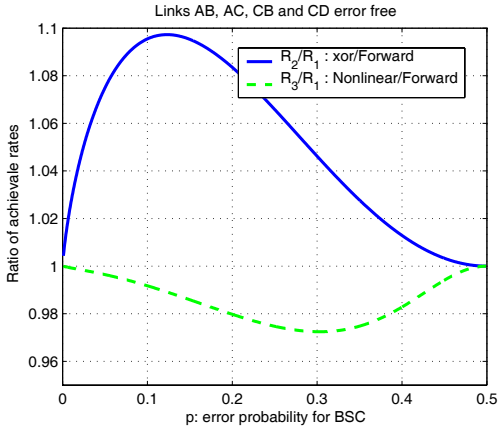

Fig. 5. Performance comparison for the network in Fig. **??** when only edges BF, CF and DF are BSC(p) and the others are error-free.

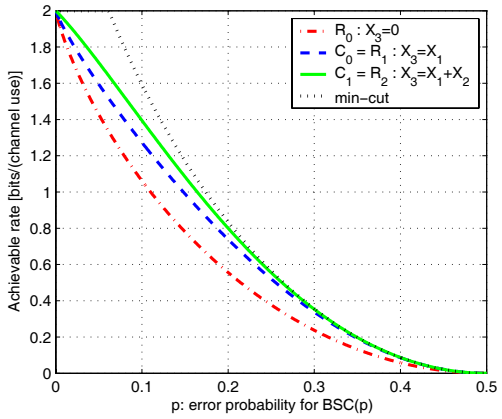

Fig. 6. Performance comparison for the network in Fig. **??** when only edges BF, CF and DF are BSC(p) and the others are error-free.

Finding suitable analytical tools of the analysis of the problem for larger values of N without need of exhaustive

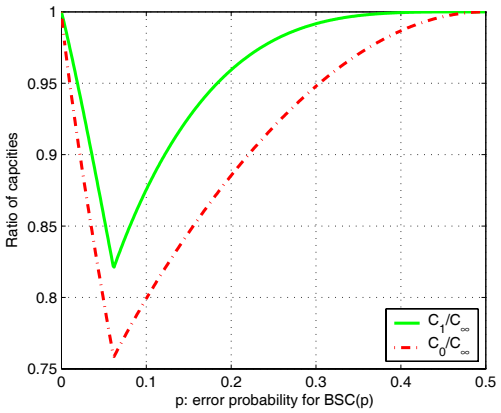

Fig. 7. Ratio of capacity C_N/C_∞ for $N = 0$ and $N = 1$ for the network in Fig. **??** when only edges BF, CF and DF are BSC(p) and the others are error-free.

numerical search is part of on-going work. An approach based on the error-exponent and on the worst-channel case as in [**?**] is under investigation.

The question is whether decoding and re-encoding at intermediate nodes is the a good communication strategy. The answer in not obvious as there are examples in the literature on relay channels [**?**] that shows strategies like "amplify and forward" or "compress and forward" are preferable to "decode and re-encode".

IV. Connections with Coding

From the previous discussion we saw that, for small values of the noise parameter, it is optimal to perform XOR at node D. We notice that those value of p are actually the ones at which commercial networks operate, thus the ones of greater practical importance. In the scenario analyzed in Example **??** XOR is optimal for all values of p.

In this latter case, the problem is equivalent to the problem of maximizing the composite capacity of a BSC(p) preceded by an encoder. The problem can be stated in more generality as follows. Consider a BSC(p) and design a code with k input bits and n output bits, such that the composite capacity of the encoder followed by the BSC(p) is maximized. Our case corresponds to $k = 2$ and $n = 3$. The optimal encoder is linear. In general, the optimal encoder is non-linear. If we restrict the intermediate nodes to perform linear operations, then the capacity is a function of the weight distribution of the code as discussed in [**?**].

The interesting point is that even when links AB, AC, BD and CD do introduce errors, provided that the noise parameter is small, the same intuition applies. Thus, the connection with linear coding, may allows us to translate results from linear code design to network codes and routing strategies, and to increase our intuition on the observed results. Good codes described in [**?**], could be mapped to network codes that perform linear operations over independent information streams to maximize the network throughput.

V. Conclusions

In this work we have shown that

- Network coding (mixing independent information streams) gives benefits for unicast connections over noisy channels.
- Channel coding and routing can not be separated. Depending not only on the network topology and the communication scenario, optimal coding and routing depend of the actually noise parameter, i.e., there is not intermediate processing that is uniformly better for any combination of channel parameters.
- For the network under investigation, it appears that linear processing is optimal for relatively small noise parameter while non-linear processing is optimal for large noise parameter. In the middle regime, forwarding is best. This observation suggests that limited complexity processing can greatly increase the network performance (over forwarding) for those values of the channel noise that are of practical importance in commercial network.
- The analysis for large values of N based on arguments similar to [**?**] indicates that large value of N are not needed. However, an open question is to analytically bound the gain provided by finite complexity processing for limited values of N without exhaustive search. Interesting approaches based on randomized strategies, very popular in the statistical physics and algorithmic communities, are currently under investigation.
- The generalization to other network topologies and other communication scenarios is currently under investigation. The interesting question is whether the insights gained with our small network example are valid for larger networks.

VI. Acknowledgments

We would like to thank Shlomo Shamai for valuable discussions and feedback.

References

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

[2] T. Cover and J. Thomas, "Elements of Information Theory," *Wiley Series.*

[3] D. Tuninetti, C. Fragouli, "Processing along the way: forwarding vs. coding", *ISITA 2005*, Parma (Italy) Oct. 2004.

[4] T. Cover et al., *presentation at private workshop after ISITA 2005*, Parma (Italy) Oct. 2004.

[5] G. Kramer and S.A. Savari, "On networks of two-way channels" *DIMACS Workshop on Algebraic Coding Theory and Information Theory*, Rutgers University, Piscataway, NJ, (USA), Dec. 2003.

[6] S-Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. Inform. Theory,* Vol. 49, pp. 371–381, Feb. 2003.

[7] S. J. MacMullan, O. M. Collins, "The Capacity of Binary Channels that Use Linear Codes and Decoders", *IEEE Trans. Inform. Theory,* Vol. 44(1), pp. 197-214, Jan. 1998.