ACCEPTED FROM OPEN CALL

Network Coding Mythbusting: Why It Is Not About Butterflies Anymore

Muriel Médard, Frank H. P. Fitzek, Marie-José Montpetit, and Catherine Rosenberg

ABSTRACT

Network coding has been shown to have radical implications for improving current network and storage systems. Because of its disruptive nature, both in terms of techniques and implications, it had naturally led to debate and confusion. This article seeks to dispel some of the misconceptions still associated with network coding, which we term its enduring myths.

INTRODUCTION

Network coding (NC) has evolved greatly since its formalization in [1], and has been steadily moving from mathematical concepts to implementations with different network technologies and architectures. However, in both analysis and implementation, the goals of network coding remain the same: enable the efficient transportation of information, using the full capabilities of the network layers and judiciously exploiting available storage and processing. This move is driven by the requirement to alleviate the strain on wireless resources at the edge. Indeed, the emphasis on content dissemination, the increasing heterogeneity of the network, and the need to enable efficient control of data in core and edge networks demands that we rethink radically and revisit the way networks are designed. NC is rapidly emerging as an important enabler for these new paradigms.

The basic principle of NC is to consider data in the network not as immutable bits, but as information that can be combined algebraically. Essentially, instead of storing and forwarding strings of zeroes and ones in the network, NC allows these strings to undergo algebraic operations [2]. While this progression may appear incremental, its consequences are drastic. In particular, the type of flow conservation principles, embodied for instance in Kirchhoff's Law, hold not in a physical sense, but rather in an algebraic domain in which degrees of freedom replace physical flows. There are several references that provide a good overview of NC from a theoretical and practical perspective, and we shall not attempt in this paper to provide a detailed overview, but simply address the aspects of NC that have sometimes led to confusion and its attendant construction of myths. We do not provide here an overview of where these myths can be found. Some of them, particularly Myths #1, #6, and #7, appear widely in the literature, sometimes as central themes of papers, more often as assumptions. Maybe more importantly, these myths have recurred in conversations with hundreds of students, in our classes and in tutorials, and with a great number of colleagues who have some acquaintance with network coding, have not yet had the opportunity to study it at length, and seek to obtain some helpful context to guide their exploration of the subject. It is to this latter audience that this paper is addressed.

MYTH #1: NETWORK CODING REQUIRES "BUTTERFLY" TOPOLOGIES

The original paper of Ahlswede *et al.* [1] provided a small useful example of its operation over a topology that has become affectionately known as "the butterfly." Figure 1 shows that example. In there, two units of information, b_1 and b_2, which may be mere bits or entire packets, are to be sent from the topmost node S to the bottom two nodes Y and Z in a multicast fashion. Assume, for the sake of simplicity (and we shall see in the course of this paper that this assumption is not at all necessary) that each link provides an erasure and error-free link that carries a single unit. The first step to be taken is fairly simple: b_1 can be sent down to node T, which sends it on its two outgoing links, and b_2 can be sent down to node U, which also sends it on its two outgoing links.

The question is then: what should be done at node W? If it transmits b_1 first then b_2, Z will receive b_1 before Y receives b_2. Hence it can be seen that the multicast rate to Y and Z is 1.5 when W is not allowed any processing on the data. The solution provided by network coding is to use the algebraic nature of the data to resolve the competition for resources between Y and Z at W. By allowing W to send a combination of b_1 and b_2 to X (in this case, just a simple sum), in the absence of other data flows, Y or Z would be able to obtain the full rate of two by "decoding," that is, recovering from this combination, the information that they do not have. We do not detail here what processing the intermediate node would do in a general network: this could be an addition (in fact a linear combination of possibly more than two packets) over some large finite field, or could be bit-wise XORs over entire packets. While the original paper on alge-

Muriel Médard is with MIT RLE.

Frank H. P. Fitzek is with Aalborg University.

Marie-José Montpetit is with MIT Media Lab.

Catherine Rosenberg is with the University of Waterloo.

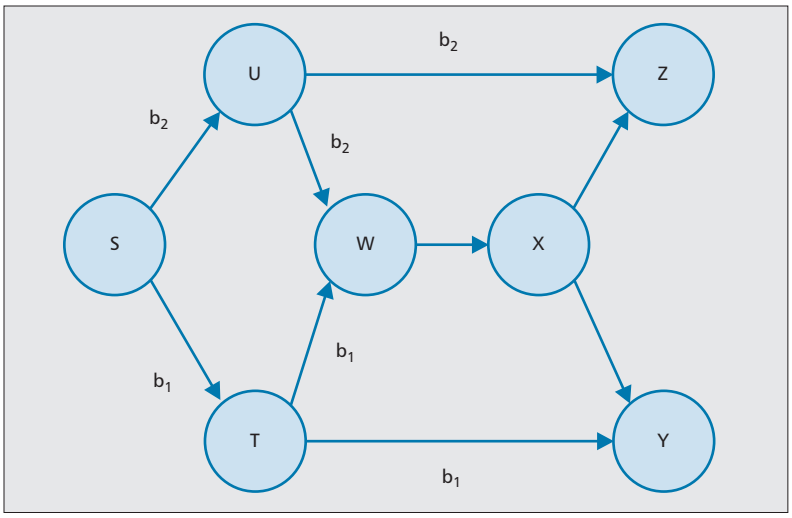

Figure 1. Butterfly example.

braic NC [2] certainly assumes knowledge of finite algebra, many practical solutions are applying codes which, compared to channel codes implemented over noisy links, are actually quite trivial.

The simplicity and clarity of the above example is very compelling but it has, unfortunately, prompted some shallow interpretations of NC. NC is not about finding butterflies in a network or re-engineering networks to create butterflies. Indeed, the problem of optimizing the use of resources for network-coded multicast [3] has nothing to do with looking for butterflies. The main reason for looking at butterflies seems to be the desire to apply the simple additive code shown above for academic illustration. As will be seen in the remainder of this paper, general topology networks (not just multicast) can benefit from NC that has been implemented in network services like storage and streaming. In essence, NC can be described as follows: if any receiver of a multicast session could, in the absence of other receivers of the same session, receive a certain rate, then it can do it in the presence of any other number of receivers. NC removes the competition among users for finite resources.

Myth #2: Network Coding Requires Complicated Codes And Waiting for Packets

NC does not need to be constructed deterministically at all as in Fig. 1 for a multicast topology. Instead, it can be constructed completely randomly in a distributed fashion using random number generators and resulting in what is now known as a "random linear network code (RLNC)" [4]. But a misconception associated with RLNC is that it requires large finite fields to be efficient, hence reducing its usefulness. This "Myth" comes from a perfunctory reading of the mathematical proof in [4]; it shows that the probability that a random code will fail to operate properly decreases with the field size. But it is also shown in this proof that the decrease is very steep, indeed exponential, in the number of bits used in the code. This means that, in practice, codes of the order of eight bits or so have been found to be quite satisfactory. However, under conditions other than multicast, it has also been shown theoretically that random codes may not be adequate and that linear codes may not be sufficient [5]. However, even then, fairly simple ad-hoc approaches can outperform uncoded approaches. We shall present one such example when we consider Myth #5.

A related misconception to the code construction presented above is in the encoding process itself: people have considered using NC in a way that requires coding to process packets together synchronously, hence holding packets in a queue until there are enough packets to satisfy the code's algebraic equation. This puzzling myth seems to be a holdover from physical-layer block codes, where all the symbols associated with a block must be received before encoding and decoding can occur.

In channel codes, block codes are often used to average out erasure errors over a certain number of symbols. Such an approach is *not necessary* in packet networks with NC. Indeed, each packet can easily bear in its header the coefficients of the packets from which it is formed. A late packet can simply be modeled as having a null coefficient associated with it. Thus, there is no need for a preassigned code and for waiting for packets to complete a block. In general, even in networks with erasures, packets may be combined without having a fixed coding rate (ratelessly) to achieve maximum throughput, without any need for coordination or knowledge of the erasures that occurred in the network [6]. There is also no need to keep in memory a large number of packets to maintain randomness, but rather very little memory suffices.

To further illustrate this, take Fig. 1 as an example. Consider a system in which a single packet can be transmitted in a time slot. If packet b_1 arrives before packet b_2, say at discrete time slot 1, it can be forwarded in the absence of packet b_2. If instead we implement coding in a way that waits for packet b_2, the latter may arrive at time 2 together with some packet a. If packet b_1 was held back until time 2, it could be combined with packet b_2 when b_2 arrives in time 2, but packet a cannot be sent in time 2 and has to wait until time 3. If packet b_1 had been sent in time 1, then another packet, say a, and b_2 could have been coded together in time 2, achieving a higher total throughput and a lower average delay.

Myth #3: The Maximum Throughput Gain of Network Coding is Two in Wireless Networks

While a throughput gain of a factor two would be very good, in fact the actual throughput of network coding in a wireless network can be much larger. This very common and stubborn myth of the seemingly hard limit at two, comes

from a misunderstanding or misreading of the work of [7]. This work showed that for undirected networks, the NC gain is at most two. The reasoning is that it must be the case that the maximum expected gain of NC in wireless networks is also two since they are undirected. But we are talking about different types of undirected networks. Li and Li [7] refer to networks that are "graph-theoretically undirected," which means that links can operate in two directions in a time-shared fashion per link. Wireless networks are not undirected in that sense, they simply have links that can operate in two directions in a time-shared fashion per node. This is a major difference. In a graph-theoretically undirected network, a node, say node 1, can be talking to node 2 for 30 percent of the time and receiving from node 2 for 70 percent of the time (the link [1, 2] is used for 30 percent of the time in the direction 1 to 2), while talking *simultaneously* to node 3 for 80 percent of the time and receiving from node 3 for 20 percent of the time (the link [1,3] is used 80 percent of the time in the direction 1 to 3). In a wireless network, this is not possible: a node cannot use several directed links simultaneously (except for broadcast, but this is then unidirectional).

NC gains can easily exceed a factor of two in wireless. For illustration, let us consider the COPE system [8] as an example. Consider Fig. 2, reproduced with slight changes by permission of Dina Katabi and Sachin Katti. Node D needs to undertake, in an uncoded system, three transmissions, with next hops shown with dashed lines in the colors of the packets to be forwarded in Fig. 2 (top). Because of the broadcast nature of the wireless medium, some packets have been already overheard and stored: those overheard packets are shown in the queues of nodes A through C. By coding together three packets, we are able to reduce the three transmissions from D to just one, as shown in Fig. 2 (bottom) as the coded packet contains enough information (degrees of freedom) to reconstitute all packets at their destination. The gains that have been shown on 802.11 systems through application of COPE, which considerably reduce congestion at certain nodes (such as node D), can be of more than one order of magnitude.

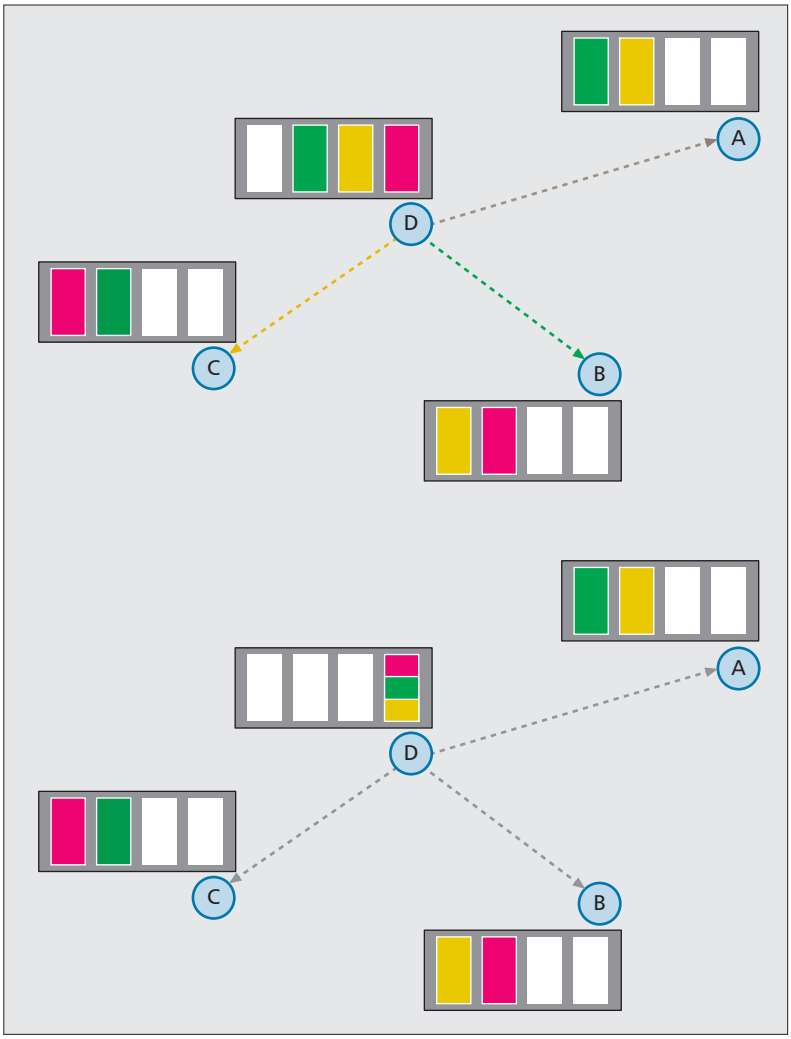

Figure 2. COPE example (a) top and (b) bottom.

Myth #4: NC is Only for Wireless Networks

The existence of this myth may seen to be very odd given Myth #3, since the latter myth would seem to indicate that network coding has limited applicability in wireless systems. NC's gains are not limited to wireless settings. Considerable gains have been shown in network coded overlay networks and for peer-to-peer (P2P) and distributed storage applications [9, 10]. Moreover, other types of coding, such as Fountain codes, which will be discussed in Myth # 6, have already been shown to be useful in settings other than wireless.

Let us consider the case of distributed storage in further detail. If NC is used, reconstruction of a file consists of recovering algebraic equations that will eventually lead to a full rank matrix that can be inverted; this is the principle of network coding. Random coding, as well as certain structured codes, will naturally provide equations that are, with very high probability, not redundant and will lead to successful decoding. However, when collecting uncoded pieces of information, the probability that a node may bear a piece that the receiver has already collected increases naturally with the number of pieces at the receiver. This problem is the classical *coupon collector*, well known to anyone who has ever started a collection. When starting a collection of stamps, sports cards or others, one begins easily accumulating new elements without getting doubles. As the collection increases, the time to obtain a missing element increases steeply. NC removes that dependence on the accumulated elements [11]. Scaling benefits can be seen in settings where many nodes seek to disseminate information to some portion of the network's nodes. In that case, the use of distributed random coding without a scheduler can match the $O(n)$ disseminations time performance of a perfect centralized scheduler [11], whereas $O(n \log (n))$ dissemination time is needed in the absence of a scheduler and coding. In addition, NC results in a much more resilient and stateless implementation: the loss of

The fact that NC can reduce performance is not, strictly speaking, a myth, since people have done implementations of NC that are not successful. The myth here is that coding is intrinsically worsening performance, whereas the problem lies in incorrect implementation.

scheduling information (or pointers) can lead to catastrophic failures to recover the information; with NC none of this information is required.

Myth #5: Performance Can Worsen with NC

The fact that NC can reduce performance is not, strictly speaking, a myth, since people have done implementations of NC that are not successful. The myth here is that coding is intrinsically worsening performance, whereas the problem lies in incorrect implementation.

We shall not cite instances of unfortunate approaches to coding, but simply show some representative examples. Many start from COPE and misapply it; for example, attempting to *create* congestion in order to lead to coding opportunities. Imagine, for instance, burdening artificially node D in Fig. 2 in order to allow combinations of many packets. This is akin to causing injury to illustrate the benefits of a new remedy. COPE can help alleviate congestion, but that is no reason for causing it in the first place. Another example involves using COPE in wireline settings, mimicking wireless through many extraneous transmissions intended to provide overhearing. Again, COPE makes use of overhearing on a broadcast channel, but that does not mean that we should seek to create broadcast channels when they are not there naturally.

Another common mistake is to negate the gains of NC in peer-to-peer or distributed storage by first selecting the preferred nodes from which to download and then coding over those nodes only. In fact, the gain that network coding provides, as sketched in our discussion of Myth #4, is that they avoid the coupon collector problem by ensuring that the requisite degrees of freedom are, with high probability, available at any minimum number of peers. Obviating the need to seek specific pieces of content is the core of the benefit that NC brings to such systems. Applying coding after the choice of peers or storage nodes has occurred fails to make use of that benefit, since a priori it is not known that the information in these nodes will be sufficient to recreate the files unless we use pointers or other extraneous information.

Another misconception, which relates to Myth #2, is to force encoding and re-encoding at every node on a path. Since network codes are linear packets, they can be recombined in the network without first decoding them (network codes are *composable*). This leads to efficient P2P implementation. Forcing the decoding at each node may reduce throughput, in some cases significantly, as will be illustrated in Myth #7. Even in cases where no loss of throughput occurs, it may lead to needless computational complexity and added delay.

While the above examples could be dismissed as simply poor network design, predictable from a sound understanding of the theory, there also exist non-trivial technical difficulties that can stymie implementation even when the design is well thought out. Some of these issues do not become apparent until implementation and require careful consideration. We provide below a brief list of pitfalls in the implementation process (we humbly note that we encountered them *all*):

- Random linear network coding (RLNC) is based on random numbers. Hence, implementations on some operating systems like Windows can fail because they use a standard library that does not provide sufficient randomness. This phenomenon, which does not affect all applications, is known to implementers of security mechanisms and does affect network coding generally. However, care must be taken to ensure sufficient randomness in RLNC.
- In implementations of RLNC with unnecessarily large Galois field, coding becomes too slow and overall performance degrades. In most applications, small fields, often even binary, suffice. There is a trade-off between the theoretical gain that is obtained by growing the field size and the practical drawbacks of working over larger fields or lengthy bit vectors in terms of complexity and delay, which in turn affect potential coding gains, as we discuss in the next point. In general, the complexity of the code should be controlled and linked to the expected behavior of the targeted service or application.
- Coding potential needs to be present. We have argued that we should not create coding possibilities at all costs. Designers must decide judiciously when and at which nodes to code. For example, transmission energy is taken into account, there may be a reason for holding back a packet in order to code it with later packets even if, as discussed in Myth #2, this is not a necessary condition for NC to work.
- Coding requires eventual decoding. Some implementations might code locally at intermediate nodes without ensuring that receivers can decode. This poses a particular challenge since coded packets that have not been decoded bear very little useful information about the original payload and, indeed, can be thought of as a very effectively encrypted version thereof (and be exploited as such).

Myth #6: Network Coding is Like a Fountain Code

A Fountain code is a rateless end-to-end erasure code. As such, packets at intermediate nodes inside the network cannot be combined. Thus, Fountain codes are not composable. On the other hand, network codes are indefinitely composable over any portions of the data path. While one could envisage constructing network codes that have a structure akin to Fountain codes in a networked setting, such codes would generally require accurate foreknowledge of the network and its current operating states. This distinction is of great importance operationally, but also in terms of performance. Consider, for instance, a system of two erasure channels in series. In the absence of erasures, each channel can transmit a single packet per time slot. However, each channel suffers from independent erasures which occur with 0.5 probability. If the source node and middle node perform a RLNC, the throughput is 0.5 packets per time slot. With an end-to-end code, the throughput is 0.25. If we

increase the number of such channels to n, the throughput with RLNC remains at 0.5, whereas that of an end-to-end code decreases exponentially with the number of channels to $(0.5)^n$. Figure 3 illustrates such a network composed of relays in tandem.

One could of course argue in the above case that erasure-correcting block codes on each link could also have achieved the optimal throughput, with each intermediate node coding and decoding. While this approach would eventually be poor in terms of average delay, particularly as the number of nodes increases, it would indeed perform as well as RLNC in terms of throughput. However, the weakness of requiring intermediate decoding can readily be seen, if we consider a system such as that shown in Fig. 4a, (reproduced by permission of Katti *et al.* with slight modifications). The numbers indicated on the wireless links represent the probability of losses. If we assume a single route, an end-to-end code would achieve a throughput of 0.25, as shown in Fig. 4a. We could of course exploit the broadcast nature of the wireless medium. In this case the relays would receive the packets by the source and forward those successfully received before. This approach, in the absence of coding, will transmit duplicate information from relay nodes R_i. One would need careful coordination among nodes, relying on consistent knowledge of their topology, interactions and, possibly, memory contents, in order to reduce duplication. But interaction among the relays is costly and hard to realize in commercial platforms as the IEEE 802.11.

If on the other side RLNC is used, all relays are able to receive the packets from the source as given in Fig. 4b. In order to avoid duplicates each relay will recode the incoming packets before forwarding them to the sink node. Doing so, no overwhelming coordination among the relays is needed. The only scheduling policy for the relays is to check whether they have something meaningful to say, for example, if the rank of the relay information matrix is larger than the sink's rank.

Myth #7: Network Coding Yields no Gains in Large Networks

The results for gossip network of [11], presented in the context of Myth #4, as well as the examples for networks scaling in depth or breadth discussed in Myth #6, illustrate the fact that the gains from NC can increase with the number of nodes in a network. Paradoxically, scaling results have been invoked to state that NC cannot improve network performance. These results generally consider the way in which the throughput point-to-point connections evolves in lossless large random networks as the number of nodes increases. Such scaling results are, at their core, arguments around bottlenecks and NC can alleviate bottlenecks (but not remove them), thus providing gains that may not be reflected in coarse order arguments. Moreover, scaling benefits may sometimes grow with parameters other than the number of nodes. The authors in [12] showed that NC may provide a gain in download delay which scales *with file size* but not in the number of nodes.

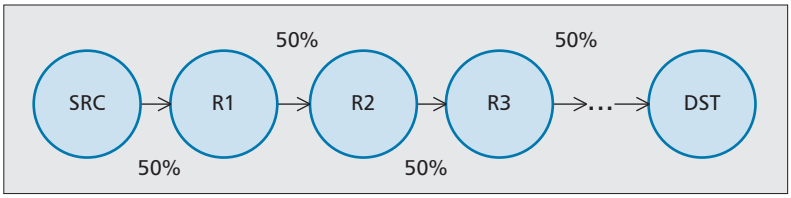

Figure 3. RLNC with nodes in tandem.

Myth #8: Acknowledgments Obviate the Need for NC

The example in Fig. 4 also helps to illustrate another persistent myth, namely that Acknowledgments obviate the need for coding. This myth probably originates from a simple and sensible remark. In the case of a point-to-point link with x percent erasures, the throughput is $100 - x$ percent, and it can be achieved either by coding using a traditional erasure-correcting code, or by repeating packets using a retransmission scheme. However, this remark does not provide the correct intuition in networks.

In the case of end-to-end acknowledgments, the first example given in Myth #6 immediately demonstrates that end-to-end acknowledgments will not, by themselves, lead to the same throughput as network coding. Indeed, end-to-end acknowledgments and fountain codes both consider the entire network operating as a single link per receiver and, by ignoring the capabilities of intermediate nodes, do not permit full usage of network capacity.

A more subtle point emerges when we consider perfect acknowledgments on a link by link basis. Let us revisit the first example invoked in Myth #6, that of a series of erasure channels in a daisy chain. If acknowledgments consumed no resources, such as time or bandwidth, and experienced no erasures (unlike the forward links), and if nodes were perfectly coordinated, then we could rely uniquely on such acknowledgments and we would not need coding. Moreover, the issue we raised at the end of our discussion of Myth #7, that of excessive use of resources in some links when we use end-to-end codes, does not arise. For the example of a perfect link followed by a link with probability of erasure 0.5, the first link would not need to carry redundant packets. However, the situation changes if we consider the network represented in Fig. 4. Acknowledgments might be useful but in general would not allow us to solve the ornery issues around relay node coordination that we mentioned in the context of Myth #7. Thus, they would not lead to maximum throughput. In any case, idealization of acknowledgments as requiring no network resources and having no erasures of their own do not generally hold, unless the forward channels are themselves free of erasures. It is now known that in some wireless networks, such as cellular telephony systems, the acknowledgment channels themselves become

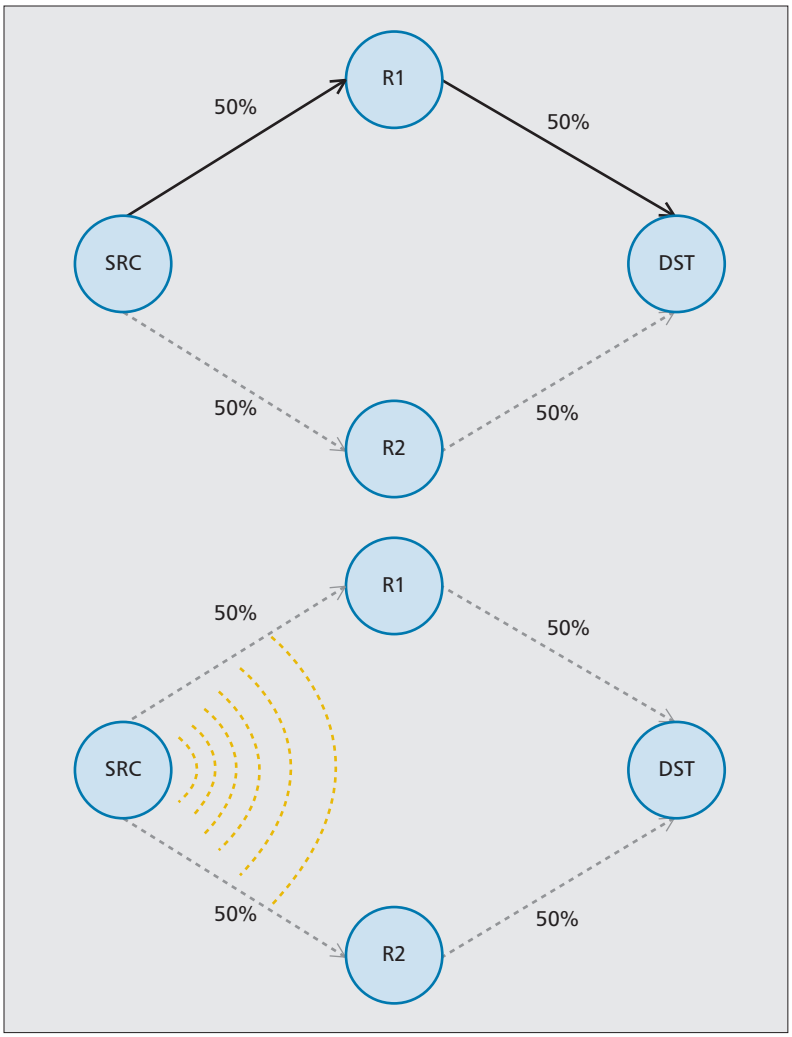

Figure 4. a) top: no coding and b) bottom: RLNC.

bottlenecks, exasperating efforts to improve throughput in times of congestion.

Myth #9: There Are No Practical Applications of NC

The last myth deals with the practical application of network coding or whether it can be realized on embedded systems at all and is the introduction to the second paper in this series. Indeed, until very recently NC was in the realm of advanced mathematics and theoretical networking. However, even in its early days it had shown considerable value for file transfer [13] and peer to peer networking [14]. More recently, a number of startups such as Steinwurf in Denmark and Streambolico in Portugal have also been launched to exploit the advantages of NC in commercially deployed devices and networks.

Network coding is also compatible with existing transport protocols. Recently, TCP/NC has been shown to be compatible with TCP [15] but improves its performance greatly over wireless links. This work inserts a NC layer between the TCP layer and the IP layer. Hence, the approach is independent of the underlying network technology. This creates a way for TCP to resist channel impairments over any network with low SNR be it cellular, WiFi, or wired technologies like powerline. With the current video explosion over the Internet, further applications of TCP/NC are emerging. One of the most interesting is web and TCP acceleration [15] where TCP/NC enhances the performance of video-rich applications by close to one order of magnitude in congested conditions and to extend the range of WiFi and broadband wireless cells.

To bust this myth further, we refer to the implementation of network coding on commercial mobile smart phone platforms, as presented in [16]. In this work NC was implemented for iOS. The scenario focuses on proximity sharing of video streaming [16] among mobile phones. The content to share is stored on a single mobile phone but should be displayed simultaneously on several neighboring devices using IEEE 802.11 ad hoc mode. The advantage of using NC in such a scenario is threefold. First, it helps to solve the coupon collector problem. Furthermore, the sharing among mobile phones, realized without any overlay network, can be extended to devices that only can be reached by multi-hop. Another benefit of using network coding is that each device that participates in the sharing is enabled to help the user community after receiving the first coded information, thus, reducing the burden on the original source.

Conclusion

We have intentionally kept the number of myths that we addressed very small and we hope the reader by now will be able to bust his or her own favorite myth. A partial list of debunkable myths that go beyond the scope and length of this article include: the need for excessive overhead; the need to coordinate NC with physical layer codes; that NC is only for multicast; that NC suffers from pollution attacks; that one can replace NC with storage at nodes, etc. We hope that after reading this paper network architects and engineers will be encouraged to invest in NC to meet the challenges from the rapid growth of multimedia traffic, and the need to accommodate new applications in a highly mobile and heterogeneous eco-system.

Acknowledgments

The authors would like to acknowledge the team of graduate students, post-doctoral fellows, and colleagues without whom this research would not be possible, in particular: Nadia Fawaz, Soheil Feizi, Kerim Fouli, Janus Heide, MinJi Kim, Ali Parandehgheibi, Morten Pedersen, and Shirley Shi.

References

[1] R. Ahlswede *et al.*, "Network Information Flow," *IEEE Trans. Info. Theory*, vol. 46, no. 4, 2000, pp. 1204–16.
[2] R. Koetter and M. Médard, "An Algebraic Approach To Network Coding," *IEEE/ACM Trans. Net.*, vol. 11, no. 5, 2003, pp. 782–95.
[3] D. Lun *et al.*, "Minimum-Cost Multicast over Coded Packet Networks," *IEEE/ACM Trans. Net. (TON)*, vol. 14, 2006, pp. 2608–23.
[4] T. Ho *et al.*, "A Random Linear Network Coding Approach to Multicast," *IEEE Trans. Info. Theory*, vol. 52, no. 10, 2006, pp. 4413–30.

[5] R. Dougherty, C. Freiling, and K. Zeger, "Insuffciency of Linear Coding in Network Information Flow," *IEEE Trans. Info. Theory*, vol. 51, 2005, pp. 2745–59.
[6] D. Lun *et al.*, "On Coding for Reliable Communication over Packet Networks," *Physical Commun.*, vol. 1, no. 1, pp. 3–20, 2008.
[7] Z. Li and B. Li, "Network Coding: The Case of Multiple Unicast Sessions," *Proc. 44th Annual Allerton Conf. Commun., Control, and Computing*, Sept.–Oct. 2004.
[8] S. Katti *et al.*, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Trans. Net.*, vol. 16, no. 3, 2008, pp. 497–510.
[9] S. Acedánski *et al.*, "How Good is Random Linear Coding Based Distributed Networked Storage?" *Proc. 1st Workshop on Network Coding, Theory and Applications (NetCod)*, April 2005.
[10] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution," *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005.
[11] S. Deb and M. Médard, "Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering," *Proc. Allerton Conf. Commun., Control, and Computing*, Sept. 2004.
[12] A. Eryilmaz, A. Ozdaglar, and M. Médard, "On Delay Performance Gains from Network Coding," *Proc. 40th Annual Conf. Information Sciences and Systems*, 2006, Princeton, NJ USA.
[13] P. R. C. Gkantsidis and J. Miller, "Anatomy of a P2P Content Distribution System with Network Coding," *IPTPS*, 2006.
[14] Z. Liu *et al.*, "UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding," *Proc. INFOCOM,* 2010.
[15] J. Sundararajan *et al.*, "Network Coding Meets TCP: Theory and Implementation," *Proc. IEEE*, Mar. 2011.
[16] P. Vingelmann *et al.*, "Synchronized Multimedia Streaming on the iPhone Platform with Network Coding," *IEEE Commun. Mag.* — Consumer Communications and Networking Series, June 2011.

Biographies

Muriel Médard [F] (medard@mit.edu) is a Professor in EECS at MIT, where she received all her degrees. She was awarded the IEEE Kirchmayer Prize, the IEEE ComSoc/IT Joint Paper Award, and the IEEE William R. Bennett Prize. She received the MIT Edgerton Faculty Achievement Award and was named a Gilbreth Lecturer by the NAE. She has been TPC co-chair of several conferences and served as President of the IEEE Information Theory Society.

Frank H. P. Fitzek (ff@es.aau.dk) is a Professor in the Department of Electronic Systems, University of Aalborg, Denmark. He was awarded the NOKIA Champion Award seven times, the NOKIA Achievement Award (2008), the Danish SAPERE AUDE research grant (2010), and the Vodafone Innovation price (2012). His research focuses on wireless and mobile networks, mobile phone programming, network coding, cross layer and energy efficient protocol design, and cooperative networking.

Marie-José Montpetit [SM] (marie@mjmontpetit.com) is a Researcher with the Comparative Media Studies at MIT. From 2009-2012 she was the Research Laboratory of Electronics at MIT and before was a researcher at Motorola. Her research interests are include Internet television systems, advanced video protocols and mobile networks.

Catherine Rosenberg [F] (cath@uwaterloo.ca) is a Professor in Electrical and Computer Engineering at the University of Waterloo. Since June 2010 she holds the Tier 1 Canada Research Chair in the Future Internet. From 1999 to 2004, Prof. Rosenberg was a Professor in the School of Electrical and Computer Engineering at Purdue University. She is a Fellow of the Canadian Academy of Engineering. Her research interests are mainly in two areas: Wireless Networking and Energy Systems.